VICTOR DE OLIVEIRA MATOS

ALGORITMOS HEURÍSTICOS PARA FORMAÇÃO DE CLUSTERS 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

M433a
2013

Matos, Victor de Oliveira, 1987-

Algoritmos heurísticos para formação de clusters em redes de sensores sem fio / Victor de Oliveira Matos. – Viçosa, MG, 2013.

xiii, 62f. : il. (algumas color.) ; 29 cm.

Orientador: José Elias Claudio Arroyo.

Dissertação (mestrado) - Universidade Federal de Viçosa.

Referências bibliográficas: f.59-62.

1. Cluster (Sistema de computador). 2. Heurística. 3. Otimização combinatória. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 004.35

VICTOR DE OLIVEIRA MATOS

ALGORITMOS HEURÍSTICOS PARA FORMAÇÃO DE CLUSTERS 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: 25 de julho de 2013.

Luciana Brugiolo Gonçalves

Stenio Sã Rosário Furtado Soares

José Elias Claudio Arroyo
(Orientador)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ANOVA	ANalysis Of VAriance
BCDCP	Base station Controlled Dynamic Clustering Protocol
CH	Cluster Head
EEMC	Energy-Efficient Multi-level Clustering
EL	Elegible List
GPS	Global Position System
GRASP	Greedy Randomized Adaptative Seach Procedure
GRASP-C	Greedy Randomized Adaptative Seach Procedure for Clustering
GRASP-H	Greedy Randomized Adaptative Seach Procedure Hierarchical
GRASP+PR	Greedy Randomized Adaptative Seach Procedure with Path Relinking
ID	Identificador
LC	Lista Candidatos
LEACH	Low Energy Adaptative Clustering Hierarchy
LEACH-C	Low Energy Adaptative Clustering Hierarchy Centralied
MIPS	Million Instructions Per Second
MIT	Massassuchets Institute of Technology
PEGASIS	Power Efficient Gathering in Sensor Information Systems
PR	Path Relinking
QoS	Quality of Service
RAM	Random Access Memory
RL	Restriction List
RPD	Relative Percentual Deviation
RSSF	Redes de Sensores Sem Fio

RESUMO

MATOS, Victor de Oliveira, M.Sc., Universidade Federal de Viçosa, Julho de 2013. **Algoritmos Heurísticos para formação de *clusters* em redes de sensores sem fio**. Orientador: José Elias Claudio Arroyo.

Redes de sensores sem fio (RSSF) são um tipo de rede ad-hoc caracterizada por sensores que possuem recursos limitados e são responsáveis por monitorar diversos tipos de ambientes e enviar os dados coletados para uma estação-base. Os sensores geralmente são dispositivos pequenos, baratos e possuem energia limitada. Desta forma, é importante utilizar protocolos de roteamento que gerenciam de maneira eficiente a energia dos sensores. Existem diversas maneiras para transmitir as informações coletadas para a estação-base. As técnicas de roteamento baseadas em *clusters* serão o foco principal deste trabalho. Clusterização consiste em agrupar os sensores, onde alguns agem como líder, conhecido como *cluster head*, que são responsáveis por gerenciar a comunicação do grupo. Como a formação de *clusters* em uma RSSF é um problema NP-Difícil, neste trabalho é proposto o uso da meta-heurística GRASP para obter configurações eficientes de redes de sensores baseados em clusterização. Foram desenvolvidos duas versões do algoritmo GRASP, uma versão para topologia de de um único nível e a outra versão que considera uma topologia multinível. No algoritmo para topologia de nível simples, foi testado também um procedimento de intensificação baseada na técnica *Path Relinking*. Para avaliar o desempenho dos algoritmos, foi desenvolvido um simulador que é executado em ciclos (ou *rounds*). A cada *round* determina-se uma configuração da rede e em seguida é feita a transmissão de dados pela rede. Os resultados obtidos foram comparados com os protocolos da literatura, LEACH, LEACH-C e EEMC.

ABSTRACT

MATOS, Victor de Oliveira, M.Sc., Universidade Federal de Viçosa, July, 2013. **Heuristic algorithms for cluster formation in wireless sensors network**. Orientador: José Elias Claudio Arroyo.

Wireless sensor networks (WSN's) are a type of ad hoc network characterized by a large amount of sensors that have limited resources and are responsible for monitoring the enviroment and sending the collected data to a base-station. The sensors are usually small devices, inexpensive and have finite amount of energy. Thus, it is important to develop energy aware routing protocols. There are several ways to send the collected information to the base station. Routing techniques based on clusters will be the main focus of this work. Clustering consist of grouping the sensors, where some of them act as leader, known as cluster heads, which are responsible for managing the group communication. As the formation of optimal clusters in a WSN is a problem known to be NP-Hard, this paper proposes the use of the meta-heuristic GRASP to obtain efficient topologies in sensor networks based on clustering. We developed two versions of the GRASP algorithm. A version for single level topology and another version that considers multilevel topology. The algorithm for single level topology, was also tested with an intensification procedure based on the Path Relinking technique. To evaluate the performance of the algorithms, we develop a simulator that runs in cycles (or rounds). Each round determines a configuration of the network and the data transmission is made over the network. The results were compared to literature protocols, LEACH LEACH-C and EEMC.

Capítulo 1

Introdução

Redes de sensores sem fio (RSSF) consistem em um sistema formado por nós sensores distribuídos em uma determinada área geográfica, e uma estação base. Os sensores são responsáveis por monitorar o ambiente e repassar os dados coletados para a estação base. As aplicações de uma RSSF variam desde uso em segurança doméstica e monitoramento remoto de pacientes até aplicações militares e de prevenção/detecção de desastres. Devido a algumas de suas aplicações, os sensores podem estar posicionados em ambientes hostis e inacessíveis, como campos de batalha, florestas e oceanos. Nesses casos, a localização dos sensores pode ser programada, podendo ser feita de maneira aleatória, como por exemplo, lançando os sensores de um avião. Dessa forma, é necessário que os sensores formem uma rede ad-hoc sem fio para realizar suas ações (Abbasi e Younis, 2007) (Akyildiz *et al.*, 2002) (Ruiz, 2003). A Figura 1.1 demonstra os passos da formação de uma RSSF.

Os sensores são dispositivos pequenos e baratos, geralmente compostos por um microprocessador, memória, um transmissor *wireless*, um localizador GPS, uma bateria e um ou vários dispositivos de sensoriamento. O microprocessador é responsável pelo processamento central do sensor, utilizado para realizar tarefas básicas como processamento de sinais, correção de erros, etc. O localizador GPS tem a função de repassar para o *sink* ou para sensores vizinhos, a localização exata do sensor. O rádio transmissor *wireless* é responsável pela função de comunicação do sensor com os demais membros da rede. A bateria é responsável por fornecer energia ao sensor, tendo capacidade limitada. Os dispositivos de sensoriamento são responsáveis por monitorar o ambiente a fim de detectar algum alteração, captando os sinais físicos, químicos ou biológicos (Brittes, 2007) (Akyildiz *et al.*, 2002).

Os sensores são responsáveis por realizar o monitoramento de algum fenômeno no ambiente e repassar as informações coletadas para uma estação base (*sink*) localizada

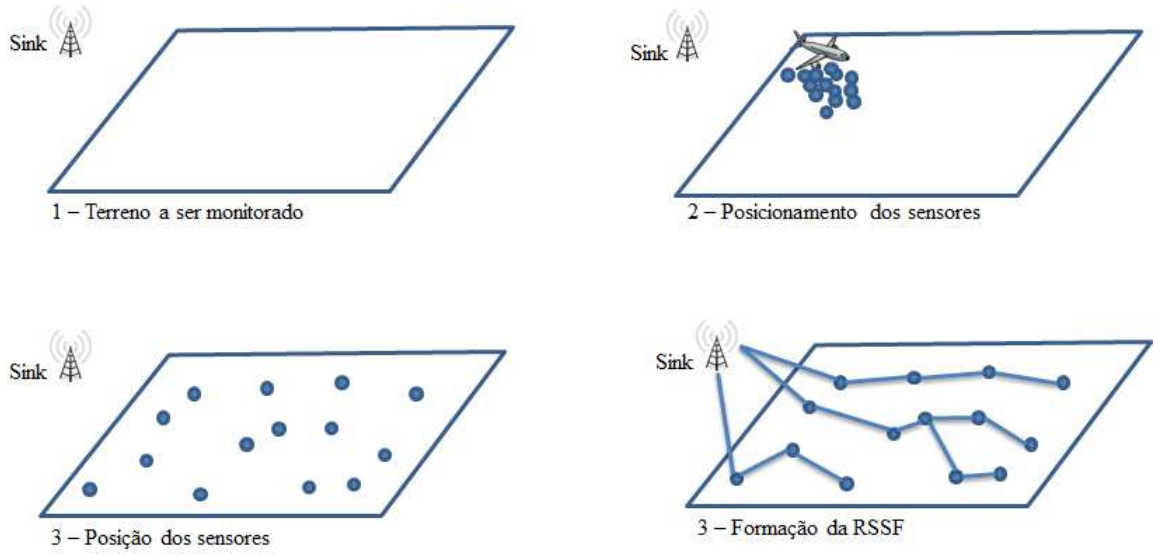


Figura 1.1: Formação de uma RSSF

geralmente em um local afastado. O *sink* se diferencia dos demais sensores por não possuir restrições de hardware e não realizar sensoriamento, agindo como um *gateway* de comunicação da RSSF com o exterior.

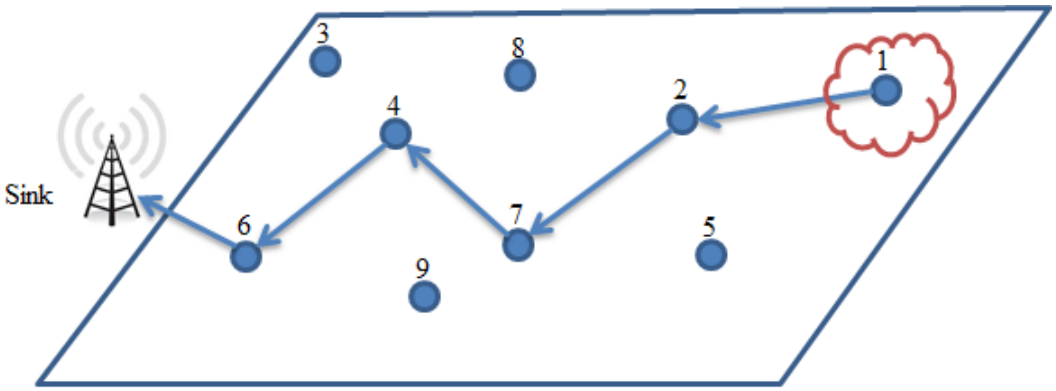
Os sensores se comunicam através de um canal sem fio. A disseminação de dados pela rede pode acontecer em diversas situações. Em uma rede com monitoramento contínuo, os sensores constantemente enviam informações para a estação base, enquanto no monitoramento dirigido a eventos, os sensores apenas enviam informações quando um evento de interesse ocorre. Pode haver ainda sistemas híbridos, onde os sensores alternam entre o monitoramento contínuo e dirigido a eventos. (Brittes, 2007)

Diferente das redes convencionais, esse tipo de rede é composta por uma grande quantidade de nós com recursos limitados, como por exemplo, energia, alcance máximo de transmissão e capacidade de processamento. Devido a estas limitações, os sensores são muito mais suscetíveis a falhas do que os membros de uma rede ad-hoc tradicional. Isso acarreta uma constante mudança de topologia, já que muitos sensores acabam sendo perdidos ou ficando sem energia, exigindo que os mesmos se reorganizem em uma nova topologia.

Quando um sensor fica sem energia, ele é considerado morto e deixa de realizar o monitoramento do ambiente. Devido a natureza das aplicações em RSSF, pode ser impossível fornecer mais energia aos sensores, Assim, enquanto protocolos de redes ad-hoc tradicionais focam em *Quality of Service* (QoS), protocolos em RSSF's focam principalmente no gerenciamento eficiente da energia dos sensores a fim de prolongar o tempo de vida da rede (Akyildiz *et al.*, 2002).

Nas RSSF, devido às restrições de *hardware*, o alcance da transmissão é limitado. Assim, os sensores geralmente trabalham de maneira conjunta, servindo como nós intermediários, repassando mensagens de outros sensores para que estas possam chegar à estação base, formando uma rede ad-hoc *multi-hop* (Ruiz, 2003). A Figura 1.2 exemplifica uma rede onde o sensor 1 detecta um fenômeno qualquer e repassa a informação coletada para a estação base utilizando os sensores de número 2, 7, 4 e 6 como intermediários.

Figura 1.2: Exemplo de uma RSSF *multi-hop*



A comunicação entre os sensores é o que mais consome energia nas RSSF. O consumo de energia para realizar uma transmissão depende da distância e de fatores relacionados ao ambiente, como por exemplo, obstáculos no ambiente. Para transmitir 1Kb de informação a uma distância de 100 metros, um sensor simples consome aproximadamente 3 *joules* de energia. Em contrapartida, um processador modesto que executa 100 milhões de instruções por segundo por *watt* (MIPS/W) é capaz de executar 300 milhões de instruções com a mesma quantidade de energia. Assim, dependendo da aplicação da RSSF, pode ser mais vantajoso refinar as informações coletadas localmente a fim de reduzir a quantidade de dados transmitidos (Pottie e Kaiser, 2000).

Dependendo da aplicação, uma RSSF pode ser classificada de diversas maneiras. Segundo (Brittes, 2007) e (Ruiz, 2003) as classificações são feitas de acordo com a configuração (Tabela 1.1), o tipo de coleta de dados (Tabela 1.2), o tipo de comunicação (Tabela 1.3) e o tipo de processamento (Tabela 1.4). Os requerimentos de uma RSSF variam dependendo da sua aplicação. Assim, pode-se adotar estratégias diferentes baseada na classificação da rede. Por exemplo, em uma aplicação que realiza o monitoramento apenas durante a noite, os sensores podem ser desligados durante o dia a fim de economizar energia.

Tabela 1.1: Classificações de uma RSSF segundo a configuração

Composição	Homogênea	Os sensores possuem a capacidade de hardware iguais. O software pode ser diferente.
	Heterogênea	Os sensores possuem capacidade de hardware distintas.
Organização	Hierárquica	Os sensores formam grupos com um sensor exercendo o papel de líder.
	Plana	Os sensores não estão agrupados.
Mobilidade	Estacionária	Os sensores não se movimentam pela região a ser monitorada.
	Móvel	Os sensores se movimentam. Não existe uma posição fixa para cada sensor no terreno.
Densidade	Balanceada	Distribuição e concentração dos nós por unidade de área considerada ideal, conforme o objetivo e aplicação da rede
	Densa	Alta concentração de nós por unidade de área.
	Esparsa	Baixa concentração de nós por unidade de área.

Tabela 1.2: Classificações de uma RSSF segundo o tipo de coleta de dados

Coleta	Periódica	Os sensores realizam a coleta dos dados em intervalos pré-definidos, como por exemplo, durante a noite ou durante o dia.
	Continua	Os sensores coletam os dados de forma ininterrupta.
	Reativa	Os sensores coletam os dados apenas quando ocorrem eventos de interesse ou quando solicitado pelo observador.

Energia é um ponto crítico em RSSF, já que seu uso de maneira ineficiente pode ocasionar a morte prematura de sensores, deixando áreas sem monitoramento ou até mesmo deixando a rede totalmente desconexa. Assim, os protocolos e algoritmos utilizados pelos sensores devem levar em conta o consumo de energia. A informação da quantidade de energia restante (também chamada de energia residual) de uma região da rede é chamada de mapa de energia (Chan e Han, 2009). O mapa de energia de uma RSSF pode ser útil para detectar sensores ou regiões que estão sobrecarregados e estão consumindo mais energia. Esse tipo de informação é muito útil para prolongar o tempo de vida da rede.

Tabela 1.3: Classificações de uma RSSF segundo a comunicação

Comunicação		
Disseminação	Periódica	Os sensores enviam os dados em intervalos agendados.
	Continua	Os sensores disseminam os dados continuamente.
	Sob Demanda	Os sensores disseminam os dados somente quando requisitado.
	Sob Eventos	Os sensores disseminam os dados somente quando ocorre um evento de interesse.
Tipo de Conexão	Simétrica	Todas as conexões existentes entre os sensores possuem o mesmo alcance.
	Assimétrica	As conexões entre os sensores têm alcance diferente.
Transmissão	Simplex	Os sensores apenas realizam a transmissão de dados.
	Half-Duplex	Os sensores podem transmitir e receber dados em instantes diferentes.
	Full-Duplex	Os sensores podem transmitir e receber dados ao mesmo tempo.
Alocação do Canal	Estática	A fim de minimizar a interferência, cada sensor recebe uma fatia diferente da frequência (FDMA - *Frequency division miltiple Access*), de código (CDMA - *Code division multiple Access*), do espaço (SDMA - *Space Division Multiple Access*) ou ortogonal (OFDM - *Orthogonal frequency division multiplexing*) para realizar a transmissão diminuindo assim a colisão de dados.
	Dinâmica	Neste tipo de rede não existe atribuição fixa de largura de banda. Os nós disputam o canal para comunicação dos dados.
Fluxo de Informação	*Flooding*	Os sensores fazem *broadcast* de suas informações para os vizinhos, os quais fazem *broadcast* para outros até alcançar o ponto sorvedouro. Esse tipo de fluxo provoca um alto *overhead* mas fica imune à mudanças dinâmicas na rede e à ataques de impedimento de serviço (DoS - *Denial of service*)
	Multicast	Os nós formam grupos e usam o *multicast* para se comunicar entre os membros do grupo.
	Unicast	Os nós sensores podem se comunicar diretamente com o ponto de acesso usando protocolos de roteamento *multi-hop*.
	Gossiping	Os nós sensores selecionam os nós para os quais enviam dados.
	Bargaining	Os nós enviam dados somente se o nó destino manifestar interesse, havendo um processo de negociação.

Tabela 1.4: Classificações de uma RSSF segundo o processamento

Processamento		
Cooperação	Infra-estrutura	Os nós executam procedimentos relacionados à infra-estrutura da rede como, algoritmos de controle de acesso ao meio, roteamento, eleição de lideres, descoberta de localização e criptografia.
	Localizada	Os nós executam além dos procedimentos de infra-estrutura, algum tipo de processamento local básico como tradução de dados coletados.
	Correlação	Os nós realizam procedimentos de correlação de dados como fusão, supressão seletiva, contagem, compressão, multiresolução e agregação.

A redução do consumo de energia dos sensores vem sendo alvo de muitas pesquisas no desenvolvimento de protocolos para RSSF. Basicamente, os protocolos podem ser divididos em quatro grandes classes: protocolos de roteamento, protocolos de clusterização e protocolos de agendamento de ativação e desativação (*sleep-and-awake*) (Yi *et al.*, 2007).

Nos protocolos de roteamento, é formada uma rota *multi-hop* até a estação base, possibilitando que os sensores realizem transmissões mais curtas. Nos protocolos de ativação e desativação, um sensor pode ser desativado durante um certo período de tempo e depois é reativado, diminuindo consideravelmente o consumo de energia. Esse tipo de protocolo é aplicável apenas em situações onde existem sensores redundantes, ou seja, dois ou mais sensores monitorando a mesma área. Caso contrário, a região deixaria de ser monitorada quando o sensor fosse desativado. Nos protocolos de clusterização, os sensores formam grupos, onde um sensor age como líder, sendo responsável por gerenciar toda a comunicação do grupo para a estação base.

Este trabalho foca especificamente nos protocolos da classe de clusterização. Assim, maiores informações sobre os protocolos de roteamento e de ativação/desativação, podem ser obtidas em (Singh *et al.*, 2010), (Padmanabhan e Kamalakkannan, 2011) e (Kumar e Chauhan, 2011).

