

LAERCIO MARTINS CARPES

AD*: ALGORITMO DE ROTEAMENTO EM
REDES DE SENSORES BASEADO EM
INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Curitiba PR
Dezembro de 2007

LAERCIO MARTINS CARPES

AD*: ALGORITMO DE ROTEAMENTO EM
REDES DE SENSORES BASEADO EM
INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Área de concentração: *Ciência da Computação*

Orientador: Fabrício Enembreck

Curitiba PR

Dezembro de 2007

C297a
2007

Carpes, Laercio Martins
AD* : algoritmo de roteamento em redes de sensores baseado em
inteligência artificial distribuída / Laércio Martins Carpes ; orientador,
Fabrício Enembreck. – 2007.
xiii, 78 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná,
Curitiba, 2007
Bibliografia: f. 71-78

1. Roteadores (Redes de computação). 2. Detectores. 3. Inteligência
artificial. I. Enembreck, Fabrício. II. Pontifícia Universidade Católica do
Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004.62
006.3



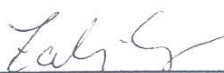
Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 20/2007


Aos 13 dias do mês de dezembro de 2007 realizou-se a sessão pública de defesa da dissertação “AD*: **Algoritmo de Roteamento em Redes de Sensores Baseado em Inteligência Artificial Distribuída**”, apresentada pelo aluno **Laercio Martins Carpes** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Fabrício Enembreck
PUCPR (Orientador)


_____ assinatura


APROVADO
(aprov/reprov.)

Prof. Dr. Marcelo Eduardo Pellenz
PUCPR



APROVADO

Prof. Dr. Jean Marcelo Simão
UTFPR



APROVADO

Conforme as normas regimentais do PPGIA e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.


Prof. Dr. Alessandro Lameiras Koerich
Diretor do PPGIA PUCPR

Dedico este trabalho à minha esposa Liliane, aos meus pais João e Carlinda e aos meus irmãos Emerson, Juarez e Lairton.

Agradecimentos

A Fabrício Enembreck, pelo seu tempo dedicado em ser o orientador para este trabalho, principalmente pela paciência e incentivo diante de minhas limitações.

A Liliane Carpes, pela compreensão e paciência, quando de minha ausência em virtude dos estudos.

A Marlon de Souza Lopes, pelas oportunidades que me permitiram caminhar rumo a mais esse objetivo.

A Bráulio Coelho Ávila, pelo conhecimento transmitido em Engenharia de Conhecimento.

A Edson Emílio Scalabrin, pela dedicação nas aulas ministradas e ajuda prestada.

A Alessandro Lameiras Koerich, pela motivação dada para que eu fizesse o curso de Mestrado.

Sumário

Lista de Figuras	viii
Lista de Tabelas	ix
Lista de Algoritmos	x
Lista de Abreviações	xi
Resumo	xii
Abstract	xiii
1 Introdução	1
1.1 Problema	2
1.2 Objetivo	3
1.3 Motivação	4
1.4 Hipótese	4
1.5 Organização do Trabalho	5
2 Inteligência Artificial Distribuída	6
2.1 Considerações Iniciais	6
2.2 Princípios de Inteligência Artificial Distribuída	6
2.3 Agentes Inteligentes	8
2.3.1 Agentes versus Objetos	9
2.3.2 Tipos de Comportamento	9
2.4 Sistemas Multi-Agentes	11
2.4.1 Interação em SMA	11
2.5 Resolução Distribuída de Problemas	13

2.6	Considerações Finais	19
3	Redes de Sensores	20
3.1	Considerações Iniciais	20
3.2	Definições	20
3.3	Tipos de redes	24
3.4	Tipo de rede escolhido	26
3.5	Arquitetura de Comunicação	27
3.5.1	Camada de Aplicação	27
3.5.2	Camada de Transporte	28
3.5.3	Camada de Rede	28
3.5.4	Camada de Enlace	29
3.5.5	Camada Física	29
3.6	O problema do roteamento	29
3.7	Algoritmos de roteamento	32
3.8	Redes de Sensores, RDP e SMA	36
3.9	Considerações Finais	39
4	Algoritmo AD*	40
4.1	Considerações Iniciais	40
4.2	Princípios do AD*	40
4.3	Modelo de Consumo de Energia	43
4.4	Funcionamento do Algoritmo	44
4.5	Arquitetura do Nó Sensor	49
4.6	Considerações Finais	50
5	Experimentos	51
5.1	Considerações Iniciais	51
5.2	Metodologia de Avaliação	51
5.3	Resultados	54
5.4	Considerações Finais	66
6	Conclusões	68