O agrupamento de sensores em *clusters* (clusterização) tem sido amplamente utilizado em RSSF como um método eficiente para gerenciar a comunicação dos sensores com o *sink*. Nesse tipo de abordagem, a rede é subdividida em grupos, onde cada grupo (*cluster*) possui um sensor atuando como líder do grupo, chamado de *cluster head* (*CH*). Além de proporcionar escalabilidade à rede, os *clusters* possibilitam que os

sensores realizem transmissões mais curtas, o que tornam seu uso vantajoso no ambiente das RSSF.

O roteamento de dados entre os sensores passa a ser feito de maneira local, dentro de cada *cluster*, reduzindo assim o tamanho da tabela de roteamento em cada sensor (Akkaya e Younis, 2005). Além disso, o roteamento *intra-cluster* reduz o consumo da largura de banda da rede, evitando o envio de informações redundantes (Younis *et al.*, 2003). Outra vantagem é a maior estabilidade da topologia da rede dentro de cada grupo, já que os sensores apenas se comunicarão com o CH de seu *cluster*, contribuindo também para a redução do *overhead* de gerenciamento de comunicação (Hou *et al.*, 2005). O CH também pode exercer funções de gerenciamento otimizado, como executar algoritmos simples de agendamento de ativação/desativação para reduzir o consumo de energia dos sensores (Younis *et al.*, 2003) e ainda utilizar técnicas de agregação de dados a fim de diminuir a quantidade de informação que será enviada para o *sink* (Dasgupta *et al.*, 2003) (Rajagopalan e Varshney, 2006).

De acordo com (Abbasi e Younis, 2007), os algoritmos de clusterização podem ser subdivididos em três grandes grupos, dependendo da estrutura da rede, o modelo operacional e o propósito da clusterização. Esses grupos são: propriedades dos *clusters*, capacidade dos *cluster heads* e quanto ao processo de clusterização. As Tabelas 1.5, 1.6 e 1.7 fornecem um maior detalhamento de cada grupo respectivamente.

1.1 Motivação e Objetivos do Trabalho

Ao longo dos últimos anos, as RSSF têm sido alvo de muitos estudos tanto na parte de *hardware* quanto na de *software*. As diferentes aplicações desse tipo de rede geralmente requerem protocolos específicos de comunicação de forma proporcionar o uso eficiente dos recursos do sensores.

Os problemas de otimização para as RSSF geralmente são versões de problemas clássicos da área de pesquisa operacional, como por exemplo, roteamento e clusterização, que são problemas NP-Difícil. Assim, é natural a utilização de meta-heurísticas para resolver esses problemas. O objetivo principal deste trabalho é aplicar a meta-heurística GRASP para formar *clusters* na RSSF. O GRASP foi escolhido por ser uma meta-heurística que se mostrou eficiente na resolução de problemas similares, como o

Tabela 1.5: Taxonomia baseada nas propriedades do *cluster*

Quantidade de *clusters*	Fixa	A quantidade de *clusters* é fixa durante todo o tempo de vida da rede
	Variável	A quantidade de *clusters* pode variar
Topologia *intra-cluster*	Fixa	Os sensores se comunicam diretamente com o *cluster head*
	Adaptativa	A comunicação com o *cluster head* pode ser feita utilizando sensores intermediários. Utilizada quando as restrições de alcance dos sensores impedem a conexão direta.
Conectividade *inter-cluster head*	Conexão direta	Os *cluster heads* se comunicam diretamente com o *sink*.
	Multi-hop	O *cluster head* não se comunica diretamente com o *Sink*, sendo necessário o uso de outros sensores como intermediários.
Estabilidade	Adaptativo	Um sensor se torna membro de diferentes *clusters* durante o tempo de vida da rede.
	Assumida	Um sensor não muda de *cluster* durante o tempo de vida da rede.

das P-medianas e por ser um método ainda não utilizado para resolver o problema abordado neste trabalho. Como objetivos secundários, podemos destacar:

- Testar a eficiência do GRASP para formar os *clusters*;
- Adicionar uma fase de pós-otimização baseada no *Path Relinking* ao GRASP
- Desenvolver uma versão do algoritmo para a topologia multi-nível;
- Desenvolver um simulador para executar a coleta e transmissão dos dados pela rede;
- Analisar o desempenho dos algoritmos propostos em relação a protocolos existentes na literatura;
- Analisar o desempenho dos algoritmos com relação as diferentes topologias;

Tabela 1.6: Taxonomia baseada nas capacidades do *cluster head*

Mobilidade	Estacionário	O *cluster head* se mantém fixo, o que facilita o gerenciamento da comunicação do *cluster*.
	Realocável	O *cluster head* se movimenta apenas em distância limitada, posicionando-se de maneira mais adequada a fim de melhorar a conectividade do *cluster*.
	Móvel	O *cluster head* se movimenta sem limitações pela rede, fazendo com que os *clusters* mudem com frequência.
Tipo dos sensores	Recursos abundantes	Os *cluster heads* são sensores diferenciados, possuindo consideravelmente mais capacidade computacional, recursos de comunicação e energia.
	Sensor comum	Os *cluster heads* são sensores comuns, possuindo limitações de hardware e energia.
Função	*Relay*	Age apenas como sensor intermediário entre o *cluster* e o *sink*.
	Agregação de Dados	Os *cluster heads* executam algoritmos de agregação de dados, para eliminar informações redundantes e reduzir a quantidade de dados enviados.
	Sink	Os *cluster heads* atuam como *sink*, tomando decisões quando um fenômeno é detectado.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte maneira: O Capítulo 2 contém uma revisão de trabalhos relacionados existentes na literatura. Nesse capítulo é feita uma análise dos protocolos da clusterização utilizados para comparação com os algoritmos propostos.

No Capítulo 3 são apresentados os parâmetros do sistema assim como uma definição formal do problema e a representação da solução. Nesta parte são detalhados os passos de cada algoritmo proposto.

O Capítulo 4 contém a análise dos parâmetros de cada algoritmo assim como um detalhamento da parte de simulação das transmissões.

Os resultados obtidos neste trabalho estão presentes no Capítulo 5 juntamente com uma comparação entre os algoritmos implementados.

O último Capítulo apresenta uma revisão geral do que foi desenvolvido neste

Tabela 1.7: Taxonomia baseada no processo de clusterização

Metodologia	Distribuída	Quando a própria rede troca mensagens, elege os *cluster heads* e formam os clusters.
	Centralizada	Quando o *sink* forma os *clusters* e repassa a topologia escolhida para os sensores.
	Híbrida	Quando os *cluster heads* sao escolhidos de maneira centralizada e os sensores membros se juntam aos *clusters* de maneira autônoma.
Objetivo	Tolerância a falha	Os *clusters* são formados de modo a minimizar a perda de mensagens pela rede, garantindo uma maior quantidade de mensagens entregue ao destino.
	Balanceamento de carga	Os *clusters* são formados visando distribuir de maneira igual a carga da rede, evitando que apenas alguns sensores sejam super utilizados enquanto outros fiquem sub utilizados.
	Conectividade	Busca garantir a existência de um caminho entre todos os sensores da rede e o *sink*.
	Outros	Outras finalidades, como maximizar o tempo de vida da rede, formar a menor quantidade de *clusters* possiveis, entre outros.
Seleção dos *cluster heads*	pré-determinada	É executado algum algoritmo para a determinação dos *cluster heads*.
	Aleatória	Os *cluster heads* sao escolhidos de maneira aleatória dentre os sensores da rede.
Complexidade	Constante	A complexidade do algoritmo não se altera dependendo do tamanho da rede e da quantidade de *cluster heads*.
	Variável	A complexidade aumenta ou diminui dependendo da quantidade de sensores na rede ou da quantidade de *clusters* que serão formados.

trabalho e as conclusões obtidas a partir dos testes executados.

Capítulo 2

Revisão da Literatura

Diversos protocolos de clusterização foram propostos na literatura. Este capítulo tem como objetivo oferecer uma explicação sobre o funcionamento de protocolos de clusterização presentes na literatura. Os protocolos LEACH, LEACH-C e EEMC são detalhados de maneira mais profunda, pois são utilizados na comparação com os algoritmos propostos.

2.1 Low Energy Adaptative Clustering Hierarchy

Pelo algoritmo LEACH(*Low Energy Adaptative Clustering Hierarchy*) (Heinzelman *et al.*, 2000), cada sensor possui uma probabilidade de se tornar *cluster head* e faz um *broadcast* com a sua decisão para os demais nós. Os sensores que não forem atuar como *cluster head* se juntam ao *cluster* cujo *cluster head* pode ser atingido com o menor custo de energia, geralmente sendo o mais próximo.

A função de *cluster head* é muito mais custosa do que a função de um sensor comum, já que além de realizar o monitoramento, o *cluster head* também é responsável por executar algoritmos de processamento de dados simples e também serve de intermediário entre todos os sensores do *cluster* e o *sink*. Assim, a energia dos sensores que são eleitos *cluster heads* tende a se esgotar muito mais rapidamente, o que põe em risco a funcionalidade da rede, já que caso um *cluster head* fique sem energia, todo o *cluster* perde a capacidade de comunicação com o *sink*.

Para amenizar esse problema, o LEACH utiliza o conceito de *rounds*. Um *round* pode ser definido como um período de tempo que a rede permanece com uma determinada topologia de modo que, ao início de cada *round*, uma nova topologia com novos *cluster heads* é formada. Um *round* é composto por duas fases distintas: a fase de *setup*, seguida pela fase de transmissão (*steady-state*). A primeira fase é responsável

pela eleição dos *clusters heads* e formação dos *clusters*, ou seja, a topologia da rede é definida. A fase de transmissão é onde a simulação da rede realmente é executada, ou seja, os sensores realizam o monitoramento, coletam os dados, repassam-nos para seu *cluster head* que por fim os envia para o *sink*. No LEACH, o *round* tem a duração de 20 segundos. Basicamente, a cada simulação do LEACH, diversos *rounds* são executados, até que a rede seja considerada inativa. A Figura 2.1 apresenta as etapas do LEACH durante uma simulação.

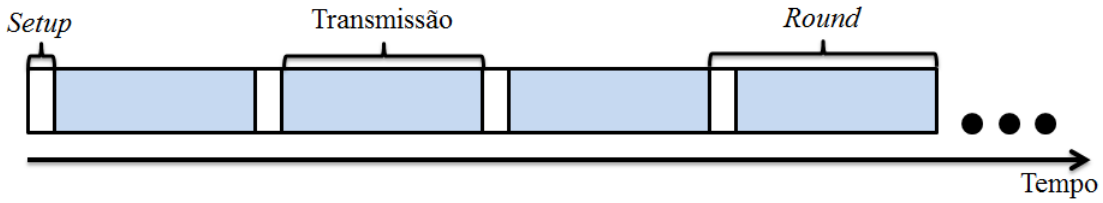


Figura 2.1: Gráfico da simulação do protocolo LEACH.

A cada fase de *setup*, o LEACH obriga os sensores a elegerem *cluster heads* diferentes, evitando que os mesmos sensores sejam eleitos repetidas vezes. Esse tipo de rotatividade visa buscar uma distribuição da carga entre todos os sensores de maneira uniforme. O número esperado de sensores que não foram eleitos *cluster heads* nos primeiros r *rounds* pode ser estimado como $n-(p\times r)$, onde n é a quantidade de sensores da rede, e p é o número esperado de *cluster heads* eleitos em cada *rounds*. Assim, espera-se que em $\frac{n}{p}$ *rounds*, todos os sensores da rede já tenham exercido a função de *cluster head*. Para formar os *clusters*, cada sensor sorteia um número aleatório T entre 0 e 1, e o sensor i será *cluster head* se:

$$T(i) = \begin{cases} \frac{p}{n-p\times(r mod \frac{n}{p})}, & \text{se} \quad i \in G, \\ 0, & \text{Caso contrário.} \end{cases} \tag{2.1}$$

Onde n é a quantidade de sensores da rede, p é a quantidade de *clusters* que deseja-se formar, r é o *round* atual e G é o conjunto de sensores que não foram eleitos *cluster heads* nos últimos $\frac{n}{p}$ *rounds*. Com isso, passados $\frac{n}{p}$ *rounds*, espera-se que todos os sensores da rede possuam aproximadamente a mesma quantidade de energia. Para tal, o LEACH assume que a quantidade inicial de energia dos sensores seja igual e todos os sensores transmitem quantidades iguais de dados durante a fase de transmissão. Assim, em redes onde o envio de informações seja orientado a eventos, a função de probabilidade deve levar em conta a energia residual dos sensores, priorizando a escolha de sensores que possuam maior quantidade de energia para serem eleitos como *cluster head*. Dessa forma, a função de probabilidade é ajustada de acordo com a energia

residual do sensor em relação a energia residual de toda a rede, sendo dada pela Equação 2.2:

$$P(i) = MIN(\frac{E_i(t)}{E_{total}(t)} \times k, 1) \tag{2.2}$$

Onde $E_i(t)$ é a energia residual do sensor e $E_{total}(t)$ é o somatório da energia residual de todos os sensores da rede. Porém, para utilizar a equação 2.2, cada sensor deve possuir uma estimativa da energia de toda a rede.

2.2 Low Energy Adaptative Clustering Hierarchy Centralized

O LEACH-C (*Low Energy Adaptative Clustering Hierarchy Centralized*) (Heinzelman *et al.*, 2002) foi desenvolvido como uma extensão do LEACH para resolver os problemas de seu antecessor. Formar os *clusters* de maneira distribuída é vantajoso por não necessitar de intervensão externa, porém o LEACH não garante uma quantidade fixa de *clusters* nem uma distribuição de forma igual dos *cluster heads* pela rede. O LEACH-C contorna esse problema executando um algoritmo centralizado no *sink* para eleger os *cluster heads*.

Enquanto a fase de transmissão do LEACH-C é idêntica a do LEACH, na fase de *setup*, todos os sensores da rede enviam informações para o *sink* como sua energia residual e posição (em caso de redes móveis). O *sink* então executa a meta-heurística *simulated annealing* (Ishibuchi e Murata, 2004) para eleger os *cluster heads* de maneira eficiente. O algoritmo leva em conta a quantidade de energia de cada sensor, formando uma lista de sensores elegíveis para serem *cluster head*. Um sensor i só é considerado elegível se:

$$E(i) > \frac{\sum_{j=0}^{n} E(j)}{n} \tag{2.3}$$

Onde E é a função que retorna a quantidade de energia de um dado sensor e n é a quantidade de sensores da rede. Isso garante que apenas os sensores que possuírem mais energia serão escolhidos para atuarem como *cluster head*.

O *simulated annealing* é uma meta-heurística iterativa que parte de uma solução inicial e realiza perturbações nesta solução a fim de encontrar soluções melhores, aceitando movimentos de piora da solução com uma dada probabilidade que diminui ao longo do tempo. O Algoritmo 1 descreve passo a passo a meta-heurística do LEACH-C. Inicialmente, é montada uma lista (EL) dos sensores candidatos a serem *cluster heads*.

Uma outra lista *CH* é formada contendo p sensores escolhidos de forma aleatória de *EL* (Figura 2.2-2).

Como podemos ver nas linhas 5 e 6, para cada sensor de *CH* é feita uma perturbação em suas coordenadas, somando algum valor aleatório, obtendo uma nova coordenada no terreno (Figura 2.2-3). Como essa nova coordenada gerada pode ser um ponto onde não há nenhum sensor, a função *MAP_CH* faz o mapeamento da nova coordenada para o sensor mais próximo a ela, elegendo-o como novo *cluster head* (Figura 2.2-4). Se a nova solução obtida for melhor do que a anterior, então é feita a substituição das soluções. Porém, se a nova solução não for melhor, ainda existe uma probabilidade *pr* de ser feita a substituição das soluções, como pode ser visto na linha 12. Essa probabilidade é maior nas primeiras iterações, diminuindo a cada iteração executada do algoritmo. O critério de parada do algoritmo é por número de iterações executadas, retornando a lista *CH* ao final.

Algoritmo 1 LEACH-C

```
1:  EL ← monta lista de sensores elegíveis;
2:  CH ← p sensores aleatórios de EL;
3:  while i < iters do
4:     for each c ∈ CH do
5:        Coord[c] ← Coord[c] + rand();
6:        NEW_CH ← MAP_CH(CH);
7:     end for
8:     if f(NEW_CH) < f(CH) then
9:        CH ← NEW_CH;
10:    else
11:       pr ← prob();
12:       if (rand() < pr) then
13:          CH ← NEW_CH
14:       end if
15:    end if
16: end while
17: return CH;
```

Os resultados obtidos pelo LEACH-C em **?** mostraram ser significativamente melhores que o LEACH, indicando que a adoção de meta-heurísticas para eleger os *cluster heads* de maneira mais inteligente pode ser eficiente para prolongar o tempo de vida da rede.

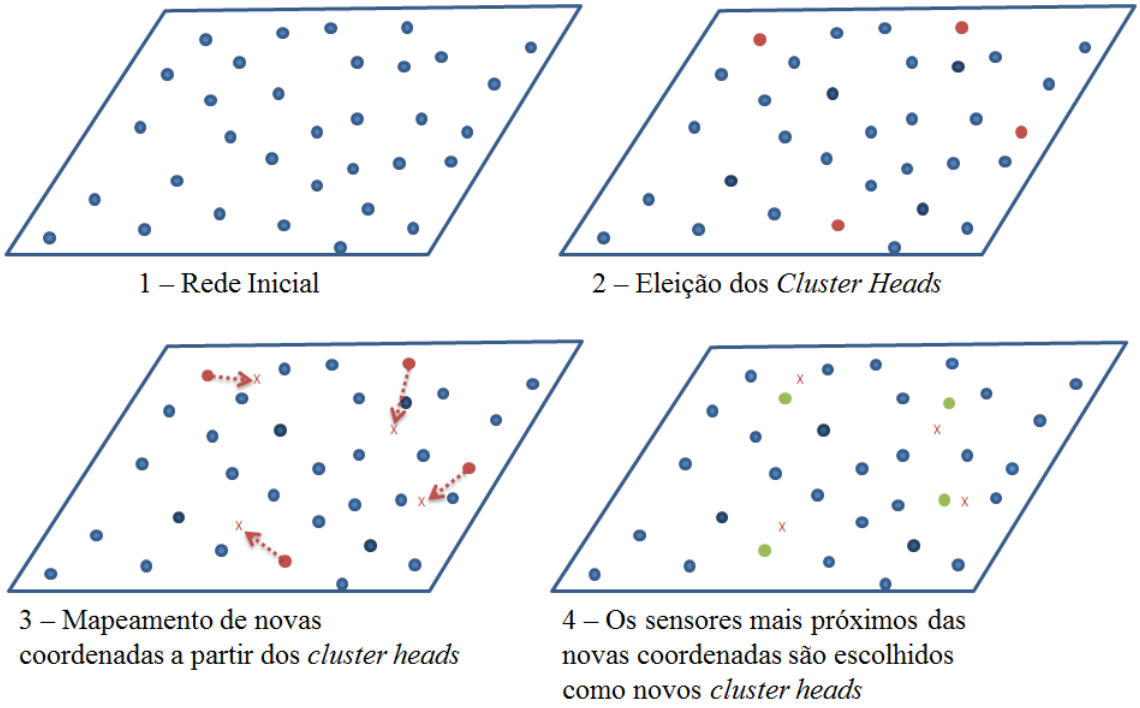


Figura 2.2: Obtenção de uma nova solução pelo *simulated aneeling* do LEACH-C

2.3 Energy Efficient Multilevel Clustering

Enquanto o LEACH e o LEACH-C formam apenas *clusters* em um único nível, o EEMC (*Energy Efficient Multilevel Clustering*), proposto em (Jin *et al.*, 2008), forma vários níveis de *clusters*. O EEMC, os *clusters* são formados de maneira distribuída, com os sensores fazendo *broadcast* de mensagens para se organizarem e montarem a topologia da rede. No esquema de topologia multi-nível, cada *cluster* pode ser subdividido em novos *clusters*. A Figura 2.3 exemplifica uma rede formada com três níveis de clusterização.

Na topologia multi-nível, os *cluster heads* de nível superior (chamados de *cluster heads* nível-1) são os responsáveis pela comunicação com a estação base. Os *cluster heads* de nível superior recebem mensagens dos de nível menor. Assim, aqueles sensores que não forem eleitos *cluster heads* em nenhum nível, enviarão os dados coletados para os *cluster heads* de nível inferior. A quantidade de saltos necessárias para um sensor membro atingir a estação base será $Q_{nivel} + 1$, onde Q_{nivel} é a quantidade de níveis de clusterização que o algoritmo forma.

A fase de *setup* do EEMC funciona em um esquema *top-down*, ou seja, os *cluster heads* de nível mais alto são eleito antes dos de nível mais baixo. Assim, inicialmente, todo sensor ativo da rede possui uma probabilidade $p1$ de se tornar *cluster head* do

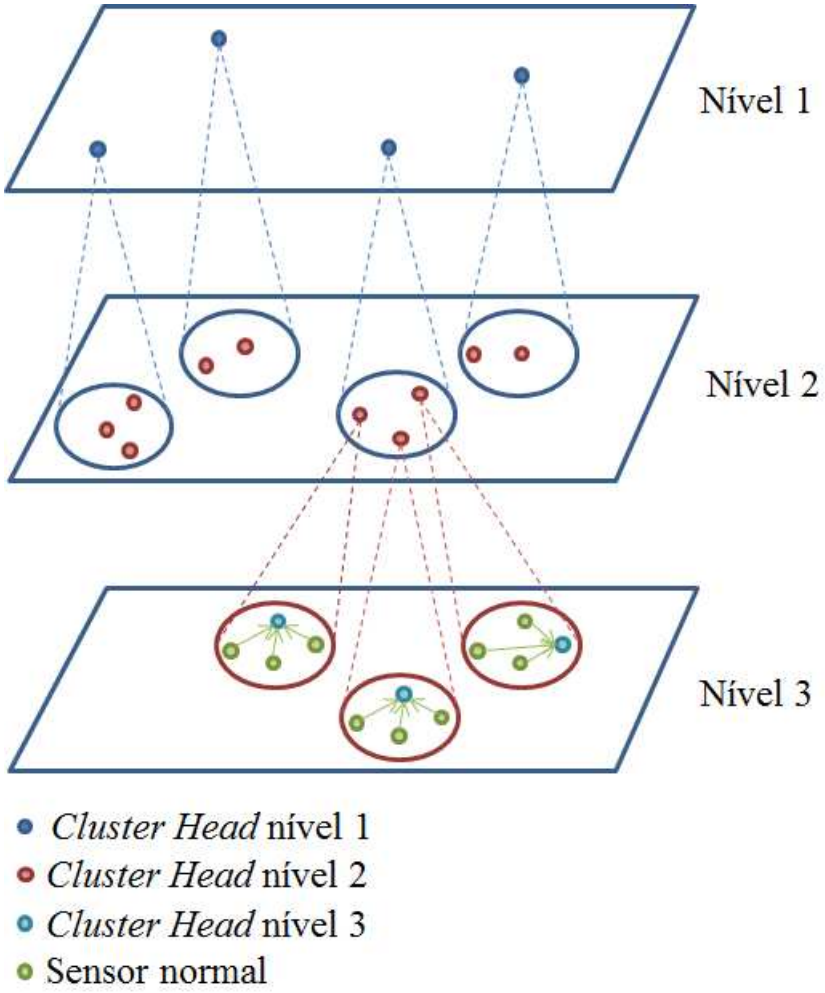


Figura 2.3: Exemplo de uma rede multinível com 3 níveis de clusterização.(adaptada de (S. Banerjee, 2001))

nível-1. Aqueles que forem eleitos para este nível, fazem *broadcast* de uma mensagem em um determinado raio. Todos os sensores que receberem a mensagem e não forem *cluster head* se juntarão ao *cluster* mais próximo. Após isso, os membros de cada *cluster* terão uma probabilidade $p2$ de se tornarem *cluster head* do nível-2. Os sensores eleitos farão broadcast de uma mensagem, e aqueles que receberem e não forem *cluster head* se juntarão ao *cluster* nível-2 mais próximo. Os passos anteriores podem ser generalizados para formar i níveis de *clusters*. Os sensores que não receberem nenhuma mensagem durante a fase de *setup* são forçados a serem *cluster heads*. No EEMC, cada *cluster* é forçado a ter pelo menos dois sensores membros. O Algoritmo 2 contém o pseudo-código do EEMC.

As probabilidades dos sensores serem eleitos *cluster heads* são calculadas baseadas na energia residual do sensor em comparação com a energia total da rede e do inverso

Algoritmo 2 EEMC(S, level, CH)

if $level = 1$ **then**
  $S = G$
end if
for cada $c \in S$ **do**
  p_c = probabilidade do sensor c ser *cluster head*
end for
for cada $c \in S$ **do**
  if $rand() < p_c$ **then**
    $CH = CH \cup c$
  end if
end for
for cada novo $ch \in CH$ **do**
  S' = todos os sensores c' que estejam a uma distância R de ch e $c' \notin CH$
  if $S'.size() >= 2$ **then**
    $EEMC(S', level + 1, CH)$
  end if
end for

da distância de cada sensor à estação base em comparação ao somatório do inverso da distância de todos os sensores à estação base. No caso dos *clusters* de menor nível, para o cálculo das probabilidades, o *cluster head* de nível superior passa a ser considerado como estação base. Assim, as probabilidades do EEMC são:

$$p_1 = N_{ch} \times (\phi \times \frac{E(i)}{\sum E(j)} + (1 - \phi) \times \frac{\frac{1}{D(j,sink)}}{\sum \frac{1}{D(i,sink)}}) \tag{2.4}$$

$$p_n = \sqrt{N_{ch}} \times (\phi \times \frac{E(i)}{\sum E(j)} + (1 - \phi) \times \frac{\frac{1}{D(i,CH_{n+1})}}{\sum \frac{1}{D(j,CH_{n+1})}}) \tag{2.5}$$

Onde N_{ch} é a quantidade esperada de *cluster heads*, ϕ é um parâmetro ajustável de prioridade na eleição dos *cluster heads*, E é a função que retorna a quantidade de energia de um dado sensor e D é a função que retorna a distância de um dado sensor a um destino pré-determinado.

Na fase de transmissão do EEMC, cada *cluster head* de qualquer nível é responsável por executar agregação de dados. Os sensores de nível mais alto recebem as mensagens dos sensores de nível mais baixo e as repassam para os *cluster heads* do nível superior, até que o nível-1 seja atingido, onde as mensagens são enviadas ao *sink*.

2.4 Outros Trabalhos Relacionados

Diversos outros trabalhos foram desenvolvidos para formar *clusters* em RSSF. O PEGASIS (*Power Efficient Gathering in Sensor Information Systems*) (Lindsey e Raghavendra, 2002) é um protocolo que forma uma cadeia de comunicação utilizando um algoritmo guloso. Cada sensor se comunica apenas com o seu vizinho mais próximo, agrega a mensagem recebida com a sua própria e a envia para o próximo sensor na cadeia. A cada *round*, um sensor é escolhido de forma aleatória para enviar os dados coletados para o *sink*. O PEGASIS garante que cada sensor receberá e enviará apenas uma mensagem. Além disso todos os sensores serão líderes (se comunicarão com o *sink*) pelo menos uma vez.

O BCDCP (*Base Station Controlled Dynamic Clustering Protocol*) (Chatterjee e Singh, 2012) é outro protocolo de formação de *clusters* centralizado na estação base, que busca dividir a rede em grupos de tamanhos iguais, em uma tentativa de balancear o consumo de energia da rede. O CLUBS (Nagpal e Coore, 1998), é um protocolo distribuído que permite um sensor pertencer a vários *clusters*. Cada *cluster* tem um raio fixo, assim todos os sensores que estiverem dentro desse raio de distância do *cluster head* são considerados como sendo alcançados com um salto. Neste algoritmo, cada sensor sorteia um número inteiro aleatório e faz uma contagem regressiva do valor sorteado até zero. Se ele não receber nenhuma mensagem de algum vizinho nesse período (a contagem chegar a zero), o sensor envia uma mensagem de recrutamento à uma distância de dois saltos e os sensores que receberem a mensagem, interrompem a sua contagem e se juntam ao *cluster*. Ao se juntar a um *cluster*, um sensor não para de escutar a rede, podendo receber outras mensagens de recrutamento e se juntar a outros *clusters*.

Diversos métodos utilizando meta-heurísticas e métodos de inteligência artificial foram propostos para resolver o problema de clusterização em RSSF's. (S. Jin e Wu, 2003) é um dos trabalhos pioneiros utilizando algoritmos genéticos (AG) para resolver o problema de clusterização em RSSF's. Como função de *fitness* é utilizado a distância entre cada *cluster head* a estação base somada com a distância de cada sensor ao seu respectivo *cluster head* juntamente com a quantidade de *cluster heads*. Foram obtidos bons resultados, servindo como base para outros trabalhos utilizando AG's.

Em Salehpour *et al.* (2008) é proposto um algoritmo memético. Neste trabalho, um algoritmo genético seguido de uma busca local é executado a fim de obter a melhor configuração de *clusters* para a rede. Além dos parâmetros adotados em S. Jin e Wu (2003), a função de *fitness* também leva em consideração a quantidade de energia residual da rede. Mehrjoo *et al.* (2011) propõem um algoritmo híbrido de AG com

colônia de abelhas artificial (CAA). A função de *fitness* adotada para avaliar as fontes de alimentos (solução) além da soma das distâncias como em (S. Jin e Wu, 2003), utiliza também a densidade do sensor (quantidade de sensores dentro de um determinado raio).

Em (Tillet *et al.*, 2002) é proposto um algoritmo baseado em otimização por enxame de partículas. Neste trabalho, a rede é subdividida gradativamente até que se obtenha a quantidade desejada de grupos. Os resultados obtidos são avaliados em comparação ao algoritmo *k-means*. Outro algoritmo que utiliza enxame de partículas foi proposto em Latiff *et al.* (2007). Neste trabalho, assim como o LEACH-C, a estação base calcula a média de energia da rede a fim de que sejam escolhidos apenas os sensores com mais energia que a média de energia da rede para serem *cluster heads*.

A lógica *fuzzy* também vem sendo empregada para formar *clusters*. Em Godbole (2012) os *cluster heads* são escolhidos baseados na quantidade de energia residual e distância para a estação base. A lógica *fuzzy* é empregada para avaliar os parâmetros para a escolha dos *cluster heads*. Outro trabalho que utiliza lógica *fuzzy* foi proposto em Gupta *et al.* (2005). As regras utilizadas foram a centralidade do sensor em relação aos demais, a energia de cada sensor e a concentração de sensores da vizinhança.

Capítulo 3

Modelo do Sistema e Algoritmo Proposto

Neste capítulo é discutido em detalhes o algoritmo proposto para resolver o problema de formação de *clusters* em RSSF. Além disso é feita uma descrição sobre o modelo do sistema adotado na rede e características da mesma assumidas para as simulações.

3.1 Modelo do Sistema

Neste trabalho foi adotado o modelo de consumo de energia dos sensores proposto em (Heinzelman *et al.*, 2000) e (Heinzelman *et al.*, 2002). Neste modelo, os sensores dissipam energia para realizar transmissões, receber dados e executar o algoritmo de agregação de dados (executado por todos os *cluster heads*). Para transmitir dados, os sensores consomem energia ao utilizar o rádio transmissor e o amplificador, enquanto que para receber mensagens, apenas o rádio transmissor é utilizado. Além disso, o rádio transmissor é capaz de controlar a potência de transmissão, utilizando somente o necessário para atingir o sensor destino. A Figura 3.1 ilustra o modelo de rádio adotado.

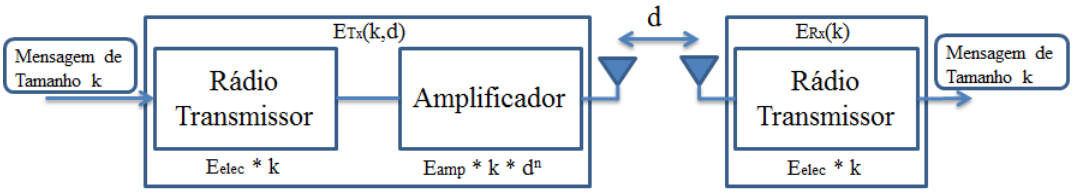


Figura 3.1: Modelo de rádio adotado. (Figura baseada em (Heinzelman *et al.*, 2000))

Usar o rádio transmissor consome $E_{elec} = 50nJ/bit$. Para utilizar o amplificador,

consome-se $E_{amp} = 100pJ/bit/m^2$. Assim, para transmitir uma mensagem de tamanho k a uma distância d ($E_{tx}(k, d)$) e para receber uma mensagem de tamanho k ($E_{rx}(k)$) consome-se respectivamente:

$$E_{tx}(k, d) = E_{elec} * k + \epsilon_{amp} * k * d^n \tag{3.1}$$

$$E_{rx}(k) = E_{elec} * k \tag{3.2}$$

onde n é a constante de dissipação do meio, que pode variar de 2 até 4, dependendo das características do ambiente onde a rede está implantada. Em ambientes planos, sem obstáculos, essa contante vale 2, enquanto em ambientes muito acidentados ou com vários obstáculos, essa constante vale 4. Neste trabalho, assumimos que a constante de dissipação n é sempre igual a 2.

O algoritmo de agregação de dados executado pelos *cluster heads* consome $E_{da} = 5nJ/bit$ de energia. Em um *cluster* com h membros, cada membro transmite uma mensagem de tamanho k para o *cluster head*. Assim, assumimos que, independentemente do valor de h, o algoritmo de agregação sempre conseguirá comprimir todas as mensagens do grupo em uma única mensagem de tamanho k.

O modelo de rede e de sensores adotado é similar ao proposto em (Heinzelman *et al.*, 2000). As características principais são descritas a seguir:

- Cada sensor monitora constantemente o ambiente e sempre possui dados para enviar ao *sink*.
- O *sink* está localizado em um ponto afastado do terreno e possui energia ilimitada.
- Todos os sensores são estacionários, homogêneos e iniciam com a mesma quantidade de energia.
- Os sensores são capazes de controlar a potência de transmissão de modo que será consumido apenas a quantidade necessária de energia para atingir o destino.
- Todos os sensores são capazes de alternar entre a função de *cluster head* e a função de sensor membro.
- Não há limite de distância para realizar as transmissões. Assim, todo sensor é capaz de alcançar todos os outros sensores da rede com apenas um salto.
- A quantidade de *clusters* é fixada de acordo com a quantidade de sensores ativos na rede.

Utilizando o modelo de energia apresentado, pode-se obter o número ótimo de *cluster* da rede. O estudo feito em (Heinzelman *et al.*, 2002) indica que o número ótimo de *clusters* em uma rede gira em torno de 5% da quantidade de sensores da rede. Tomando como base esse estudo, a quantidade de *clusters* adotada neste trabalho é fixada em 5% do total de sensores ativos.

O problema de formação de *clusters* em RSSF's pode ser definido formalmente como: Dado um conjunto G de sensores posicionados de maneira aleatória em um terreno e uma estação base (*sink*) S localizado em um ponto afastado. Para uma topologia com apenas um nível, o problema consiste em encontrar um conjunto de n_{ch} sensores para exercerem a função de *cluster head* de forma a minimizar a distância ao quadrado de cada sensor ao seu *cluster head* e de cada *cluster head* ao *sink* seja a menor possível.

Para topologias com mais de um nível, o problema consiste em encontrar um conjunto de $n_{ch}i$ sensores para exercerem a função de *cluster head* nível i, onde i é o nível do *cluster*, de forma a minimizar distância ao quadrado de cada sensor ao seu respectivo *cluster head* ou ao *sink*.

3.2 Representação da Solução

Para representar a solução do problema, foi utilizado um par de vetores: *CH* e *I*. O primeiro vetor é de tamanho n_{ch} e contém a ID de todos os sensores que forem eleitos como *cluster head*. O segundo vetor possui tamanho igual a n onde cada índice representa um sensor e o valor de cada posição representa o *cluster head* associado ao sensor. Nessa abordagem, o *sink* é representado como sendo o sensor S. A Figura 3.2 representa uma solução com $n_{ch} = 3$ e $n = 16$ na topologia com um único nível.

O mesma representação é utilizada para a topologia multinível. Nesse caso, todos os *cluster heads* que forem eleitos em algum nível, farão parte do vetor *CH*. A Figura 3.3 exemplifica uma rede juntamente com a solução para a respectiva topologia.

Como a função objetivo leva em conta apenas a distância de cada sensor ao destino de sua transmissão, apenas o vetor I é utilizado para avaliar uma solução. O vetor *CH* serve como base para a construção do vetor I, já que o valor de cada posição de I poderá conter apenas os elementos de *CH* e o *sink*.

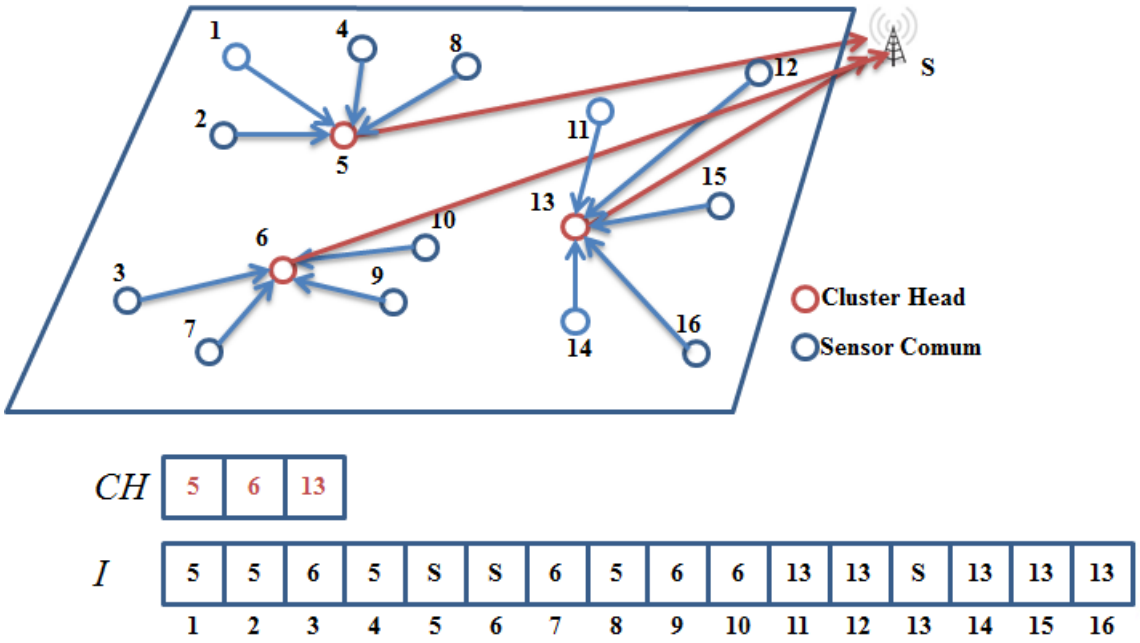


Figura 3.2: Representação da solução na topologia de um único nível

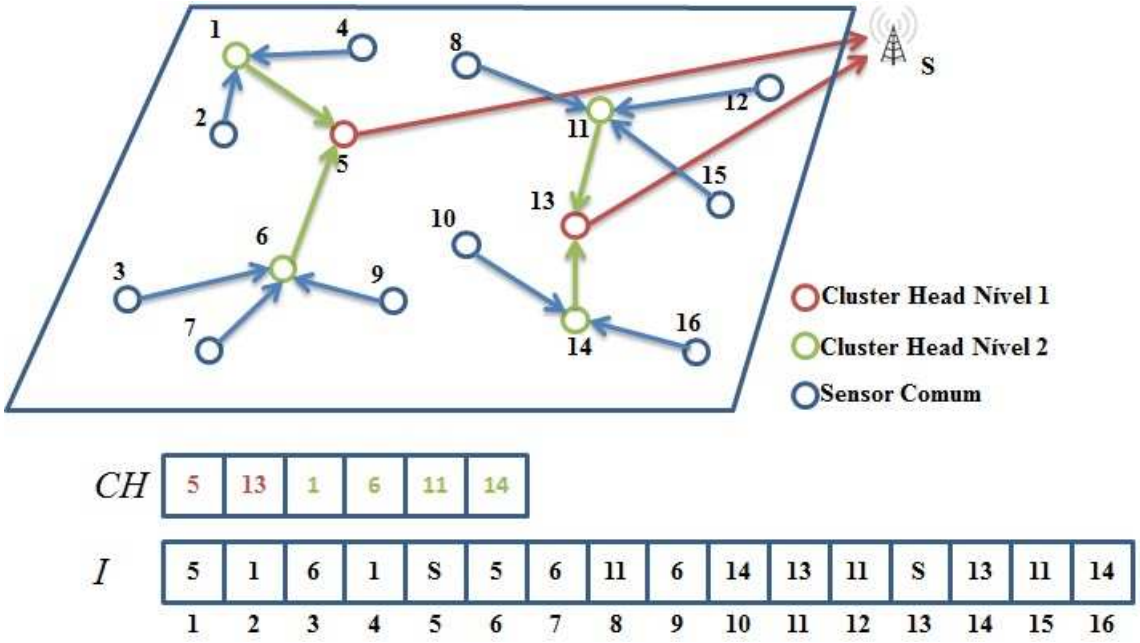


Figura 3.3: Representação da solução na topologia multinível

3.3 Algoritmos Propostos

Neste trabalho são propostos três algoritmo baseado na meta-heurística GRASP para resolver o problema de formação de um número fixo de *clusters* em RSSF. Este problema pode ser considerado como sendo uma versão do problema das p-medianas onde

os pontos (sensores) podem ser escolhidos como sendo medianas (*cluter heads*) e serão responsáveis por atender as demandas dos clientes (sensores membros). Os problemas de clusterização são problemas de otimização combinatória conhecidamente NP-Difíceis (Agarwal e Procopiuc, 1999). Assim, o desenvolvimento de uma algoritmo que resolva o problema de forma exata em tempo computacional viável é improvável.