Lista de Figuras

2.1	Mapa rodoviário simplificado da Romênia [60]	17
3.1	Componentes de um nó sensor [1]	21
3.2	Mica Motes - MICA2 [67]	21
3.3	Rockwell: WINS	22
3.4	SensorWeb	22
3.5	Pilha de protocolo [1]	28
3.6	Agentes antes de formarem o time	38
3.7	Agentes após formarem um time	39
4.1	Estrutura da Mensagem	42
4.2	Rede de sensores	44
4.3	Rede de sensores	47
4.4	Rede de sensores	48
4.5	Modelo do Sensor	50
5.1	Rede de sensores com 1000 nós	54
5.2	Rede de sensores utilizando o MAXEnergy para roteamento	56
5.3	Média de Energia Restante	60
5.4	Média de Energia Consumida	61
5.5	Sensor com Menor Energia	61
5.6	Média de Mensagens Trocadas Entre os Nós	62
5.7	Média de Nós que Compoem a Rota	63
5.8	Desvio Padrão da Energia Restante	63
5.9	Resultado do Teste de Friedman	66

Lista de Tabelas

2.1	Distância em linha reta até Bucharest (estimativa heurística)	18
5.1	Dados dos Experimentos com o AD*(N->1)	57
5.2	Dados dos Experimentos com o AD*(N->2)	58
5.3	Dados dos Experimentos com o GBRE	58
5.4	Dados dos Experimentos com o GBRMaxEnergy	59
5.5	Número de Rotas até a queda da rede (NumRoute)	64
5.6	Aplicação do Teste de Friedman (Medidas Originais)	65
5.7	Aplicação do Teste de Friedman (Medidas Classificadas)	65

Lista de Algoritmos

1	Pseudo-código do A*	16
2	Pseudo-código do AD* (Parte I)	45
3	Pseudo-código do AD* (Parte II)	46
4	Pseudo-código para simular tráfego de dados	55

Lista de Abreviações

RSSF	Redes de Sensores sem Fio
SMA	Sistemas Multi-Agentes
MAS	Multi-Agent Systems
RDP	Resolução Distribuída de Problemas
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
EB	Estação Base
RS	Redes de Sensores
IP	Internet Protocol
WSN	Wireless Sensor Network

Resumo

As inovações tecnológicas introduzidas pelo avanço nos sistemas microeletrônicos e as comunicações sem-fio proporcionaram às redes de sensores sem-fio (RSSF) uma ampla variedade de aplicações comerciais e militares. Essas RSSF possuem de dezenas à milhares de nós-sensores sujeitos a falha com enlace sem-fio e recursos limitados (e.g. energia, processamento e banda). Eles são geralmente depositados em lugares de difícil acesso, sendo assim necessário utilizar algoritmos que possibilitem a descoberta de rotas que gerenciem os limitados recursos disponíveis. Além disso, é necessário garantir um longo tempo de vida para rede. Em RSSF, a rota é estabelecida pela interação entre os sensores, não havendo um controle centralizado. Isto nos motivou a propor um algoritmo baseado em Inteligência Artificial Distribuída chamado AD*. Este tem como objetivo estabelecer o melhor caminho (rota) de um sensor origem até um destino passando por sensores intermediários. A estratégia considera a distância entre cada sensor e o consumo de energia. O AD* foi comparado com outros algoritmos conhecidos da literatura, verificando excelentes resultados para balancear o consumo de energia na rede durante o estabelecimento de rotas.

Palavras-chave: Sistemas Multi-Agente, Redes de Sensores, Inteligência Artificial Distribuída.

Abstract

The technological innovations introduced by the progress in the micro-electronics systems and the wireless communications provided the Wireless Sensor Networks (WSN), a wide variety of commercial and military applications. Those WSN possess of dozens to thousands of sensor-node subjects the fail with connection wireless and limited resources (e.g. energy, processing and band). They are usually deposited in places of difficult access, being therefore necessary to use algorithms that make possible the discovery of routes, still managing the limited available resources. Besides, it is necessary to guarantee a long time of life for net. In WSN the route is established by the interaction among the sensor ones, not possessing a centralized control. That motivated us to propose an algorithm based on Distributed Artificial Intelligence called AD*, tends as objective establishes the best path (route) of a origin sensor to a destiny one going by middlemen sensor. The strategy considers the distance between each sensor one and the consumption of energy. AD* was compared with other known algorithms of the literature, verifying excellent results to balance the consumption of energy in the net during the establishment of routes.

Keywords: Multi-Agent Systems, Sensor Networks, Distributed Artificial Intelligence.

Capítulo 1

Introdução

As redes de sensores sem fio (RSSF) têm sido alvo de pesquisas já faz algum tempo, principalmente devido às inovações tecnológicas introduzidas pelo avanço nos sistemas microeletrônicos e as comunicações sem-fio. O princípio de uma rede de sensores sem-fio é o uso de uma quantidade grande de nós-sensores sujeitos a falha com enlace sem-fio entre eles. Esses nós, também chamados de sensores, devem ser componentes de baixo custo, baixo consumo e pequenos no tamanho.