Além das abordagens propostas para o problema, foi desenvolvido um simulador para executar a fase de transmissão de dados. Na simulação, inicialmente os os algoritmos de clusterização são executados para formar uma topologia de rede. Com a topologia estabelecida, as transmissões são feitas e a energia gasta é subtraída da energia dos sensores. A simulação é executada até o critério de parada ser atingido (a rede ser considerada morta). Essa fase é importante para avaliar o desempenho dos algoritmos ao longo de todo o tempo de vida da rede. Essa fase é dividida em *rounds*, como proposto pelo LEACH e uma nova configuração da rede é formada a cada *round* (o algoritmo de clusterização é executado novamente).

O GRASP (*Greedy Randomized Adaptative Search Procedure*) é uma meta-heurística proposta em (Feo e Resende, 1995) bastante utilizada para resolver problemas de otimização combinatória para obter soluções de boa qualidade em tempo computacional aceitável. O GRASP é um método iterativo onde a cada iteração é obtida uma nova solução, e é constituído de uma fase de construção seguida por uma fase de busca local. O algoritmo executa essas duas fases em diversas iterações até um critério de parada ser atingido, que pode ser um número máximo de iterações, tempo de processamento, entre outros.

A fase de construção é responsável por gerar uma solução inicial válida para o problema utilizando um algoritmo guloso parcialmente aleatório. Nessa fase é importante ressaltar a presença da aleatoriedade na geração das soluções iniciais, garantindo que a cada iteração do GRASP, o algoritmo terá uma solução inicial potencialmente diferente. A fase de busca local é responsável por tentar melhorar a solução inicial obtida na fase anterior a fim de obter um ótimo local. Ao final do algoritmo, a melhor solução encontrada é retornada como sendo a nova topologia válida da rede.

Neste trabalho foram desenvolvidas três versões diferentes do algoritmo GRASP. O GRASP-C (*GRASP Clustering*) é um algoritmo composto apenas pelas fases básicas do GRASP. Uma outra versão denominada GRASP+PR foi desenvolvida como uma extensão do GRASP-C, incorporando também uma uma fase de intensificação baseada no *Path-Relinking*. A terceira versão (chamada de GRASP-H) foi desenvolvida a fim de formar uma hierarquia de *clusters*, caracterizando uma topologia de rede multi-nível.

3.3.1 Greedy Randomized Adaptative Search Procedure Centralized

O primeiro algoritmo, chamado de *Greedy Randomized Adaptative Search Procedure Centralized* (GRASP-C) segue a fórmula básica do GRASP: uma fase de construção seguida por uma fase de busca local. Essas duas fases são repetidas até um critério de parada ser atingido. O Algoritmo 3 representa o pseudocódigo do GRASP-C. O algoritmo GRASP-C possui dois parâmetros: max_it e α. O primeiro se refere ao critério de parada do algoritmo, que foi definido como sendo um número máximo de iterações sem a atualização da melhor solução. Já o α é o parâmetro que controla o tamanho da vizinhança a ser procurada na busca local. No GRASP-C são permitidos apenas movimentos de melhora. Ou seja, a atualização da solução ocorre somente se a solução obtida pela busca local (CH) for melhor do que a melhor solução encontrada até o momento (CH_b), como pode ser visto na linha 7. Ao final do algoritmo, a melhor solução (CH_b) é retornada.

Algoritmo 3 GRASP-C (max_it, α)

```
1:  f(CH_b) ← ∞;
2:  it ← 0;
3:  LC ← elegible(G) (Conjunto de sensores candidatos a serem cluster head)
4:  while it < max_it do
5:     CH ← sampleGreedy(LC);
6:     CH ← BuscaLocal(CH, α, LC);
7:     if f(CH) < f(CH_b) then
8:        CH_b ← CH;
9:        it ← 0;
10:    else
11:       it ← it + 1
12:    end if
13: end while
14: return CH_b
```

O vetor LC é responsável por armazenar todos os sensores da rede que poderão ser candidatos a atuarem como *cluster head* no *round* atual. Para um sensor ser candidato a *cluster head*, ele deverá possuir uma quantidade mínima de energia de modo que ele consiga exercer essa função durante o *round*. Isso permite que sensores bem localizados sejam escolhidos mais vezes, enquanto ele possuir energia suficiente. Considerando *clusters* de mesmo tamanho, podemos estimar a quantidade de energia que um sensor deverá possuir para conseguir exercer a função de *cluster head* em um *round*.

Seja RL a quantidade de dados pré-definido para a duração de um *round*, e k o

tamanho da mensagem. Assim, $(RL/k)/n_{ch}$ pode ser definida como sendo a quantidade esperada de mensagens que serão transmitidas por cada *cluster*. Em outras palavras, pode-se estimar a quantidade de mensagens que um *cluster head* terá que receber e realizar a agregação de dados. Assim, a Equação 3.3 indica a fórmula da energia mínima para um sensor ser candidato a *cluster head*.

$$E_{min} = (RL/k)/n_{ch} \times E_{rx}(k) \times E_{DA} + E_{tx}(k, d_{sink}) \tag{3.3}$$

A seguir é descrito em detalhes cada uma das fases do GRASP-C.

3.3.1.1 Fase de Construção

A construção de uma solução inicial para o problema é feita utilizando o algoritmo *Sample Greedy*, proposto por Resende e Werneck (2003) para o problema das P-Medianas. O algoritmo recebe como parâmetro uma lista LC de elementos que serão considerados para a construção da solução. O *Sample Greedy* inicia com uma solução vazia ($CH = \emptyset$) e, a cada iteração, um novo elemento é adicionado a CH.

Em cada iteração, ao invés de ser escolhido o melhor elemento dentre todos os pertencentes a LC, o algoritmo considera apenas um conjunto Q de elementos escolhidos aleatoriamente de LC. É escolhido então o melhor elemento de Q para ser adicionado à solução CH. O tamanho de Q é definido como o proposto em Resende e Werneck (2003): $q = \lceil log_2(\frac{n}{n_{ch}}) \rceil$, onde n é a quantidade de sensores na rede e n_{ch} é a quantidade fixa de *cluster heads* que deverão ser escolhidos.

A ideia é fazer q pequeno o suficiente para reduzir significativamente o custo computacional do algoritmo (quando comparado a um algoritmo puramente guloso, onde todas as possíveis inserções serão testadas) e para garantir um nível razoável de aleatoriedade, de modo que a cada iteração uma nova solução potencialmente diferente possa ser gerada. A Figura 3.4 exemplica um passo da construção de uma nova solução pelo algoritmo *sample greedy*.

O algoritmo é executado até que a solução CH esteja completa, ou seja, contenha a quantidade n_{ch} de *cluster heads*. n_{ch} é um valor conhecido *a priori* que leva em conta a quantidade de sensores da rede. O pseudocódigo descrito no Algoritmo 4 descreve os passos do algoritmo construtivo *Sample Greedy*.

3.3.1.2 Busca Local

A busca local é um método de melhoria baseado na busca em vizinhança. Uma vizinhança pode ser definida como um conjunto de soluções que podem ser obtidas a

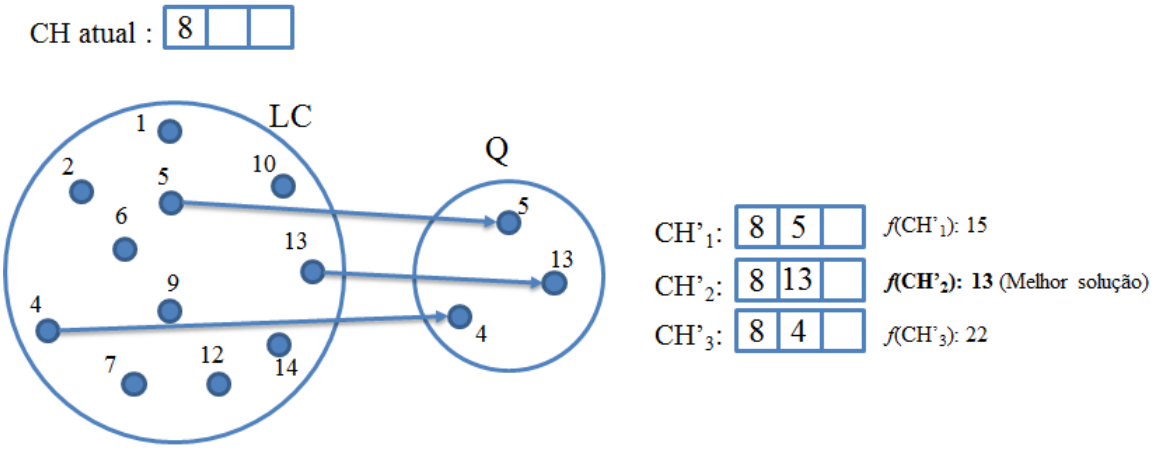


Figura 3.4: Exemplo da construção da solução inicial (algoritmo *Sample Greedy*)

Algoritmo 4 *sampleGreedy*(LC)

1: $CH \leftarrow \emptyset$
2: **while** $|CH| < n_{ch}$ **do**
3: $\quad Q \leftarrow$ conjunto de q sensores escolhidos aleatoriamente de LC
4: $\quad ch \leftarrow$ escolhe o melhor elemento de Q para ser *cluster head*
5: $\quad CH \leftarrow CH \cup \{ch\}$
6: **end while**
7: **return** CH

partir de uma solução inicial realizando um determinado movimento. Neste trabalho, o movimento escolhido foi a troca de sensores. O algoritmo recebe uma solução inicial CH gerada pelo método construtivo e efetua trocas de *cluster heads*. A troca de *cluster head* determina uma solução vizinha removendo um elemento ch de CH e inserindo um outro elemento $c \notin CH$. A Figura 3.5 exemplifica um movimento empregado na busca local. No exemplo dado, são escolhidos dois elementos de cada conjunto (CH e $LC - CH$) e é feita a troca desses elementos entre os grupos, gerando uma outra solução, denominada CH'. Esse tipo de movimento vêm sido comumente utilizado em procedimentos de busca local para o problema das P-Medianas e têm provado ser eficiente (Resende e Werneck, 2003).

Testar todas as possíveis trocas pode ser computacionalmente custoso, assim, para reduzir a quantidade de soluções vizinhas, optou-se por utilizar um subconjunto C de sensores que serão candidatos (sensores que poderão ser trocados com ch). Esse conjunto é formado por uma porcentagem α do total de sensores de $LC - CH$.

Cada sensor ch é trocado com todos os sensores $c \in C$. Dentre todas as trocas realizadas, a solução CH^* que obtiver o melhor valor da função objetivo é escolhida. Se CH^* for melhor que a solução inicial CH, CH é atualizado com CH^*.

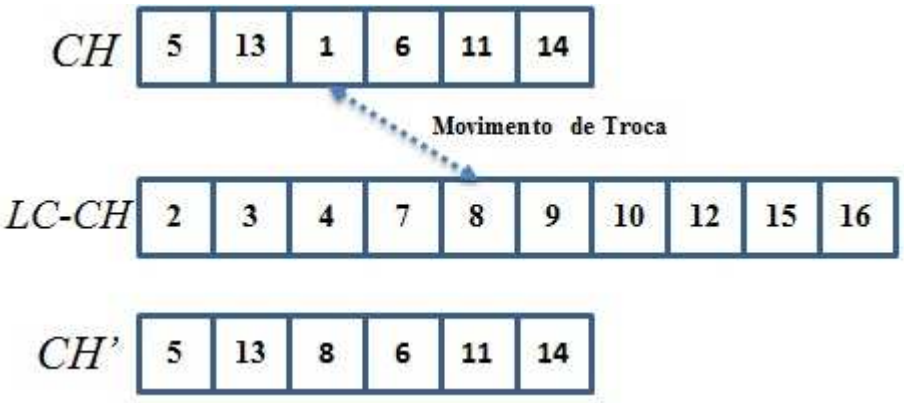


Figura 3.5: Exemplo de um movimento de troca feita pela busca local

O algoritmo testa a remoção de todos os sensores $ch \in CH$. Essa remoção é feita de maneira gulosa removendo primeiramente o pior sensor, ou seja, é removido o sensor que contribuir menos para o valor da função objetivo.

Caso ao remover um sensor *ch* nenhuma solução melhor for obtida, esse sensor é colocado em uma lista de proibição (adicionado a uma lista R). Quando uma solução melhor for obtida, o conjunto R é esvaziado, possibilitando novamente a remoção de sensores anteriormente proibidos em iterações futuras. O algoritmo termina quando $|R| = n_{ch}$, ou seja, quando todos os *cluster heads* forem proibidos de ser removidos. O Algoritmo 5 detalha o procedimento de busca local.

3.3.2 Greedy Randomized Adaptative Search Procedure + Path Relinking

Nesta versão do algoritmo, a fase de construção e de busca local são as mesmas do algoritmo GRASP-C. No algoritmo *Greedy Randomized Adaptative Search Procedure + Path Relinking* (GRASP+PR) é utilizada uma fase de intensificação, baseada no *Path-Relinking* (Glover, 1996), a fim de melhorar ainda mais os resultados obtidos pela busca local. O procedimento de *Path-Relinking* é detalhado na sub-seção a seguir.

3.3.2.1 Path-Relinking

A técnica de *Path-Relinking* é uma mecanismo com o propósito de combinar intensificação e diversificação explorando a trajetória que conecta duas soluções de boas qualidades (soluções elite) produzidas anteriormente durante a fase de busca local. O *Path-Relinking* necessita de um par de soluções distintas, digamos CH_o (solução origem) e CH_g (solução guia).

Algoritmo 5 *Busca Local*(CH, α, LC)

1: $R \leftarrow \emptyset$; { lista de sensores proibidos de serem removidos}
2: **while** $|R| < n_{ch}$ **do**
3: $ch \leftarrow$ escolhe o pior sensor $ch \in CH$ tal que $ch \notin R$;
4: $C \leftarrow$ escolha aleatoriamente $\alpha \times |LC - CH|$ sensores diferentes de $(LC - CH)$;
5: $CH^* \leftarrow CH$;
6: **for** cada sensor $c \in C$ **do**
7: $CH' \leftarrow$ solução obtida de CH através da troca de ch e c;
8: **if** $f(CH') < f(CH^*)$ **then**
9: $improved \leftarrow true$;
10: $CH^* \leftarrow CH'$;
11: **end if**
12: **end for**
13: **if** *melhorou* **then**
14: $CH \leftarrow CH^*$;
15: $R \leftarrow \emptyset$;
16: **else**
17: $R \leftarrow R \cup \{ch\}$;
18: **end if**
19: **end while**
20: **return** CH;

Existem várias formas de implementar o *Path Relinking* Resende e Ribeiro (2005), tais como:

- *forward relinking*: o caminho é construído de uma solução CH_o, que possui a pior qualidade, em direção à CH_g;
- *backward relinking*: o caminho é construído de uma solução CH_o, na qual possui a melhor qualidade, em direção à CH_g;
- *bidirecional relinking*: dois caminhos são construídos, sendo o primeiro de CH_o em direção a CH_g e outro de CH_g em direção à CH_o;
- *mixed relinking*: dois caminhos são explorados simultaneamente originado da CH_o e da CH_g, e se encontram em uma solução intermediária;

Neste trabalho, foram implementadas duas versões do *Path Relinking*: *forward Path-Relinking* e *Backward Path-Relinking*. A versão *Forward Path-Relinking* obteve resultados melhores e portanto foi a adotada neste trabalho.

O procedimento de *Path-Relinking* parte da solução origem (CH_o) e gradativamente a transforma na solução guia (CH_g). A transformação é feita inserindo elementos de $CH_g - CH_o$ e removendo elementos de $CH_o - CH_g$. Em outras palavras, o caminho

entre duas soluções é construído por movimentos de troca. A quantidade de passos necessários para transformar CH_o em CH_g é $|CH_g - CH_o|$ (que é o mesmo a $|CH_o - CH_g|$, que é a diferença simétrica de CH_o e CH_g). A cada passo da transformação, a melhor solução obtida é escolhida. Soluções cada vez menos parecidas com a solução origem e cada vez mais parecidas com a solução guia são geradas. O *Path-Relinking* retorna a melhor solução encontrada no caminho entre as duas soluções. A Figura 3.6 exemplifica a construção do caminho entre duas soluções.

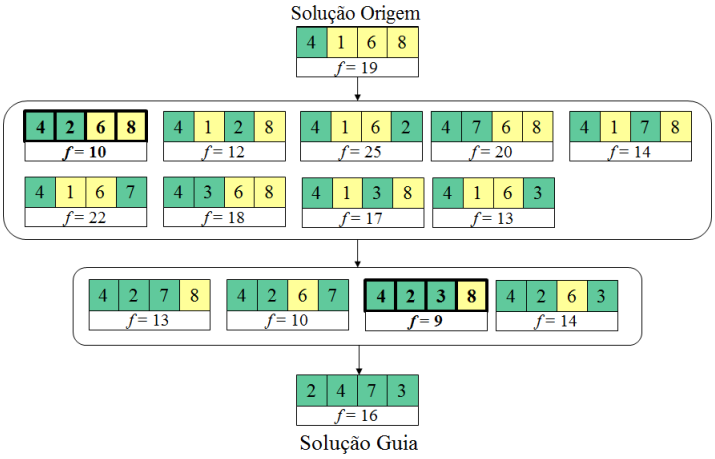


Figura 3.6: Exemplo do *Path Relinking* (caminho entre duas soluções

O Algoritmo 6 detalha cada passo do algoritmo GRASP+PR. Nesta versão do *Path Relinking* é utilizado um *Pool* de soluções elites de tamanho p_{size}. Esse conjunto é atualizado pela função *addSolutionPool(CH)* (linha 8) onde CH é uma solução potencialmente boa obtida anteriormente pela busca local do algoritmo *GRASP+PR*. Esse procedimento verifica se o conjunto está cheio ($|Pool| = p_{size}$), se $CH \notin Pool$ e se CH é diferente o suficiente das soluções do $Pool$. Se essas condições forem satisfeitas, o $Pool$ é atualizado com a inserção da nova solução CH, substituindo uma das soluções elites se $|Pool| = p_{size}$. A solução a ser removida será aquela que for mais diferente entre todas as do $Pool$. Quando o conjunto elite estiver cheio, inicialmente escolhe-se qual solução será candidata a deixar o $Pool$, nesse caso, a solução escolhida não participa do cálculo das diferenças com a nova solução candidata. Em caso de empate, o valor da função objetivo é utilizado como critério para desempate. Em caso de persistência no empate, a escolha é feita de forma aleatória. A diferença entre as soluções é calculada da seguinte forma:

$$Dif(CH) = \sum_{i=1}^{P_{size}} |CH - CH_i| \tag{3.4}$$

Se $|Pool| < p_{size}$ a solução é simplesmente inserida no $Pool$. Esse tipo de abordagem busca garantir a presença de soluções consideravelmente diferentes no $Pool$, permitindo ao algoritmo uma maior exploração do espaço de busca. A Figura 3.7 mostra um exemplo da remoção e inserção de uma solução no $Pool$. No exemplo, a solução menos diferente ($CH6$) vai ser removida do $Pool$ para a inserção da solução candidata, cuja diferença é calculada para todas as soluções do $Pool$.

Algoritmo 6 GRASP+PR (max_it, α)

```
1:  f(CH_b) ← ∞;
2:  Pool ← ∅;
3:  it ← 0;
4:  LC ← conjunto de sensores candidatos a serem cluster head;
5:  while it < max_it do
6:     CH ← sampleGreedy(LC);
7:     CH ← localSearch(CH, α, LC);
8:     addSolutionPool(CH);
9:     for each CH' ∈ Pool do
10:       if CH ≠ CH' then
11:          CH_pr ← forwardPathRelinking(CH, CH');
12:          if f(CH_pr) < f(CH) then
13:             CH ← CH_pr;
14:             addSolutionPool(CH);
15:          end if
16:       end if
17:    end for
18:    if f(CH) < f(CH_b) then
19:       CH_b ← CH;
20:       it ← 0;
21:    else
22:       it ← it + 1
23:    end if
24: end while
25: return CH_b
```

3.3.3 Greedy Randomized Adaptative Search Procedure Hierarchical

O *Greedy Randomized Adaptative Search Procedure Hierarchical* (GRASP-H) é uma versão recursiva do algoritmo GRASP proposto para formar *clusters* multiníveis. Esta versão do algoritmo recebe três parâmetros iniciais: max_it, $level$ e L_C. O primeiro

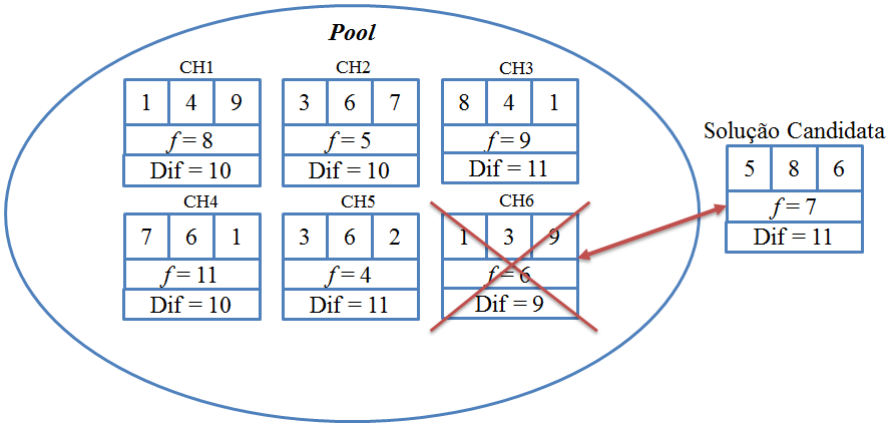


Figura 3.7: Exemplo da remoção e inserção de uma nova solução no *Pool*

parâmetro é o mesmo utilizado nas outras versões do GRASP, ou seja é o critério de parada.

O parâmetro *level* se refere ao nível da clusterização, de modo que a cada subdivisão de um *cluster* ocorre, esse parâmetro é incrementado em uma unidade. O parâmetro L_c contém a lista de sensores que farão parte do processo de clusterização no nível atual. No caso da topologia com um único nível, esse parâmetro não era necessário pois todos os sensores eram considerados. Neste caso, ao fazer a subdivisão de um *cluster*, apenas os sensores deste *cluster* deverão ser considerados. Outro parâmetro interno do algoritmo é chamado de *max_level*, que é responsável por controlar a profundidade da clusterização, ou seja, em quantos níveis a rede será subdividida. O pseudocódigo apresentado no Algoritmo 7 descreve passo a passo o GRASP-H.

A função $elegible(L_c)$ (linha 3) é a função que seleciona os sensores que estarão aptos a serem *cluster heads* no nível atual. Nesta versão, o critério de aptidão utilizado foi o mesmo do LEACH-C, ou seja, um sensor só sera candidato se possuir mais energia que a média de energia de todos os sensores da rede. Neste caso, para o cálculo da média, é considerado apenas os sensores de L_c. Como pode ser visto, a cada iteração do algoritmo, a lista L_c contém todos os sensores membros de cujo *cluster head* é *ch* e em seguida, na linha 10, o algoritmo é chamado recursivamente com uma nova lista de sensores candidatos.

A fase de construção do GRASP-H utiliza o *Sample Greedy*, o mesmo utilizado nas demais versões do algoritmo. A busca local, a fim de reduzir o custo computacional do algoritmo foi simplificada. O Algoritmo 8 apresenta o pseudocódigo da busca local do GRASP-H. A estratégia utilizada nesta versão da busca local foi a mesma utilizada para selecionar uma amostragem no *Sample Greedy*. De todos os candidatos a serem *cluster head*, é escolhido uma amostra de tamanho n_{ls}, que é definido como sendo

Algoritmo 7 GRASP-H $(max\_it, level, L_c)$

1: $f(CH_b) \leftarrow \infty$;
2: $it \leftarrow 0$;
3: $LC \leftarrow elegible(L_c)$ (Conjunto de sensores candidatos a serem *cluster head*)
4: **while** $it < max\_it$ **do**
5: $CH \leftarrow sampleGreedy(EL)$;
6: $CH \leftarrow localSearch(CH, EL)$;
7: **if** $level \leq max_{level}$ **then**
8: **for** each $ch \in CH$ **do**
9: $L_c \leftarrow$ Sensores cujo *cluster head* é ch
10: GRASP-H$(max_it, level + 1, L_c)$
11: **end for**
12: **end if**
13: **if** $level = 1$ **then**
14: **if** $f(CH) < f(CH_b)$ **then**
15: $CH_b \leftarrow CH$;
16: $it \leftarrow 0$;
17: **else**
18: $it \leftarrow it + 1$
19: **end if**
20: **end if**
21: **end while**
22: **return** CH_b

$n_{ls} = \lceil log_2(\frac{|EL|}{|CH|}) \rceil$ baseado em Resende e Werneck (2003). O critério de parada da busca local é dado pelo número de iterações, ou seja, a busca local explorará ls_it vizinhanças da solução atual CH. Ao final do algoritmo, a melhor solução encontrada é retornada.

Algoritmo 8 *localSearch*(CH, EL)

1: **while** $it < ls\_it$ **do**
2: $ch \leftarrow$ selecione um sensor aleatório $\in CH$;
3: $C \leftarrow$ escolha aleatoriamente n_{ls} diferentes sensores de $(EL - CH)$;
4: $CH^* \leftarrow CH$;
5: **for** cada sensor $c \in C$ **do**
6: $CH' \leftarrow$ solução obtida de CH trocando os sensores ch e c;
7: **if** $f(CH') < f(CH^*)$ **then**
8: $improved \leftarrow true$;
9: $CH^* \leftarrow CH'$;
10: **end if**
11: **end for**
12: **if** $improved$ **then**
13: $CH \leftarrow CH^*$;
14: **end if**
15: $it \leftarrow it + 1$
16: **end while**
17: **return** CH;

Capítulo 4

Benchmark de instâncias do problema e análise dos parâmetros dos algoritmos

Como não existe um conjunto de instâncias disponíveis na literatura específica para o problema abordado neste trabalho, foi gerado aleatoriamente um conjunto de instâncias para avaliar os algoritmos propostos, em comparação com os outros protocolos da literatura. Inicialmente, foi feita uma comparação apenas dos algoritmos LEACH, LEACH-C, GRASP-C e GRASP+PR. Em seguida para analisamos se há vantagens no uso de topologias multi-níveis em comparação a topologias com apenas um nível, o algoritmo GRASP-H foi comparado com os protocolos LEACH e LEACH-C e com o GRASP-C. Também foi testado o protocolo EEMC para topologias multi-níveis. Devido ao diferente tipo de topologia adotada, optou-se por comparar apenas o GRASP-C com o GRASP-H com o intuito de demostrar a eficiência de uma topologia em relação a outra.

Instâncias contendo $n = \{50, 60, 70, 80, 90, 100, 125, 150, 200, 300\}$ sensores foram geradas, em um terreno quadrado de tamanho 100×100 metros. A estação base para todas as instâncias é posicionada no ponto $x = 150, y = 150$, longe da área de monitoramento. Para cada quantidade de sensores, foram geradas 10 diferentes instâncias, resultando em um total de 100 instâncias. As instâncias foram ordenadas de acordo com a quantidade de sensores e numeradas de 1 a 100 conforme pode ser visto na Tabela 4.1. Cada instância é composta por um arquivo de texto que contém inicialmente a quantidade de sensores n, as coordenadas x e y de cada sensor e por último a posição da estação base.

Tabela 4.1: Relação entre o número da instância e a quantidade de sensores.

Número da instância	Quantidade de sensores
1-10	50 sensores
11-20	60 sensores
21-30	70 sensores
31-40	80 sensores
41-50	90 sensores
51-60	100 sensores
61-70	125 sensores
71-80	150 sensores
81-90	200 sensores
91-100	300 sensores

4.1 Análise dos parâmetros

Todos os algoritmos possuem parâmetros básicos que influenciam em seu comportamento, na qualidade das soluções obtidas e no tempo computacional do algoritmo. Dessa forma, esses parâmetros foram submetidos a um processo de calibragem, a fim de verificar o impacto de cada parâmetro no desempenho do algoritmo e escolher o valor mais adequado. Alguns parâmetros foram definidos conforme estudos anteriores presentes na literatura.

Cada configuração de rede obtida pelos algoritmos possui um valor para a função objetivo, que é o valor da distância total. Dessa forma, cada configuração é comparada com outra utilizando a métrica de desvio relativo percentual, RPD (*Relative Percentual Deviation*), que pode ser calculado pela seguinte formula:

$$RPD\% = 100 \times \frac{f_{algorithm} - f_{best}}{f_{best}} \tag{4.1}$$

onde $f_{algorithm}$ é o valor da função objetivo (distância total) obtida por um dado algoritmo e f_{best} é o valor da função objetivo (distância total) da melhor solução (aquela com a menor distância total entre os sensores e os *cluster heads)* obtida dentre todos os algoritmos comparados. Para o cálculo de f_{best} e $f_{algorithm}$ foram consideradas a

média dos valores da distância total para os cinco primeiros *rounds*. Esse critério foi adotado para proporcionar uma avaliação mais justa, analisando também a influência do critério de elegibilidade de cada algoritmo em uma sequencia de *rounds*.

A fim de validar os resultados, foi feito uma análise de variância (ANOVA) (Montegomery, 2007) para verificar se as diferenças observadas são estatisticamente diferentes. Os testes foram feitos utilizando o valor obtido pelo RPD como variável de resposta. Também foi feito um teste de Tukey para múltiplas comparações a fim de verificar para quais valores os resultados são estatisticamente diferentes. Para cada teste, foram selecionadas uma instância aleatória de cada grupo e para cada instância ela foi executada cinco vezes para cada algoritmo com cada valor de parâmetro a ser avaliado.

As subseções a seguir descrevem a metodologia empregada para avaliação dos parâmetros α, P_{size} e max_it do GRASP-C e GRASP+PR e dos parâmetros ls_it e max_it do GRASP-H.

4.1.1 Análise do parâmetro α da busca local

Inicialmente, analisa-se o parâmetro que controla o tamanho da vizinhança testada na busca local dos algoritmos GRASP-C e GRASP+PR. Para tal teste, foram considerado para $\alpha \in \{0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0\}$. Para a definição deste parâmetro, max_it foi deifinido como 40 iterações sem melhora.

$\alpha = 0,1$ indica que apenas 10% das possibilidades de trocas serão testadas, enquanto que $\alpha = 1,0$ indica um algoritmo exaustivo, ou seja, será feita uma busca em toda a vizinhança e a melhor solução encontrada será retornada. Neste teste foram gerados, então, dez versões do GRASP-C e do GRASP+PR, uma para cada valor de α. O gráfico apresentado na Figura 4.1 representa o gráfico com 95% de intervalo de confiança para a média.

Como pode-se perceber, para $\alpha = 0.1$ e $\alpha = 0.5$, os resultados obtidos são estatisticamente diferentes, já que não há interseções entre os dois intervalos. Porém, para os valores de α variando de 0.2...1.0 os intervalos contém interseções, indicando que não há diferença significativa entre os resultados obtidos. A melhor média foi obtida com $\alpha = 0.7$. Porém, a fim de reduzir o custo computacional, o valor escolhido foi 0.5. Nos algoritmos de busca local, percorrer muitas soluções vizinhas é a fase mais custosa do algoritmo.

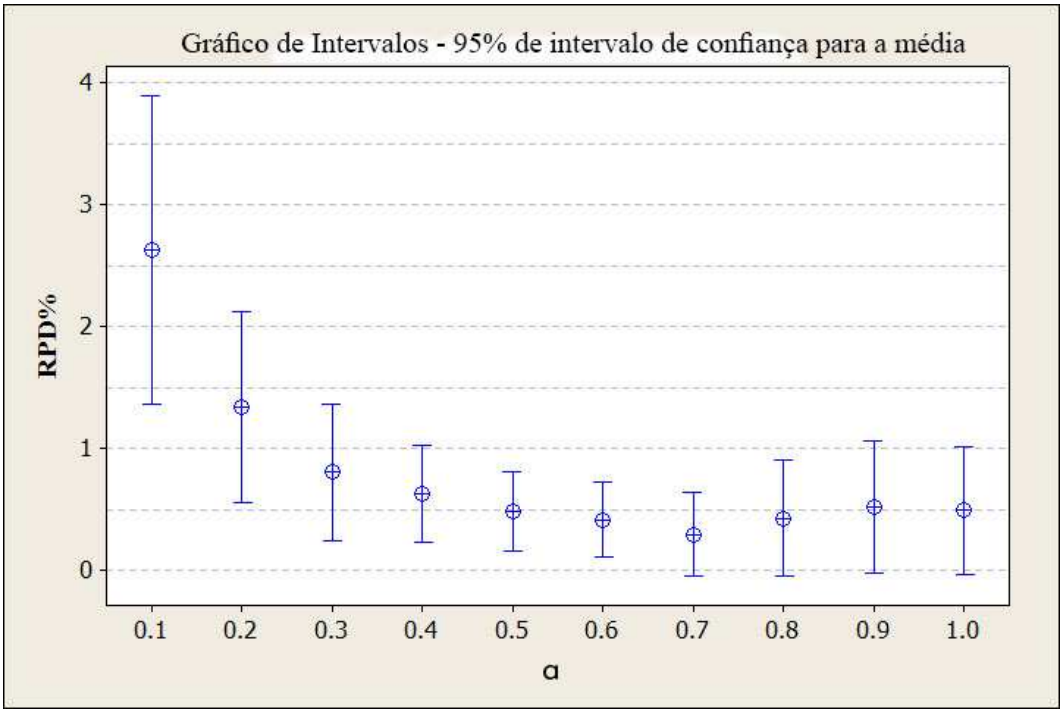


Figura 4.1: Gráfico de intervalos para o parâmetro α da busca local.

4.1.2 Análise dos parâmetros max_it do GRASP-C e P_{size} do GRASP+PR

Com o parâmetro α já definido, é necessário definir os parâmetros P_{size} (tamanho do *Pool* do GRASP+PR) e max_it (condição de parada do GRASP-C e GRASP+PR). Para esses parâmetros, foram testadas as seguintes combinações: $P_{size} \in \{5, 10, 15, 20\}$ soluções e $max\_it \in 5, 10, 20, 40$ iterações sem melhora. No total, foram testadas 16 combinações diferentes. O gráfico apresentado na Figura 4.2 ilustra os intervalos dos resultados obtidos com um intervalo de confiança de 95%.

O resultado do ANOVA indica que há resultados estatisticamente diferentes. Pode-se ver no gráfico que os melhores resultados são obtidos quando o valor de max_it vale 40. As duas melhores médias são obtidas com $P_{size} = 10$ e $max\_it = 40$ e $P_{size} = 20$ e $max\_it = 40$. No entanto, essas duas combinações não são estatisticamente diferentes quando escolhemos $P_{size} = 10$ e $max\_it = 20$ (os intervalos se interceptam), que foram os valores escolhidos para os parâmetros. Um *Pool* de soluções muito grande é muito custoso, pois o *Path Relinking* gera e avalia diversas soluções para formar o caminho de uma solução a outra.

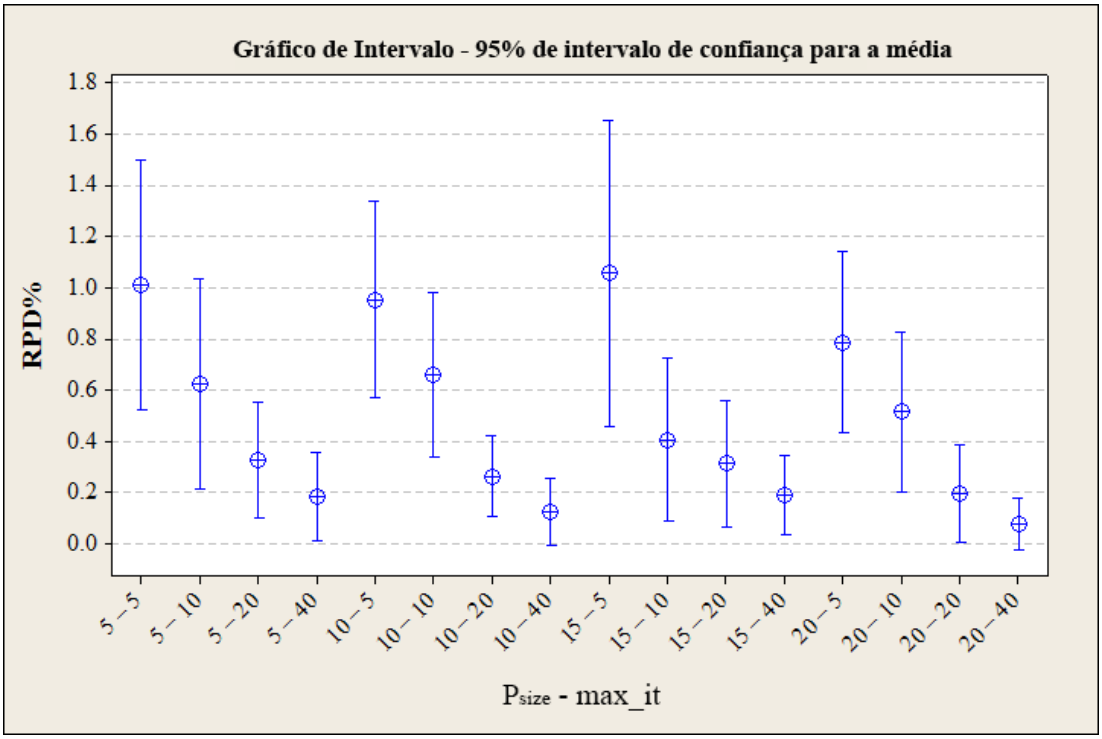


Figura 4.2: Gráfico de intervalos para os parâmetros P_{size} e max_it

4.1.3 Análise dos parâmetros ls_it e max_it do GRASP-H

Também foi feita uma calibragem dos parâmetros do GRASP-H. Nesse caso, os parâmetros analisados foram os ls_it, que controla a quantidade de iterações que a busca local irá fazer e o max_it que é o critério de parada do algoritmo.