Os sensores de uma RSSFs são dispositivos autônomos com baixa capacidade de processamento, comunicação e sensoriamento. Entre as características destas redes destacam-se o grande número de sensores e o ambiente em que estes sensores estão depositados que geralmente é de difícil acesso. Como os sensores são componentes com limitações em seu poder computacional, energia, processamento e comunicação, se faz necessário um uso extremamente otimizado dos recursos desta rede. Caso contrário, seu tempo de vida útil será extremamente baixo. Isso implica em reduzir a quantidade de informações transmitidas por ela.

Em ambientes inteligentes futuros, as redes de sensores sem fio serão importantes em detectar, coletar e disseminar informações. Aplicações de sensores representam um novo paradigma para operação de rede, proporcionando objetivos diferentes das redes sem fio tradicionais. As RSSF permitem prover a cobertura de uma região através da união de sensores espalhados, possibilitam tolerância a falha através de alto nível de redundância e é indicada para regiões onde a infraestrutura para o recarregamento de energia dos sensores não esteja disponível. Podem ser utilizados em território inimigo para detonar minas ou avisar da presença de inimigos, regiões vulcânicas, etc. Além disso, este tipo de rede também pode ser empregado adequadamente onde não é possível o uso de redes a cabo.

As RSSF são utilizadas para coletar, detectar, estimar eventos de interesse e/ou executar determinada ação em um sensor remoto, além de prover mecanismos de comunicação.

As RSSF diferem das redes tradicionais pela grande quantidade e densidade de nós sensores que a compõe, nós sujeitos a falha, topologia dinâmica, comunicação *broadcast*, e principalmente a capacidade limitada de energia destes sensores. Uma rede de sensores é autônoma e exige cooperação entre os nós para executar as tarefas para quais foram criadas. A maioria dos problemas relacionados com RSSFs está relacionada com a limitação de energia da rede e a topologia dinâmica da rede durante seu tempo de vida e as alterações que os nós sensores sofrem durante o passar do tempo diminuindo sua capacidade de transmissão e sensoriamento.

1.1 Problema

Quando estamos tratando de RSSF muitas são as dificuldades envolvidas pelo fato dos sensores possuírem recursos limitados. A separação espacial entre os sensores, o volume de informações que trafega entre eles, a presença de obstáculos e a falha de sensores são problemas que devem ser considerados neste tipo de rede. Desta forma, o consumo de energia da rede torna-se um aspecto crítico, pois o fim da energia de determinado sensor, provocará falha de transmissão. Em muitos casos a grande quantidade de sensores na rede ou até mesmo sua disposição torna impraticável a correção de falhas desta natureza, diminuindo drasticamente o tempo de vida dela.

O tempo de vida de um sensor depende da quantidade de energia disponível. Portanto, esta é uma característica que definitivamente deve ser considerada no desenvolvimento de qualquer tecnologia que pretenda ser empregada em RSSF. Sem um gerenciamento eficiente dos recursos de uma RSSF, o tempo de vida deste modelo de rede pode se degradar rapidamente.

O problema que este trabalho pretende abordar consiste na determinação da melhor trajetória (melhor caminho) do sensor origem até o sensor destino passando por sensores intermediários. As dificuldades envolvidas na busca de soluções para este problema devem-se, em grande parte, às alterações das condições de tráfego podendo ocasionar um congestionamento e a queda de determinado sensor, fazendo com que seja necessário refazer o roteamento em determinados momentos, de forma a equilibrar o consumo de energia.

1.2 Objetivo

Nossa proposta é um criar um novo algoritmo de roteamento para estabelecer uma rota para o tráfego de informações de uma origem até seu destino passando por nós intermediários, porém, com um gerenciamento de energia de forma a balancear o consumo da rede.

Para resolver o problema de roteamento em RSSF existem inúmeras técnicas já pesquisadas e outras que estão sendo desenvolvidas. Uma destas técnicas chamada SMECN [2] (Small Minimum Energy Communication Network) cria um sub-grafo da rede de sensores contendo uma rota com o mínimo gasto de energia. Já na técnica de inundação [2] (Flooding) o sensor envia os dados para todos seus vizinhos independente destes dados já terem sido enviados anteriormente para eles. No Gossiping [62] o sensor envia os dados para um vizinho escolhido aleatoriamente, enquanto que no SPIN [2] (Sensor Protocols for Information via Negotiation) os dados só são enviados para aqueles sensores que tem algum interesse na informação. No LEACH [2] (Low-energy adaptive clustering hierarchy) os sensores formam agrupamentos para evitar o desperdício de energia com informações redundantes, onde os nós enviam seus dados para o sensor líder de seu agrupamento. No Difusão direta [2] [36] (Directed Diffusion) são criados gradientes para combinar os dados em rede evitando redundância, sendo os sensores endereçados pelos dados que monitoram e não por seus endereços na rede.

Os algoritmos citados possuem características semelhantes a técnica que está sendo proposta neste trabalho. Contudo, nem todos trabalham o problema do gerenciamento de energia, mas um protocolo de roteamento que utilizar de maneira mais adequada a quantidade de energia disponível na rede, utilizando-se de rotas formadas pelos nós com mais energia estará sempre procurando o equilíbrio da energia na rede. Contudo, é necessário que além do melhor equilíbrio de energia restante, o protocolo também economize o máximo de energia para o estabelecimento das rotas.

Como em uma RSSF os sensores estão espalhados na rede de forma irregular, e a rota é estabelecida pela interação entre eles, não possuindo um controle centralizado, o roteamento se transforma em um problema de distribuição, o que nos levou a optar por utilizar técnicas de Inteligência Artificial Distribuída para resolvê-lo.

Ao olhar para uma rede de sensores estamos tratando, na realidade, de um grafo onde os sensores são os nós do grafo e as arestas representam o custo de comunicação entre eles. Nosso objetivo é estabelecer um caminho (rota) de um sensor origem até um sensor destino passando

por sensores intermediários, considerando uma heurística para balancear o consumo de energia na rede prolongando o tempo de vida da mesma e, portanto, reduzindo o custo de comunicação.

1.3 Motivação

Algoritmos de busca em grafos têm sido amplamente explorados em Inteligência Artificial para a solução de problemas complexos. Conforme citação anterior, as RSSFs podem ser vistas da perspectiva de um grafo, onde poderíamos aplicar um algoritmo de busca para encontrar o melhor caminho para a informação trafegar de um elemento até outro dentro deste grafo. Nossa motivação é que podemos fazer uso de uma técnica de busca eficiente em grafos para efetuar o roteamento em RSSF. Para isto, devemos considerar algumas das características específicas encontradas nestas redes, tais como consumo de energia e distância, visando o estabelecimento da melhor rota com base em uma heurística que deve ser estabelecida.

Sensores podem ser usados para sensoriamento contínuo ou apenas detecção de um evento, além da possibilidade de acionarem atuadores locais. Muitas são as aplicações [1] possíveis com este tipo de rede, tais como, aplicações médicas no monitoramento de pacientes, ambientais para detecção de incêndio/enchentes, na agricultura para monitorar umidade/temperatura, aplicações domésticas criando ambientes inteligentes, aplicações militares para controle, comunicação, inteligência, vigilância e/ou sistema de mira para munição inteligente, aplicações comerciais como gerência de inventários, monitoramento da qualidade de produtos, brinquedos interativos, detecção de incêndio em florestas, pesquisa meteorológica, monitoramento de níveis de poluição, etc.

1.4 Hipótese

Acreditamos que técnicas de Inteligência Artificial (IA) e Inteligência Artificial Distribuída (IAD) podem contribuir para o desenvolvimento de algoritmos eficientes de roteamento em RSSF, com base nos resultados satisfatórios que a IA tem alcançado com técnicas baseadas em busca em grafos.

Em uma RSSF os sensores estão espalhados na rede de forma irregular, e a rota é estabelecida pela interação entre eles, não possuindo um controle centralizado. Portanto, é possível classificar o ambiente de uma rede de sensores de diferentes formas:

- dinâmico: múltiplos eventos ocorrem simultaneamente e não podem ser controlados;

- distribuído: sensores trabalham com informação local e apesar disso, devem colaborativamente encontrar soluções globalmente coerentes;
- incerto: ações são tomadas a partir de informações incompletas, imprecisas devido a situações de falhas inesperadas;
- contínuo: eventos e alterações no ambiente ocorrem em tempo real, exigindo um mecanismo de controle constante.

Nos anos 70, aplicações com essas características levaram pesquisadores como Smith [63], Hewitt [28] [29] e Lesser [43] a rever conceitos de IA e introduzir mecanismos para tratar dessas questões, dando origem à Inteligência Artificial Distribuída (IAD). É possível observar que técnicas de IAD parecem adequadas para a solução de diversas questões em redes de sensores e neste trabalho será explorado esta idéia utilizando uma técnica de IAD no problema de roteamento em RSSF.

Para resolver o problema de roteamento, utilizaremos um algoritmo de resolução distribuída de problema capaz de realizar uma busca heurística em um espaço distribuído de soluções. Além de considerar a distância do caminho percorrido e a distância estimada entre estados a técnica também utiliza a energia consumida do sensor vizinho e o consumo de energia estimado do vizinho até o destino para encontrar a melhor rota entre dois sensores.

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O capítulo 2 descreve princípios de Inteligência Artificial Distribuída, tratando de conceitos como Sistemas Multiagentes, Resolução Distribuída de Problemas e o algoritmo A*. O capítulo 3 discute princípios de redes de sensores, apresentando definições, tipos de rede e o tipo de rede que foi escolhido para desenvolver este trabalho, descrevendo também alguns algoritmos de roteamento em redes de sensores. O capítulo 4 apresenta o algoritmo AD* desenvolvido neste trabalho, descrevendo seu funcionamento e suas principais características. Em seguida, no capítulo 5 apresentamos os experimentos, descrevendo a metodologia de avaliação, os algoritmos com os quais o AD* foi comparado e discutimos os resultados encontrados. O capítulo 6 finaliza o trabalho com as conclusões resultantes e aponta algumas das possibilidades para trabalhos futuros.