Inicialmente, analisou-se os efeitos da variação do parâmetro ls_it em relação à qualidade das soluções obtidas. Para esse teste, foi fixado o valor de max_it em 40 iterações e os valores testados de ls_it foram: $10, 25, 50, 100$ iterações. Para cada valor do parâmetro, o algoritmo foi executado cinco vezes para cada instância e a média do valor da função objetivo nos cinco primeiros rounds foi escolhida para a análise.

A Figura 4.3 mostra o gráfico de intervalos com 95% de intervalo de confiança para a média. Como os intervalos estão sobrepostos, os resultados não são estatisticamente diferentes. No entanto, pode-se ver que para $ls\_it = 10$ iterações, os resultados mostraram uma maior variação e a pior média. Com $ls\_it = 100$ iterações foram obtidas melhores soluções na média, entretanto, seu custo computacional é muito maior. Assim, para reduzir o tempo de execução do algoritmo, optou-se pela escolha de 50 iterações como sendo o valor de ls_it.

Com o parâmetro ls_it definido, deve-se fazer uma análise para chegarmos no valor adequado para o parâmetro max_it. Para tal, foi testado o algoritmo para $max\_it =$

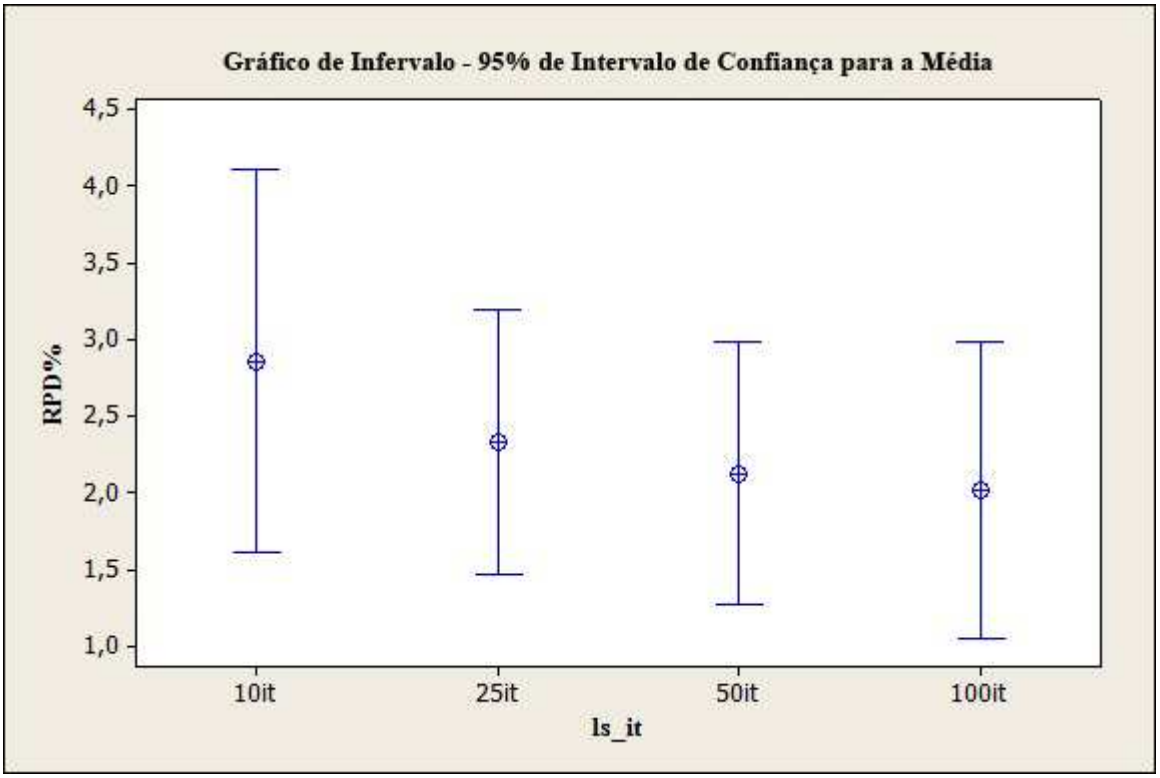


Figura 4.3: Gráfico de intervalos para o parâmetro ls_it

$\{5, 10, 20, 40\}$ iterações sem melhora. Na Figura 4.4 pode-se ver que não há diferença significativa entre os valores, já que os intervalos estão sobrepostos. No entanto, 40 iterações obteve na média, melhores soluções. Para reduzir o custo computacional, optou-se por utilizar $max\_it = 20$ iterações sem melhora.

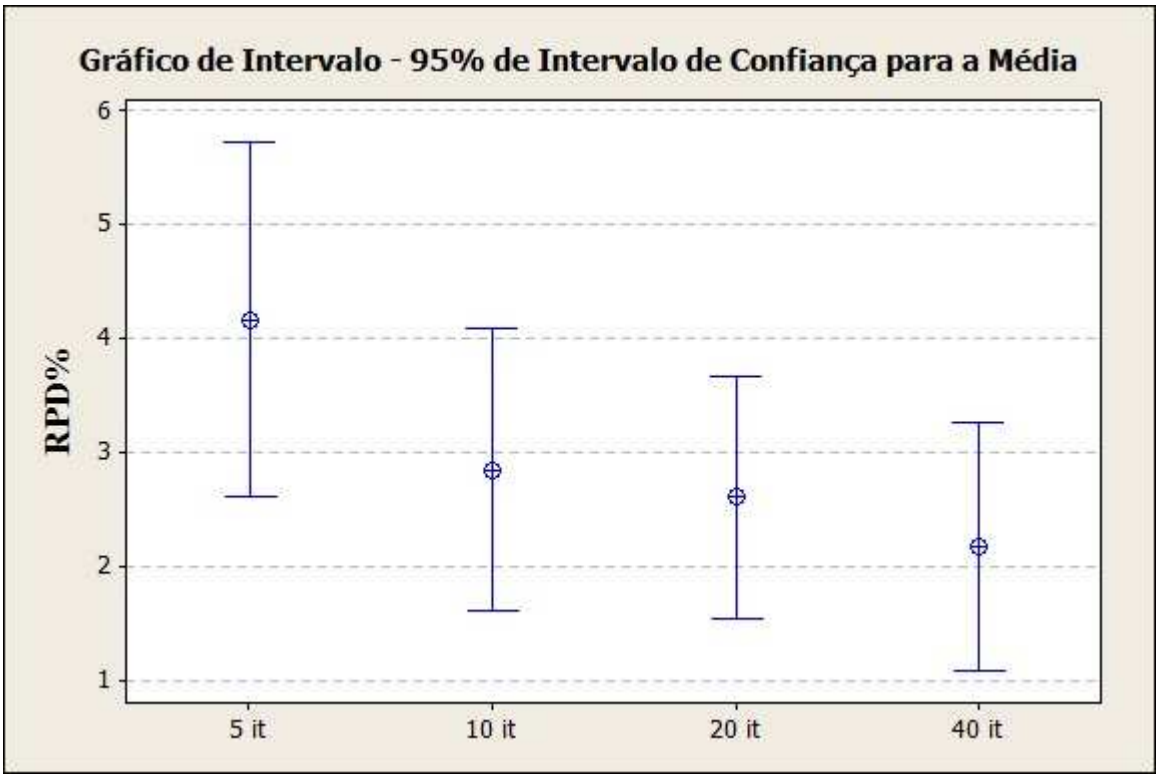


Figura 4.4: Gráfico de intervalos para o parâmetro *max_it*

Capítulo 5

Testes e resultados computacionais

Todos os algoritmos foram implementados em C++ e os testes foram executados em um Intel Core 2 Quad 3.2 GHz com 3 GB de memória RAM rodando o sistema operacional Ubuntu 10.04 64 bits sem a utilização de técnicas de *multithread*.

Todos os algoritmos trabalham com o conceito de *rounds*. Conforme dito anteriormente, cada *round* é composto de uma fase de *setup* seguida por uma fase de transmissão. Os algoritmos de clusterização são executados durante a primeira fase. Neste trabalho, o tempo de duração da fase de transmissão adotada foi definida como uma quantidade de dados que chega ao *sink*. Em outras palavras, esse é o tempo que a rede permanecerá com a mesma topologia.

A simulação termina quando a rede é considerada inativa, que foi definido como no mínimo 5% dos sensores iniciais estiverem vivos. O fluxograma da Figura 5.1 ilustra passo a passo a simulação dos algoritmos. A Tabela 5.1 sumariza todos os parâmetros e constantes adotados pelos algoritmos durante a simulação.

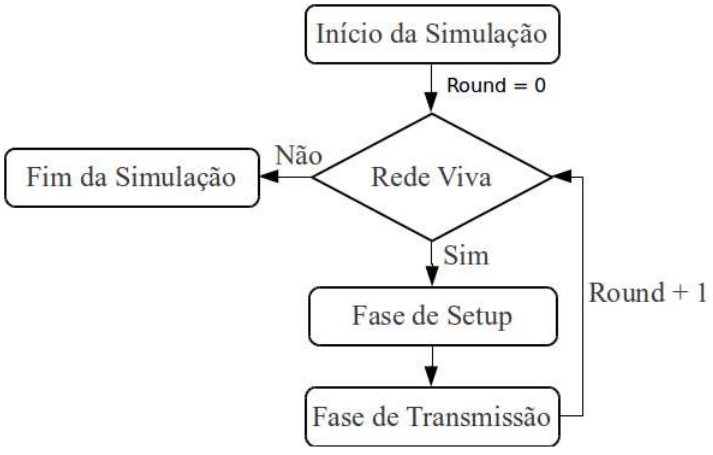


Figura 5.1: Fluxograma da simulação

Tabela 5.1: Parâmetros dos algoritmos e da simulação

Energia inicial dos sensores	$1J$
Consumo de energia do rádio (E_{elec})	$50\text{n}J/\text{bit}$
Consumo de energia do amplificador (ϵ_{amp})	$10pJ/\text{bit}/m^2$
Consumo de energia da agregação de dados (E_{DA})	$5nJ/\text{bit}$
Quantidade de *cluster heads* (n_{ch})	5% dos sensores vivos
Parâmetro da busca local do GRASP-C e GRASP+PR (α)	0.5
Parâmetro da busca local do GRASP-H (ls_it)	50 iterações
Critério de parada dos algoritmos (max_it)	20 iterações sem melhora
Tamanho do *Pool* de soluções do *Path Relinking* (P_{size})	10 soluções

Para cada teste, foram feitas 5 execuções de cada instância e a média entre essas execuções foram utilizadas como resultado. Inicialmente, foi comparado os algoritmos GRASP-C e GRASP+PR com o LEACH e LEACH-C, para a formação dos *clusters* em um único nível. Posteriormente, o GRASP-H foi comparado com o GRASP-C, LEACH-C e o EEMC, a fim de testar a eficiência de múltiplos níveis de *clusters* em relação a um único nível e utilizou-se o EEMC para validar os resultados obtidos com um protocolo cujo objetivo é o mesmo. Também foi feita uma análise entre os algoritmos desenvolvidos em relação ao tempo de execução a fim de verificar a influência do tempo em relação a qualidade das soluções obtidas.

5.1 Resultados do GRASP-C e GRASP+PR

Para avaliar a qualidade das soluções obtidas por cada algoritmo, foi feita uma análise do valor médio da função objetivo(distância total) obtida por cada um nos cinco primeiros *rounds*. Esse critério foi adotado devido a morte de sensores durante o decorrer da simulação, alterando a rede. Com isso, o valor da distância total seria influenciado pela quantidade de sensores ativos, sendo injusto uma comparação entre algoritmos onde a rede possui diferentes quantidade de sensores. Nos *rounds* iniciais da simulação é garantido que todos os sensores da rede estarão vivos, garantindo um mesmo patamar entre todos os algoritmos para a comparação.

Foram utilizado três critérios básicos para comparação entre os algoritmos. São eles: distância total (valor da função objetivo), dissipação de energia da rede e longevidade da rede (tempo de vida). Os algoritmos foram comparados utilizando o RPD.

5.1.1 Comparações em relação ao valor da distância total

Na Tabela 5.2 é apresentado a variação percentual do valor da função objetivo da cada algoritmo nos cinco primeiros *rounds* em relação a melhor solução obtida. As instâncias foram divididas em conjuntos, de acordo com a quantidade de sensores. Como pode ser visto, os algoritmos propostos GRASP-C e GRASP+PR obtiveram resultados consideravelmente melhores do que os protocolos da literatura. Em todos os conjuntos de instâncias observa-se que o GRASP+PR foi melhor que os algoritmos comparados, provando a eficiência do *Path Relinking*. Pode-se perceber também que a diferença dos algoritmos em relação ao GRASP+PR aumenta conforme o tamanho da rede aumenta.

Tabela 5.2: Valor do desvio relativo percentual médio da função objetivo nos cinco primeiros *rounds*

Conjunto de instâncias	LEACH	LEACH-C	GRASP-C	GRASP+PR
50 sensores	116,11	18,83	0,15	**0**
60 sensores	134,01	37,44	0,55	**0**
70 sensores	140,22	35,48	0,88	**0**
80 sensores	151,16	46,88	1,05	**0**
90 sensores	150,36	43,25	0,74	**0**
100 sensores	149,56	49,13	1,83	**0**
125 sensores	170,56	71,74	1,91	**0**
150 sensores	186,25	85,36	2,30	**0**
200 sensores	209,78	117,26	3,25	**0**
300 sensores	246,63	162,16	3,87	**0**

O gráfico apresentado na Figura 5.2 mostra a comparação dos valores de RPD relacionados ao custo da solução para cada uma das 100 instâncias testadas. Como pode-se perceber, o LEACH obteve os piores valores de RPD quando comparado com os demais algoritmos. Esse resultado já era esperado devido à natureza aleatória do LEACH. O LEACH-C provou ser mais eficiente que o LEACH, porém, com o aumento

da quantidade de sensores da rede seu desempenho cai consideravelmente quando comparado com o GRASP-C e o GRASP+PR.

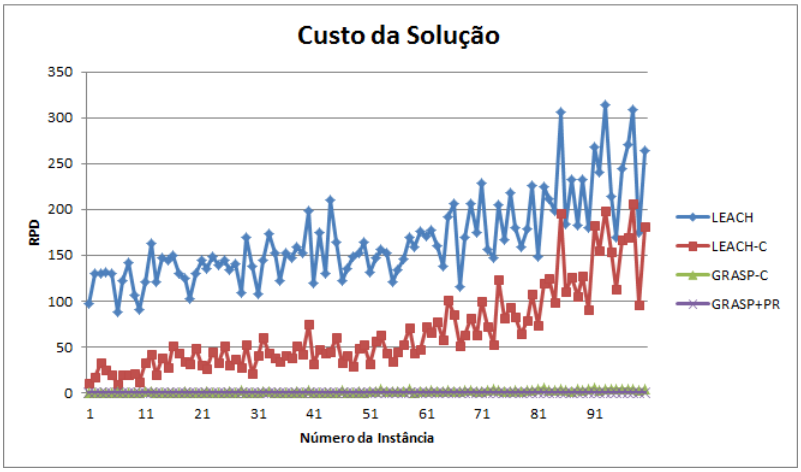


Figura 5.2: Comparação do custo da função objetivo entre os algoritmos

Uma comparação mais detalhada foi feita entre os algoritmos propostos. Os dois algoritmos obtiveram melhores resultados do que o LEACH e LEACH-C para todas as instâncias testadas. Enquanto que para as instâncias menores o GRASP-C consegue obter resultados próximos aos do GRASP+PR, conforme o tamanho da rede aumenta, o desempenho do GRASP-C piora em relação ao GRASP+PR. Dessa forma, a diferença entre a qualidade das soluções obtidas pelos dois algoritmos aumenta. Como podemos ver no gráfico da Figura 5.3, o GRASP+PR não apresenta nenhuma variação quando comparado ao GRASP-C. Com isso, podemos dizer que o GRASP+PR é estatisticamente melhor do que o GRASP-C, em relação ao valor da função objetivo. A fase de intensificação baseada no *Path Relinking* utilizada no GRASP+PR provou ser uma estratégia eficiente para obter soluções ainda melhores.

Como o consumo de energia é relativo à distância de transmissão, que é utilizada para calcular o valor da função objetivo, espera-se que soluções com menor valor da função objetivo consumam menos energia. O gráfico apresentado na Figura 5.4 mostra o resultado dos testes do ANOVA realizados comparando o consumo de energia nos primeiros cinco rounds. Novamente, o LEACH e o LEACH-C apresentaram os piores valores de RPD. Quando comparados em relação ao consumo de energia, o GRASP-C e o GRASP+PR não são estatisticamente diferente. Isso quer dizer que, na média, o comportamento dos algoritmos é semelhante. No entanto, os algoritmos propostos são estatisticamente diferentes em relação ao LEACH e LEACH-C.

Uma rede que consome menos energia, tende a permanecer viva por mais *rounds*. Em uma rede com 100 sensores, a Figura 5.5 mostra o consumo de energia a cada *round*

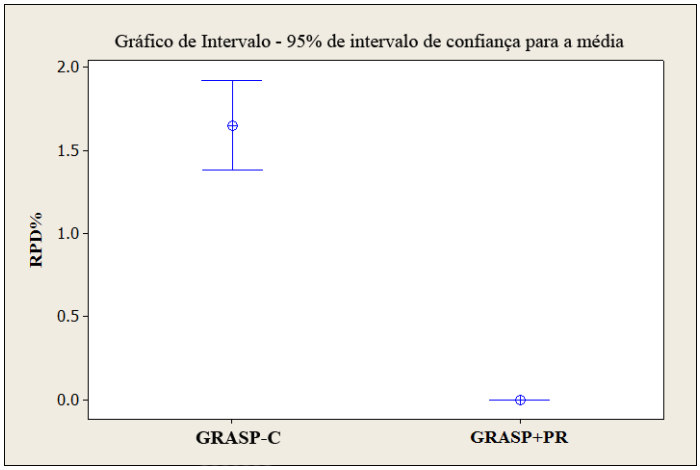


Figura 5.3: Gráfico de intervalos comparativo entre o GRASP-C e o GRASP+PR quanto à distância total

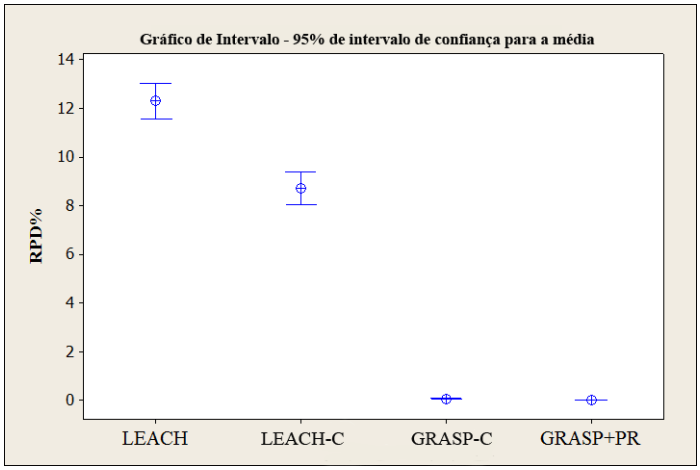


Figura 5.4: Gráfico de intervalos comparativo dos algoritmos em relação ao consumo de energia

para cada um dos algoritmos. A comparação é feita utilizando a energia residual total da rede, ou seja, o somatório da energia restante de cada um dos sensores a cada *round*. Pode-se ver que o consumo de energia elevado do LEACH (em comparação aos demais) acarreta na morte da rede de forma prematura, apresentando quedas de energia consideráveis nos *rounds* finais. Diferente do LEACH, os demais algoritmos apresentam um consumo de energia mais homogêneo. Isso acontece devido a eficiência na escolha dos *cluster heads*. Os algoritmos propostos foram capazes de manter a rede viva por mais *rounds*, com o GRASP+PR sendo superior ao GRASP-C.

O consumo de energia da rede pode ser relacionado com o valor da função objetivo de cada algoritmo. Quanto menor o valor da função objetivo, menos energia será necessária para realizar as transmissões até a estação base. Assim, a rede será capaz

de se manter viva por mais *rounds*.