Capítulo 2

Inteligência Artificial Distribuída

2.1 Considerações Iniciais

A Inteligência Artificial Distribuída (IAD) é atualmente uma área de pesquisa bastante estudada. Neste capítulo discutimos alguns dos conceitos de IAD mais relevantes para este trabalho abordando princípios de Inteligência Artificial (IA) clássica e abordagens recentes baseadas naquela, tais como Sistemas Multi-Agente (MAS) e Resolução Distribuída de Problemas (RDP).

2.2 Princípios de Inteligência Artificial Distribuída

Pesquisadores de diferentes áreas de pesquisa (e.g. Psicologia, Matemática, Matemática Computacional, Teoria dos Grafos, Pesquisa Operacional etc) iniciaram após a Segunda Guerra Mundial um esforço conjunto com o objetivo de estabelecer relações entre essas diversas áreas e desenvolver sistemas computacionais capazes de resolver problemas complexos. No ano de 1956 o termo "Inteligência Artificial" (IA) foi definido como o mais adequado para representar as aspirações de diversos cientistas que participaram do encontro de Dartmouth. Entre os mais conhecidos, participaram do evento Allen Newell, Herbert Simon, Marvin Minsky, Oliver Selfridge e John McCarthy. Desta forma, busca-se ainda com a IA compreensão e o desenvolvimento de entidades "inteligentes" [60].

De um modo geral, é possível dizer que IA é a capacidade que uma máquina possui para "imitar" o comportamento humano e desempenhar funções como "aprendizagem", "adaptação",

"raciocínio" e "poder de decisão". A IA possui uma variedade de definições [60] [9]. Algumas delas são:

"O estudo da computação que torna possível perceber, raciocinar e agir." (Winston, 1992, [72])

"IA é a parte da ciência da computação voltada para o desenvolvimento de sistemas de computadores inteligentes." (FEIGENBAUM, 1981, [4])

"Inteligência artificial é o estudo das idéias que permitem aos computadores serem inteligentes." (WINSTON, 1984, [71])

"Inteligência Artificial é o estudo das faculdades mentais através do uso de modelos computacionais." (CHARNIAK e McDERMOTT, 1985, [10])

"A arte de criar máquinas que executam funções que requerem inteligência quando executadas por pessoas." (KURZWEIL, 1990, [40])

"Inteligência artificial é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor." (RICH, Elaine e KNIGHT, Kevin, 1993, [58])

Para Winston [71] a IAD tem como alvo construir sistemas inteligentes e entender a inteligência humana. Mas o que é inteligência? É difícil definir com exatidão o que é inteligência, mas a idéia chave de inteligência dada por Brooks [9] é de que:

Inteligência é determinada pela dinâmica de interações com o mundo.

Portanto, inteligência não pode ser obtida com uma simples entidade, pois a inteligência surge pela interação entre várias entidades [38]. Isto é, a inteligência só poderá ser determinada pelo comportamento do sistema em relação ao seu ambiente. Portanto para Brooks [9] a inteligência está nos olhos do observador.

Pesquisas de IA em diversos campos de estudo contribuíram para a fundamentação dos princípios teóricos dela, os quais emergiram com duas abordagens:

- Abordagem Cognitiva: a ênfase está nos processos racionais e intelectuais. O alvo é encontrar explicações para comportamentos inteligentes baseando-se em aspectos psicológicos e processos algorítmicos. Nesta abordagem as primeiras modelagens foram baseadas em regras de produção e a lógica de predicados.

- **Abordagem Conexionista:** o foco está no modelo de funcionamento do cérebro, dos neurônios e das conexões neurais. Nesta abordagem temos os modelos de redes neurais artificiais.

Desta forma, a Inteligência Artificial Distribuída (IAD), com um enfoque diferente da Inteligência Artificial clássica, tenta dividir um problema complexo em problemas menores e mais simples que podem ser resolvidos individualmente e em paralelo. Os sistemas desenvolvidos utilizando a técnica da IAD, são indicados para resolver problemas complexos e que possuem algum nível de distribuição, como uma rede de transportes em que existem diversos elementos. Neste caso os elementos de transporte precisam se coordenar para garantir que objetivos coletivos sejam alcançados, como economia de recursos e pontualidade na realização de tarefas. É mais intuitivo analisar este problema de forma distribuída, onde os elementos possuem autonomia para definir algumas rotas, pois as condições de tráfego estão em constante adaptação [21].

Alguns autores destacam que um dos principais aspectos da IAD é a possibilidade de se criar mecanismos de busca distribuída [43]. Nesta abordagem, o espaço de busca é particionado e múltiplos resolvedores podem simultaneamente executar buscas locais em diferentes partes e posteriormente compartilhar o resultado destas buscas locais. Estes resolvedores são processos autônomos, também chamados de agentes, que através de processamento local e comunicação interprocessos são capazes de alcançar em conjunto uma inteligência global.

Na seção seguinte são discutidos alguns conceitos relacionados a agentes inteligentes, e em seguida descrevemos as duas principais áreas em que a IAD está dividida: Sistemas Multi-Agente (SMA) e Resolução Distribuída de Problemas (RDP).

2.3 Agentes Inteligentes

Muitos dos conceitos relacionados a agentes de software não tem uma definição aceita universalmente dentro da Inteligência Artificial. Entretanto, algumas delas são mais frequentemente adotadas em diferentes trabalhos. Algumas dessas definições são:

Agente é um sistema de computador capaz de receber informações sensoriais do ambiente e executar ações que alteram este ambiente de maneira autônoma sem a intervenção humana, tornando o sistema flexível e independente [38].

Em outra definição, agente é considerado uma entidade física ou virtual [24] capaz de agir no ambiente, comunicar diretamente com outros agentes e perceber o ambiente. Ele também possui habilidades e pode ofertar serviços, possui somente uma representação parcial do ambiente e possui comportamentos que o aproximam da satisfação de seus objetivos.

Já para Maes [47], os agentes autônomos são sistemas computacionais, os quais inseridos em um ambiente dinâmico e complexo, percebem e atuam automaticamente neste ambiente, e fazendo-o, compreendem um conjunto de objetivos ou tarefas para as quais foram projetados.

2.3.1 Agentes versus Objetos

As definições anteriores não são suficientes para evitar a confusão entre o conceito de agente e objeto, principalmente para quem é especialista em Engenharia de Software ou Orientação a Objetos. Mas apesar de algumas similaridades entre eles, existem diferenças significativas [38]:

- Grau de autonomia que agentes e objetos possuem: um objeto invoca um método diretamente em outro objeto, sendo dele próprio a autonomia para executá-lo e não de quem recebeu a invocação. Já com agentes a comunicação é feita por meio de requisição, ficando a decisão de atender ou não determinada requisição a cargo do agente que recebeu esta solicitação;
- Comportamento autônomo flexível: para construir sistemas com objetos o que se faz é integrar os comportamentos predefinidos nestes objetos. Por outro lado, agentes possuem um comportamento flexível (e.g. reativo, pro-ativo e social), podendo adquirir novos comportamentos ao longo de sua existência;
- Controle interno: cada agente é modelado para ter seu próprio controle de tarefa. Já no modelo de objetos, geralmente existe um controle único de thread para todo o sistema.

Dentro da Inteligência Artificial Distribuída os agentes são entidades computacionais, dotadas de capacidades cognitivas ("percepção", "raciocínio" e "memória") ou reativas, agindo ou reagindo em função dos objetivos para as quais foram concebidos.

2.3.2 Tipos de Comportamento

Geralmente, costuma-se classificar o comportamento de um agente em cognitivo ou reativo, embora seja mais comum o desenvolvimento de agentes híbridos. O comportamento

cognitivo é orientado pelos objetivos, crenças, desejos e intenções, sendo a Arquitetura BDI [56] o exemplo mais conhecido. O comportamento reativo é orientado pela função de sobrevivência, sendo o agentes de Brooks [9] [8] um exemplo típico desta categoria.

Quando dizemos que um agente é capaz de ação autônoma flexível [64], estamos dizendo também que ele pode ser:

- reativo: o agente interage com o ambiente respondendo de maneira oportuna (pontual) às alterações que ocorrem neste ambiente. Dessa forma o agente reage mediante estímulo do ambiente.
- proativo: gerando e tentando alcançar objetivos, isto é, o agente não simplesmente age em respostas a eventos ocorridos no ambiente, como toma a iniciativa quando reconhecer alguma oportunidade (quando for apropriado).
- social: é a habilidade do agente de interagir com outros agentes e humanos por meio de algum tipo de linguagem de comunicação entendível por todos para completar a resolução de seu problema e/ou cooperar com os outros.

Naturalmente, certos agentes terão características adicionais, e para cada tipo de aplicação, algumas destas características serão mais importantes que para outras. Sendo elas: reatividade, adaptabilidade, mobilidade, personalidade, interatividade, diferentes formas de percepção do ambiente de atuação, comunicabilidade.

Alguns modelos de comportamento de agentes são baseados no princípio do raciocínio prático, inspirado na teoria de raciocínio prático de humanos. O modelo de comportamento mais conhecido de raciocínio prático é baseado nos estados mentais de um agente chamado modelo crença-desejo-intenção (BDI - beliefs, desires, intentions). Neste modelo as crenças correspondem às informações que se tem sobre o ambiente, sobre si mesmo e sobre os outros agentes. Os desejos representam os estados que o agente quer alcançar, indicando as preferências do agente sobre os estados do ambiente. As intenções representam o comprometimento do agente em realizar ações no sentido de seu objetivo.

Na arquitetura reativa, o agente busca no ambiente somente o que é válido para si próprio, sem construir uma representação artificial do ambiente. Nesta arquitetura o agente é situado, sendo suas ações diretamente influenciadas por um conjunto de eventos do ambiente. O mais famoso tipo de arquitetura reativa é a arquitetura de Brooks [8] [9].