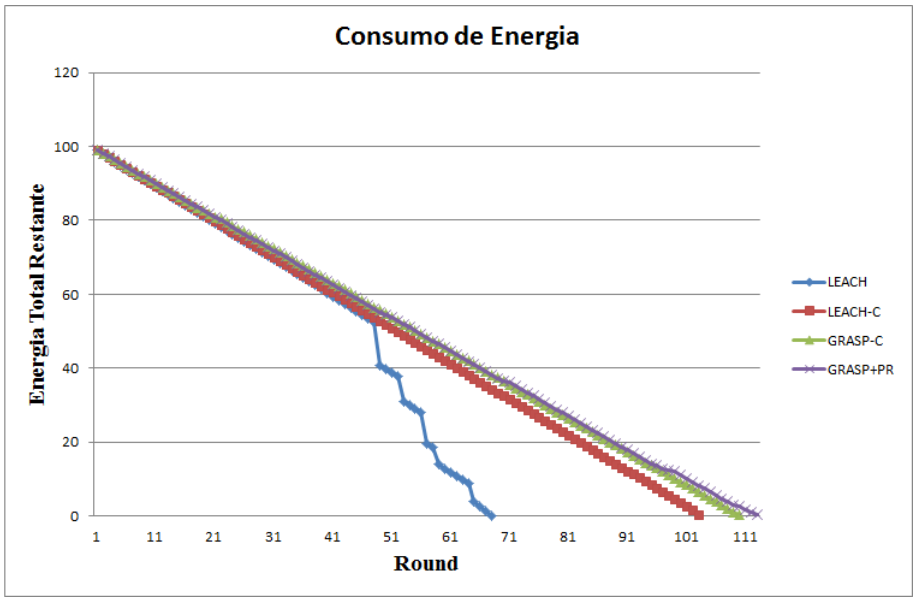


Figura 5.5: Consumo de energia a cada *round* em uma rede com 100 sensores

5.1.2 Comparações em relação a dissipação de dnergia e quantidade de transmissões

Os gráficos das Figuras 5.6 e 5.7 apresentam uma comparação em relação ao total de transmissões bem sucedidas realizadas e a quantidade de energia dissipada para realizá-las para uma instância de 100 e 200 sensores respectivamente. Uma transmissão é considerada bem sucedida quando ela chega ao *sink*.

Pode-se perceber que uma formação ineficaz dos *clusters* têm um grande impacto no final da simulação, quando os sensores possuem quantidades baixas de energia. No LEACH, depois de 50% da energia da rede ter sido dissipada, há um decréscimo considerável na quantidade de transmissões realizadas, enquanto os demais algoritmos se apresentam de forma mais homogênea. Novamente, o GRASP+PR se mostrou mais eficiente, realizando mais transmissões que as demais abordagens.

A Tabela 5.3 apresenta a quantidade média de transmissões bem sucedidas (transmissões que chegam ao *sink*) realizadas pela rede em cada conjunto de instâncias. Os algoritmos propostos realizaram consideravelmente mais transmissões do que os protocolos da literatura. O GRASP+PR obteve melhores resultados novamente. Nas maiores instâncias a diferença entre o GRASP+PR e o LEACH-C foi de aproximadamente 12% e 42% em relação ao LEACH.

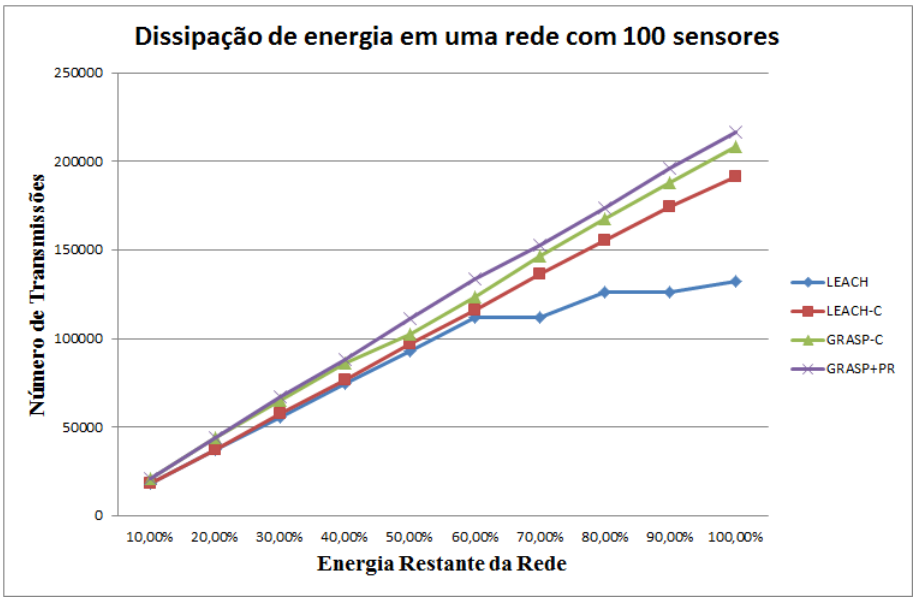


Figura 5.6: Dissipação de energia em relação a quantidade de transmissões realizadas em uma instância de 100 sensores

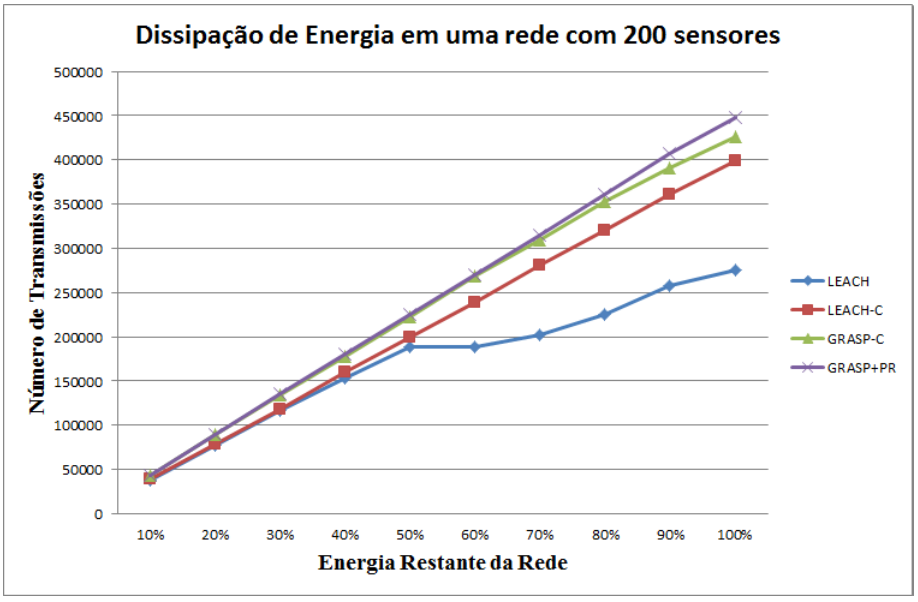


Figura 5.7: Dissipação de energia em relação a quantidade de transmissões realizadas em uma instância de 300 sensores

O gráfico apresentado na Figura 5.8 fornece uma visualização dos dados da Tabela 5.3, para cada uma das 100 instâncias. Como pode ser observado, para todas as instâncias, os algoritmos propostos foram capazes de enviar uma quantidade maior de mensagens para o *sink* em comparação com os algoritmos da literatura, confirmando a existência de uma diferença significativa entre os algoritmos.

Tabela 5.3: Quantidade total média de transmissões realizadas pela rede ($\times 10^3$)

Conjunto de instâncias	LEACH	LEACH-C	GRASP-C	GRASP+PR
50 sensores	65	87	90	**91**
60 sensores	79	109	112	**118**
70 sensores	90	131	127	**138**
80 sensores	108	150	160	**168**
90 sensores	118	173	184	**185**
100 sensores	138	191	205	**214**
125 sensores	175	244	267	**273**
150 sensores	211	296	325	**330**
200 sensores	288	398	436	**449**
300 sensores	441	608	671	**683**

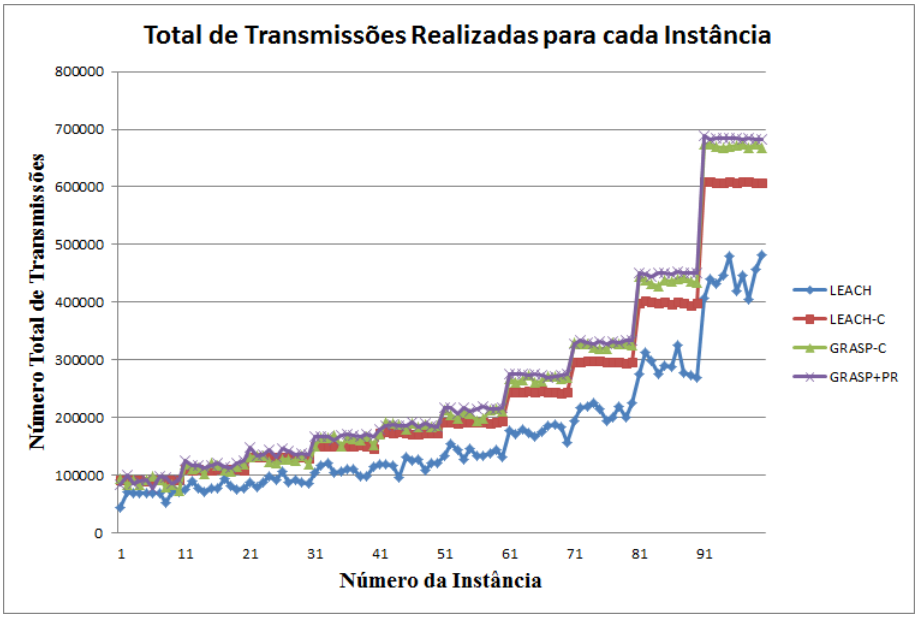


Figura 5.8: Quantidade de transmissões realizadas por cada uma das instâncias.

5.2 Resultados do GRASP-H

Neste capítulo são apresentados os resultados obtidos pelo GRASP-H, que possui como característica principal, formar uma rede com múltiplos níveis, diferentemente dos demais algoritmos testados até agora onde apenas um único nível era formado. O EEMC foi utilizado como comparação e foi submetido ao mesmo conjunto de testes. Como critério de comparação, foram utilizados o valor médio da função objetivo, a média

da quantidade de sensores vivos a cada *round*, e a média de dissipação de energia por *round*.

5.2.1 Comparação em relação ao valor da função objetivo

O gráfico da Figura 5.9 mostra a comparação do valor da função objetivo para cada instância nos cinco primeiros *rounds*. O LEACH e LEACH-C obtiveram as piores médias para o valor da função objetivo enquanto que o algoritmo EEMC obteve melhores resultados que o LEACH e LEACH-C, indicando que topologias multi-nível oferecem uma alternativa interessante quando comparadas com topologias com um único nível de clusterização. Porém, nota-se também que apesar do GRASP-C realizar clusterização em apenas um nível, ele obteve resultados consideravelmente melhores do que o EEMC. O GRASP-H obteve os melhores resultados entre todos os algoritmos testados, provando que a utilização de meta-heurísticas para a formação de *clusters* de maneira eficiente, aliada a uma topologia multinível oferece resultados consideravelmente melhores do que os outros algoritmos da literatura testados.

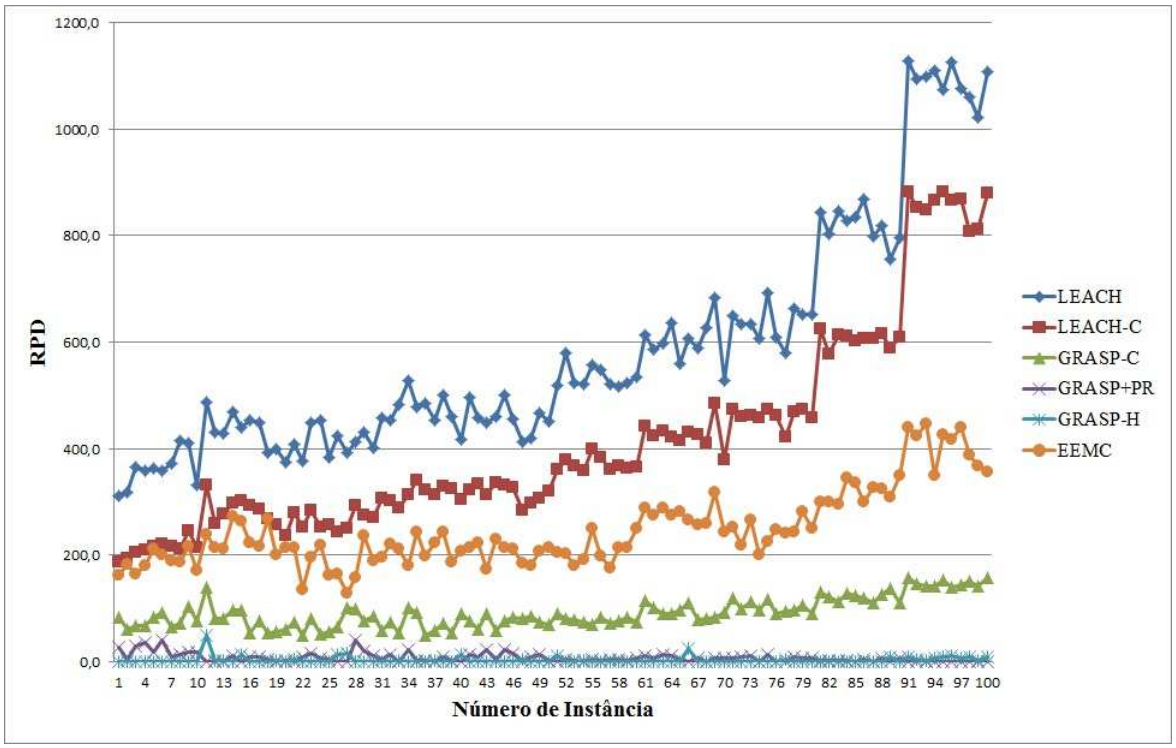


Figura 5.9: Valor médio da função objetivo para cada um dos algoritmos.

A análise do ANOVA mostra que todos os algoritmos são estatisticamente diferentes entre si. Na Figura 5.9 é fornecido uma comparação mais detalhada entre todos os algoritmos. Pode-se observar que o EEMC é estatisticamente melhor que os

algoritmos LEACH e LEACH-C. Ambos algoritmos GRASP-C e GRASP-H obtiveram os melhores resultados sendo que o segundo obteve o melhor desempenho em todas as instâncias, sendo estatisticamente melhor que todos os outros algoritmos testados.

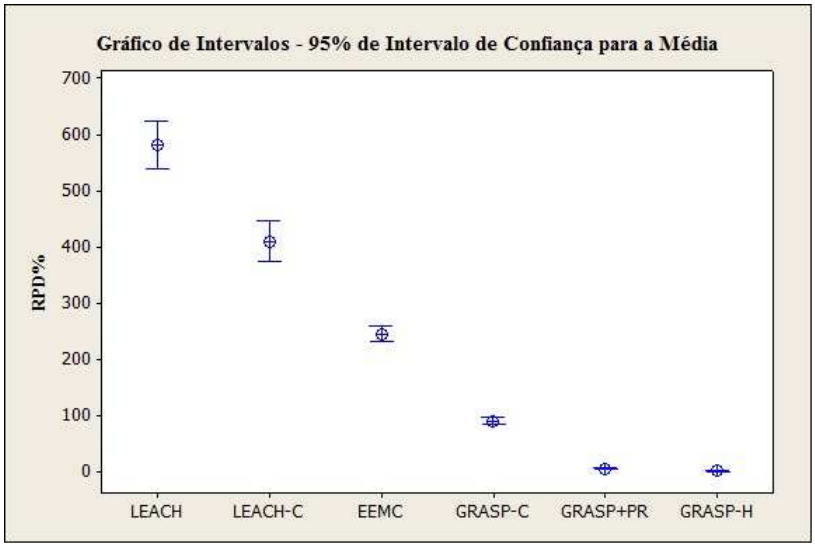


Figura 5.10: Gráfico de intervalos relativo ao valor da função objetivo para cada um dos algoritmos testados.

5.2.2 Comparação em relação a quantidade de sensores vivos e ao consumo de energia

Na Figura 5.11 está representada a quantidade de sensores vivos por *round*, para cada algoritmo, em uma instância de 300 sensores. Pode-se observar que com o LEACH os sensores esgotam a energia muito rapidamente. Apesar de no GRASP-C os sensores começarem a morrer mais cedo, a rede se mantém viva por mais tempo que o LEACH e o EEMC. A morte dos sensores no GRASP-C se inicia antes dos demais algoritmos devido a função de elegibilidade utilizada. No entanto, o GRASP-C consegue manter a rede viva por mais *rounds* do que o EEMC. Apesar da diferença entre as topologias, os melhores resultados obtidos pelo GRASP-C indicam que a formação de *clusters* de maneira inteligente proporciona um ganho considerável na qualidade das soluções obtidas. Os melhores resultados foram obtidos pelo GRASP-H, com o qual a rede foi mantida viva por mais *rounds* que os demais algoritmos, indicando que a clusterização multi-nível é uma abordagem eficiente para economizar energia da rede, estendendo seu tempo de vida.

O gráfico da Figura 5.12 mostra a energia total dissipada pelos sensores em cada *round*, para cada algoritmo. Para o LEACH e LEACH-C os sensores consomem mais

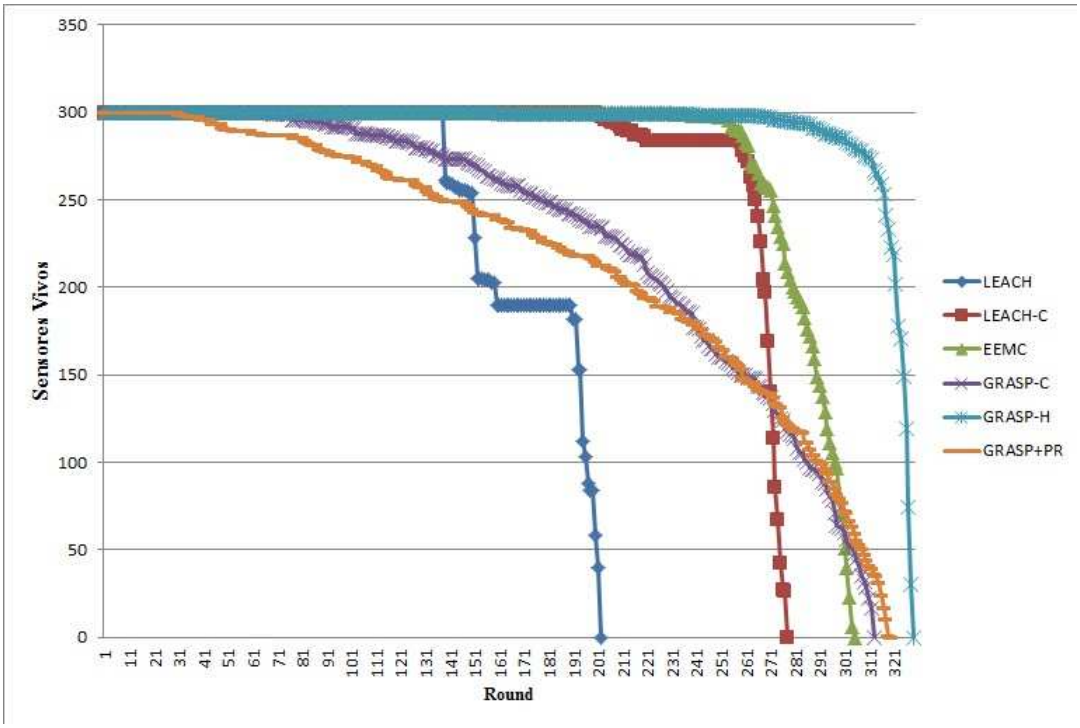


Figura 5.11: Quantidade de sensores vivos a cada *round* para cada algoritmo em uma instância de 300 sensores.

energia por rounds, enquanto para o GRASP-H os sensores consome menos energia. Comparando-se apenas os algoritmos de natureza aleatória, o algoritmo EEMC obteve resultados consideravelmente melhores que o LEACH e o LEACH-C, indicando que a utilização de vários níveis de clusterização é um método muito eficiente para economizar energia. Além disso, percebemos também que com a adoção de técnicas para formar os *clusters* de maneira mais inteligente, obteve-se um ganho considerável de desempenho devido a distribuição uniforme e a rotatividade dos *cluster heads*, garantindo assim um consumo de energia mais homogêneo pela rede.