2.4 Sistemas Multi-Agentes

A abordagem conhecida como "Sistemas Multi-Agente (SMA)" é utilizada no desenvolvimento de sistemas distribuídos complexos. A abstração chave utilizada é aquela dos agentes de software, que podem interagir com outros agentes gerando os SMA. Essa abordagem pode reduzir o custo e a complexidade associados ao desenvolvimento de sistemas distribuídos [38], além de permitir uma maior flexibilidade no processo de descoberta de soluções de problemas onde essas não podem ser previamente estabelecidas.

Para Sycara [65] e Corkill [12], a motivação pelo aumento de interesse por mais pesquisas em SMA se deve por seis razões: i) por prover soluções para problemas que são amplamente complexos para um controle centralizado; ii) por permitir a interconexão e interoperação com sistemas legados já existentes; iii) por prover soluções para problemas que podem conter uma sociedade de componentes autônomos interagindo; iv) por prover soluções que usam componentes que estão espacialmente distribuídos, e.g. redes de sensores; v) por prover soluções onde a habilidade está distribuída; vi) por aumentar a performance em função da computação concorrente que é explorada, permitindo maior confiabilidade, extensibilidade, robustez, manutenibilidade e flexibilidade.

Em um SMA, agentes de diversos tipos e diferentes competências podem co-existir, sendo necessário o estabelecimento de uma certa estrutura organizacional entre eles. O tema organização, central ao projeto de sistemas multi-agente, é definido como a identificação dos inter-relacionamentos existentes entre seus componentes básicos: o agente, o ambiente e a tarefa.

2.4.1 Interação em SMA

Um sistema multi-agente é composto por múltiplos componentes autônomos (ou semi-autônomos) chamados de agentes. Cada agente deve ser criado com a capacidade de realizar uma ou mais tarefas e de se comunicar com outros agentes. Sistemas multi-agente são escolhidos preferencialmente para representação de problemas que tenham múltiplos métodos de resolução, múltiplas perspectivas e/ou múltiplas entidades para resolução do problema. Os tipos de interações mais comuns entre os agentes são: cooperação (trabalhando juntos em direção a um alvo em comum), coordenação (organizando as atividades para que interações prejudiciais sejam evitadas), e negociação (chegando a um acordo no qual seja aceitável para todas as partes envolvidas).

A interação entre essas entidades é caracterizada pela relação dinâmica ocorrida por vários agentes, seja diretamente, por intermédio de outros agentes, ou do ambiente que eles estão compartilhando, sendo fundamental para a descoberta da solução global de um problema distribuído.

Uma interação pode ser simplesmente caracterizada por uma "marca" deixada por um agente no ambiente ou por trocas de mensagens de alto nível como, por exemplo, mensagens estruturadas de acordo com uma ACL (Agent Communication Language) proposta pela FIPA (Foundation for Intelligent Physical Agents).

Algumas das características de SMA conforme citadas por [38], [24] e [65] são diretamente dependentes de um modelo de interação adequado. Dentre essas características podemos citar:

- visão local: um agente não possui capacidade para a resolução completa de um problema;
- descentralização de interação: o controle está logicamente distribuído;
- descentralização de informação: os dados são descentralizados;
- tempo real: computação é assíncrona;
- desconhecimento da solução: a solução (a inteligência) está na interação entre os agentes e não pode ser previamente codificada em virtude da complexidade do ambiente e dos eventos imprevisíveis.

Uma das principais vantagens dos sistemas multi-agente em relação a outras técnicas de concepção de sistemas está na capacidade de adaptação a novas situações, tanto pela eliminação de agentes e/ou inclusão de novos agentes, quanto pelo proveito que tais sistemas tiram da capacidade de paralelismo e distribuição das arquiteturas de computadores. Tais características consideram também a autonomia entre os agentes de software, permitindo-se que sistemas muito complexos possam ser implementados por meio de agentes simples. Em um SMA o foco está na estruturação dos agentes, bem como, em desenvolver arquiteturas de agentes que interagem de forma autônoma, onde a solução do problema surge por meio da interação entre estes agentes. Em SMA o problema é considerado a partir de uma perspectiva ascendente (*bottom-up*), isto é, o agente é criado antes de existir o problema.

O modelo de interação mais conhecido utilizado em Sistemas Multi-Agente cognitivos é aquele baseado no quadro negro (*blackboard*) [21]. No modelo *blackboard* os agentes não

comunicam qualquer dado diretamente, mas compartilham estes dados indiretamente por meio de *blackboard*. Isto é, um agente posta os dados no *blackboard* para que os outros agentes tenham acesso a estes dados quando necessário.

A arquitetura *blackboard* pode conter vários supervisores organizados em níveis hierárquicos, onde cada um desses supervisores dirige um grupo de agentes [6]. Portanto, agentes especialistas são organizados em grupos que são coordenados por um grupo de supervisores.