5.3 Análise do Tempo de Execução dos Algoritmos

Uma análise importante a ser feita é em relação ao tempo de execução de cada algoritmo. Foram excluídos dessa análise os algoritmos LEACH e o EEMC por se tratarem de métodos puramente probabilísticos. O gráfico da Figura 5.13 apresenta o tempo médio (em segundos) para a execução de um *round* por cada algoritmo. O LEACH-C se destacou dos algoritmos propostos por ser um método rápido. Isso pode ser explicado devido a natureza do seu método de busca de vizinhança, não exigindo a análise de uma combinação de possibilidades. No entanto, esse tipo de método não faz uma

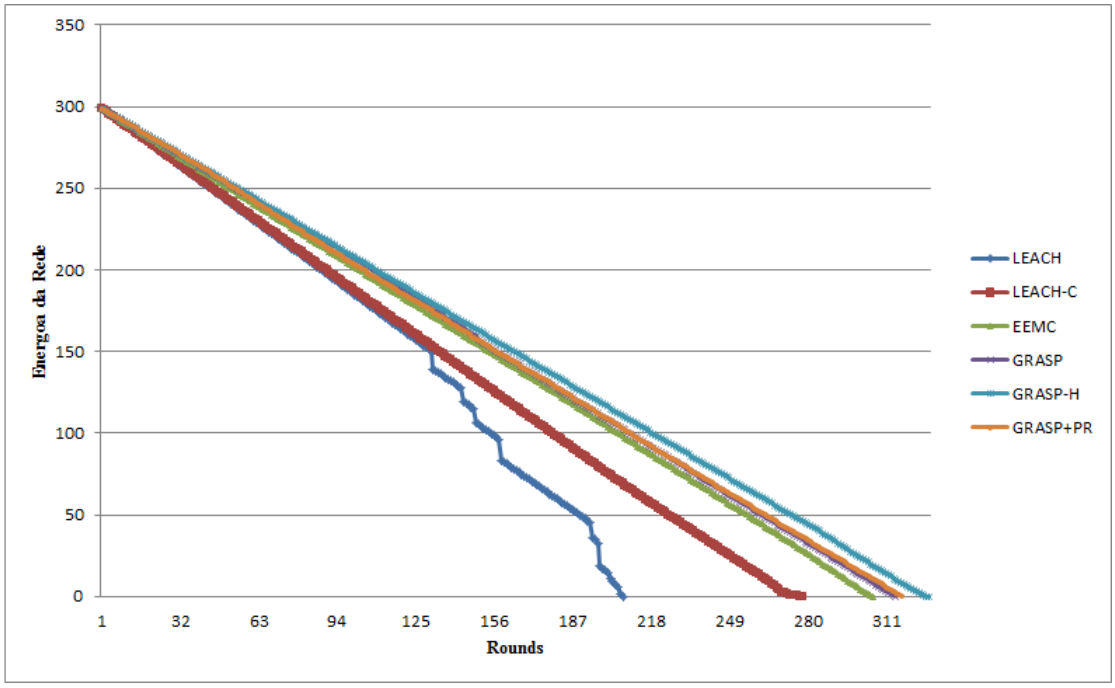


Figura 5.12: Consumo de energia a cada *round* para cada um dos algoritmos em uma instância de 300 sensores.

busca exaustiva na vizinhança, o que explica a baixa qualidade das soluções obtidas pelo algoritmo quando comparadas aos algoritmos propostos.

Dentre os algoritmos propostos, o GRASP-C é o método mais rápido, por possuir uma busca local que explora menos a vizinhança do que os demais algoritmos desenvolvidos. No GRASP-H, como a heurística é executada diversas vezes a cada *round* para as seguidas subdivisões de cada *cluster*, seu tempo de execução acaba sendo maior. É importante ressaltar que no GRASP-H a busca local é uma versão simplificada da utilizada pelo GRASP-C. O GRASP+PR possui o tempo de execução mais elevado. O método de *Path Relinking* apresentou uma melhora significativa nas soluções obtidas pela heurística. No entanto, sua adição ao GRASP ocasiona um considerável aumento no tempo de execução do algoritmo.

Uma melhor visualização dos resultados pode ser vista na Tabela 5.4, que apresenta a média do tempo de execução dos algoritmos agrupados de acordo com a quantidade de sensores.

A diferença entre o tempo de execução de cada método é considerável. Dessa forma, outro critério de comparação interessante para ser analisado é a qualidade das soluções obtidas por cada algoritmo em um mesmo tempo de execução. Para tal teste, cada algoritmo foi executado cinco vezes para eleger os *cluster heads* em um único *round*. O critério de parada utilizado foi o tempo de execução, dando a cada algoritmo

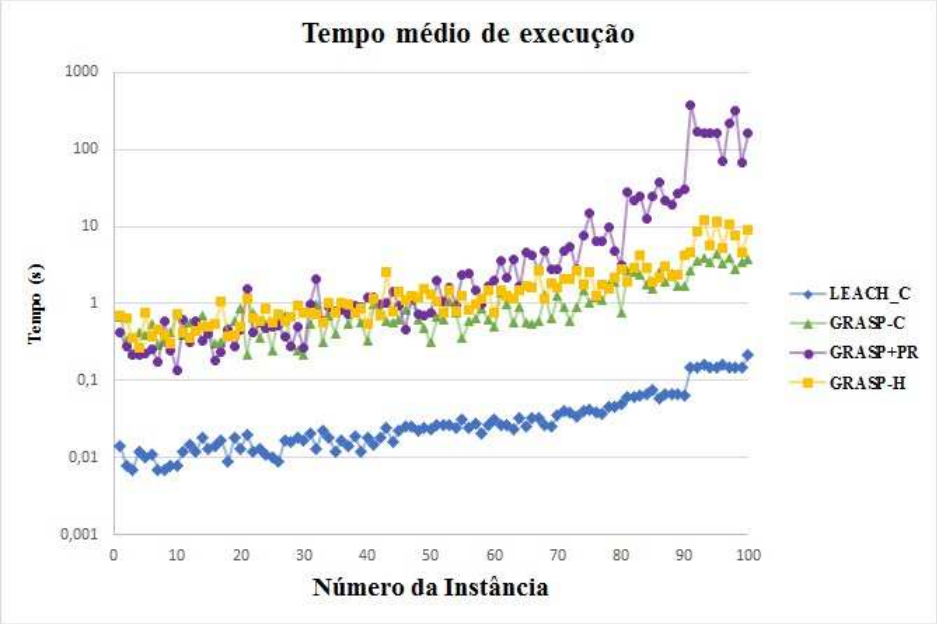


Figura 5.13: Tempo médio de execução por *round* para cada algoritmo

Tabela 5.4: Tempo médio de execução (em segundos) por *rounds* de cada algoritmo

Conjunto de instâncias	LEACH-C	GRASP-C	GRASP+PR	GRASP-H
50 sensores	0,0092	0,4322	0,2769	0,4971
60 sensores	0,0141	0,5127	0,3828	0,5047
70 sensores	0,0143	0,4257	0,5403	0,7545
80 sensores	0,0166	0,6045	0,9834	0,79
90 sensores	0,0214	0,6839	0,9453	1,3009
100 sensores	0,0264	0,6655	1,6528	1,0481
125 sensores	0,0285	0,869	3,2893	1,5792
150 sensores	0,0411	1,1622	6,6492	2,0692
200 sensores	0,0649	2,1024	24,6612	2,775
300 sensores	0,1581	3,5084	186,9042	7,9521

1 minuto para obter a melhor solução possível.

Esse tipo de comparação é mais justo, já que os tempos de execução e critério de parada dos algoritmos são distintos. Assim, a eficiência de cada método é comparada tomando como base a média das cinco execuções de cada algoritmo para todas as instâncias geradas. O GRASP-H foi omitido dessa comparação por se tratar de um tipo de topologia diferente dos demais algoritmos. O LEACH foi deixado de fora dessa

comparação por ser um método puramente probabilístico, não possuindo nenhuma técnica de inteligência computacional que pudesse melhorar a solução ao longo do tempo.

O gráfico da Figura 5.14 mostra o comparativo entre cada algoritmo. Como pode-se perceber, o LEACH-C obteve as piores médias, indicando que os algoritmos propostos são claramente superiores aos da literatura. No mesmo tempo de execução a média das soluções obtidas pelos algoritmos GRASP-C e GRASP+PR foram consideravelmente superiores ao LEACH-C. Com isso, pode-se concluir que o aumento do tempo de execução dos algoritmos propostos é justificado com a obtenção de soluções melhores.

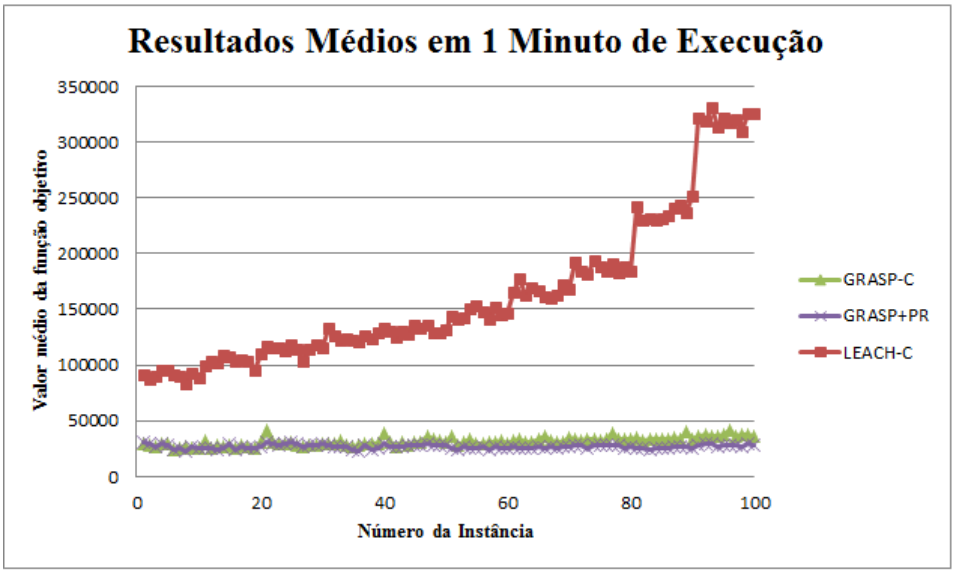


Figura 5.14: Valor médio da função objetivo de cada algoritmo em 1 minuto de execução

Capítulo 6

Conclusão

Este trabalho abordou o problema de clusterização em redes de sensores sem fio, que é um problema conhecidamente NP-Difícil. Assim, optou-se pela utilização de técnicas heurísticas a fim de obter resultados satisfatórios em tempo computacional viável.

Foram desenvolvidos três algoritmos baseados em GRASP, que é uma meta-heurística bastante utilizada para resolver problemas de otimização combinatória. A primeira versão do algoritmo, chamada de GRASP-C foi implementada utilizando técnicas de construção de soluções e busca local empregadas comumente para o problema das P-Medianas. A fim de melhorar os resultados obtidos, a segunda versão, chamada de GRASP+PR foi desenvolvida acrescentando uma fase de pós-otimização baseada no *Path Relinking*. Enquanto o GRASP-C e o GRASP+PR formam *clusters* em apenas um nível, o GRASP-H foi desenvolvido para se obter topologias multi-nível.

A fim de determinar a melhor configuração para cada algoritmo, foi feito um estudo de calibragem de parâmetros para cada algoritmo. Com isso, pode-se observar o impacto que cada parâmetro tem no algoritmo e foi possível determinar o valor com melhor custo/beneficio para cada parâmetro. Devido à falta de instâncias na literatura, foi gerado um conjunto de 100 instâncias distintas, com quantidade de sensores variando de 50 até 300.

Foram feitos diversos testes com algoritmos de clusterização presentes na literatura. O GRASP-C e o GRASP+PR foram comparados com o LEACH e o LEACH-C. O GRASP-H foi comparado com o LEACH, LEACH-C e GRASP-C a fim e averiguar a diferença entre a topologia com um único nível e da topologia multi-nível. O GRASP-H também foi comparado com o EEMC para topologias multiníveis. Foram feitas cinco execuções de cada algoritmo para cada instância e a média dessas cinco execuções foi utilizada como resultado final comparativo.

Para o GRASP-C e o GRASP+PR foram utilizados os seguintes critérios para

comparação: Valor da função objetivo, o consumo de energia por *round* e a quantidade de transmissões realizadas. Para o GRASP-H, os critérios foram: o valor da função objetivo, quantidade de sensores vivos por *round*, e a quantidade de energia dissipada pela rede.

Os resultados obtidos pelo GRASP-C e GRASP+PR provaram que os algoritmos propostos são consideravelmente mais eficientes do que os da literatura, aumentando o tempo de vida da rede e efetuando mais transmissões. A adição do *Path Relinking* se provou uma estratégia eficiente, já que os resultados obtidos pelo GRASP+PR foram superiores ao do GRASP-C. O GRASP-H tambem provou ser mais eficiente do que o os protocolos da literatura, obtendo desempenho superior também ao GRASP-C. Os resultados obtidos pelo EEMC foram melhores do que o LEACH e LEACH-C, indicando que a adoção de topologias multi-níveis é mais eficiente do que a de um único nível.

Para trabalhos futuros, pode-se empregar o uso de outros métodos heurísticos ainda não utilizados neste problema. Devido a diversidade das aplicações das RSSF, existem muitos tipos de redes com características particulares (como por exemplo sensores /*cluster heads* móveis, sensores com diferentes capacidades de processamento e diferentes tipos de coletas de dados) que podem ser abordadas utilizando clusterização com meta-heurísticas.

Referências Bibliográficas

Abbasi, A. A. e Younis, M. A survey on clustering algorithms for wireless sensor networks. *Computer Communications 30*, p. 2826–2841, 2007.

Agarwal, P. e Procopiuc, C. Exact and approximation algorithms for clustering. *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, p. 658–667, 1999.

Akkaya, K. e Younis, M. A survey on routing protocols for wireless sensor networks. *Elsevier Journal of Ad Hoc Networks*, volume 3, p. 325?349, 2005.

Akyildiz, I. F., W. Su, Y. S. e Cayirci, E. Wireless sensor networks: a survey. *Computer Networks 38*, p. 393–422, 2002.

Brittes, M. P. Uma proposta para melhoria de desempenho do protocolo leach para rssf. Dissertação de mestrado, 2007.

Chan, E. e Han, S. Energy efficient residual energy monitoring in wireless sensor networks. *International Journal of Distributed Sensor Networks*, volume 5, p. 1–23, 2009.

Chatterjee, S. e Singh, M. A centralized energy-efficient routing protocol for wireless sensor networks. *Advanced Networking and Applications*, volume 3, p. 12–18, 2012.

Dasgupta, K., Kalpakis, K. e Namjoshi, P. An efficient clustering?based heuristic for data gathering and aggregation in sensor networks. *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

Feo, T. A. e Resende, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization 6*, p. 109–133, 1995.

Glover, F. Tabu search and adaptive memory programing. *Interface in Computer Science and Operational Reaserch*, p. 1–75, 1996.

Godbole, V. Performance analysis of clustering protocol using fuzzy logic for wireless sensor network. *IAES International Journal of Artificial Intelligence (IJ-AI)*, volume 1, p. 103–111, 2012.

Gupta, I., Riordan, D. e Sampalli, S. Cluster-head election using fuzzy logic for wireless sensor networks. *Communication Networks and Services Research Conference*, p. 255–260, 2005.

Heinzelman, W. B., Chandrakasan, A. P. e Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. *33rd Hawaii International Conference on System Sciences*, 2000.

Heinzelman, W. B., Chandrakasan, A. P. e Balakrishnan, H. An application specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Networking, vol. 1, no.4*, p. 660–670, 2002.

Hou, Y., Shi, Y. e Sherali, H. On energy provisioning and relay node placement for wireless sensor networks. *IEEE Transactions on Wireless Communications 4*, volume 4, p. 2579–2590, 2005.

Ishibuchi, H. e Murata, T. Performance evaluation of genetic algorithms for flowshop scheduling problems. *Proc. 1st IEEE Conf. Evolutionary Computation*, volume 2, p. 812–817, 2004.

Jin, Y., Wang, L., Kim, Y. e Yang, X. Eemc: An energy-efficient multi-level clustering algorithm for large-scale wireless sensor networks. *Computer Networks Vol. 52, No. 3*, p. 542–562, 2008.

Kumar, S. e Chauhan, S. A survey on scheduling algorithms for wireless sensor networks. *International Journal of Computer Applications*, volume 20, 2011.

Latiff, N. M. A., Tsimenidis, C. C. e Sharif, B. S. Energy-aware clustering for wireless sensor networks using particle swarm optimization. *The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2007.

Lindsey, S. e Raghavendra, C. Pegasis: power efficient gathering in sensor information systems. *Proceedings of the IEEE Aerospace Conference*, 2002.

Mehrjoo, S., Aghaee, H. e Karimi, H. A novel hybrid ga-abc based energy efficient clustering in wireless sensor network. *Canadian Journal on Multimedia and Wireless Networks*, volume 2, 2011.

Montegomery, D. *Design and Analysis of Experiments*. John Willy and Sons, 2007.

Nagpal, R. e Coore, D. An algorithm for group formation in an amorphous computer. *Proceedings of the 10th International Conference on Parallel and Distributed Systems (PDCS?98)*, 1998.

Padmanabhan, K. e Kamalakkannan, P. A study on energy efficient routing protocols in wireless sensor networks. *European Journal of Scientific Research*, volume 4, p. 499–511, 2011.

Pottie, G. J. e Kaiser, W. J. Wireless integrated network sensors. *Communication ACM*, volume 5, 2000.

Rajagopalan, R. e Varshney, P. K. Data aggregation techniques in sensor networks: A survey. *IEEE Communication Surveys and Tutorials, Vol. 8, No. 4*, p. 48–63, 2006.

Resende, M. G. C. e Ribeiro, C. C. Metaheuristics: Progress as real problem solvers. *Kluwer Academic Publishers*, 2005.

Resende, M. G. C. e Werneck, R. F. A hybrid heuristic for the p-median problem. *Technical Report TD-5NWRCR, AT&T Labs Research*, 2003.

Ruiz, L. B. *MANNA: A Management Architecture for Wireless Sensor Network*. Tese de doutorado, 2003.

S. Banerjee, S. K. A clustering scheme for hierarchical control in multi-hop wireless networks. *Proceedings of 20th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM?01)*, 2001.

S. Jin, M. Z. e Wu, A. S. Sensor network optimization using a genetic algorithm. *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, 2003.

Salehpour, A., Afzali-Kusha, A. e Mohammadi, S. Efficient clustering of wireless sensor networks based on memetic algorithm. *[ed.] University of Tehran*, 2008.

Singh, S. K., Singh, M. P. e Singh, D. K. Routing protocols in wireless sensor networks ? a survey. *International Journal of Computer Science & Engineering Survey*, volume 1, 2010.

Tillet, J., Rao, R. e Sahin, F. Cluster-head identification in ad hoc sensor networks using particle swarm optimization. *IEEE International Conference on Personal Wireless Communications*, p. 201–205, 2002.

Yi, S., Heo, J., Cho, Y. e Hong, J. Peach: power-efficient and adaptive clustering hierarchy protocol for wireless sensor networks. *Computer Communications,vol. 30*, p. 2842–2852, 2007.

Younis, M., Youssef, M. e Arisha, K. Energy-aware management for cluster-based sensor networks. *Computer Networks*, volume 43, p. 648–668, 2003.