Este modelo torna-se proibitivo quando o tempo de resposta é um fator crítico para o sistema [21]. Um SMA também deve possuir mecanismos de coordenação para que os agentes alcancem seus objetivos que muitas vezes são complementares ou cooperativos, aumentando deste modo a performance coletiva [18] [38]. Em alguns casos a escolha do mecanismo de coordenação é efetuada em tempo de execução pelos agentes que precisam se coordenar. As principais motivações para a implementação de um mecanismo de coordenação são [21]:

- Informações Incompletas: os agentes necessitam de informações e resultados que só os outros agentes possuem;
- Recursos do ambiente são limitados: há necessidade de cooperação para otimizar o uso destes recursos;
- Recursos do agente são limitados: otimizar custo fazendo o possível para evitar ações redundantes;
- Permitir que os agentes tenham separado mas objetivos interdependentes para encontrar os objetivos deles.

2.5 Resolução Distribuída de Problemas

Para problemas onde o espaço de possibilidades é muito grande, técnicas tradicionais de resolução de problemas são ineficientes. Esses problemas são geralmente NP-Completo ou NP-Hard. Problemas clássicos de IA como o caixeiro viajante, a torre de Hanói, entre outros, geram espaços de estados proibitivos para técnicas convencionais de busca que produzem resultados satisfatórios apenas para pequenas instâncias do problema. Por outro lado, abordagens baseadas em Resolução Distribuída de Problemas decompõem o problema em problemas

menores que são então delegados a agentes de software. Quando cada agente alcança a solução de seu subproblema, é possível afirmar que uma solução global foi obtida.

Ashri [3], argumenta que existem três paradigmas de resolução distribuída de problemas baseadas em agentes: coordenação (os agentes realizam suas atividades individuais de maneira coerente, ou seja, respeitando os outros agentes), colaboração (os agentes trabalham juntos para atingir um objetivo comum) e a competição (os agentes devem compartilhar recursos para poder sobreviver e atingir objetivos individuais).

Segundo Durfee [15], a resolução distribuída de problemas usando agentes deve ser feita levando em conta os seguintes objetivos:

- acelerar a solução de um problema, privilegiando o trabalho paralelo dos agentes;
- obter várias soluções locais, utilizando as capacidades dos outros agentes para obter uma solução global final;
- melhorar a confiabilidade dos resultados, utilizando o fato que os agentes podem compartilhar os resultados parciais;
- reduzir a possibilidade de duplicação de processamento, atribuindo um subproblema a um número limitado de agentes, por exemplo a um único;
- reduzir o volume de comunicação, trocando apenas as informações necessárias.

Um ponto importante é que para implementar estes modos de cooperação, existe a necessidade de protocolos de comunicação relativamente elaborados. Estes protocolos geralmente utilizam a metáfora da negociação de contrato para propor um protocolo de alto nível, como o Contract-Net [63].

Muitas outras técnicas foram propostas com esse objetivo, como a Eco-resolução [21], a Negociação [19] [48] ou ainda técnicas baseadas em Planejamento ¹ (centralizado, parcial centralizado ou parcial distribuído). No modelo de eco-resolução os agentes têm que atingir um estado de satisfação. Quando um agente não consegue atingir o estado de satisfação porque um outro agente está atrapalhando o seu objetivo, ele o agride. Este segundo agente também pode agredir outros agentes para atingir seus objetivos. Nesse modelo, uma série de restrições é propagada reduzindo o espaço de soluções candidatas. A solução é encontrada quando todos

¹Uma discussão sobre coordenação baseada em Planejamento pode ser encontrada em [21].

os agentes estão satisfeitos. Já a negociação é o processo pelo qual dois ou mais agentes necessitam tomar uma decisão em comum para resolver um conflito. Quando ambos os agentes estão envolvidos em negociações para atingir uma meta comum a negociação é chamada Negociação Cooperativa. Um mecanismo de negociação é composto por dois elementos principais: o protocolo de negociação e uma estratégia de negociação. O protocolo de negociação define o significado por trás de cada interação entre os agentes, o que pode ser feito e quais as seqüências de ações a serem tomadas [20]. A estratégia de negociação é normalmente composta por um conjunto de planos que devem ser seguidos pelo agente para que o mesmo alcance seus objetivos.

Revisão Distribuída de Crenças e Satisfação Distribuída de Restrições são outros formalismos conhecidos para a solução de problemas complexos baseados na integração de soluções parciais. Sistemas de Revisão de crenças permitem a manipulação de conhecimento não-monotônico. Geralmente quando um resolvidor de problema recebe uma nova informação, esta pode gerar conflitos com algumas das fórmulas bem formadas que compõem sua base de conhecimentos. Portanto, um sistema de Revisão de Crenças (também conhecido como Sistema de Manutenção da Verdade) deve, a partir de uma nova informação: (i) identificar os conhecimentos conflitantes e as proposições que invalidam essa informação usando um provador de teoremas; (ii) identificar as proposições que a suportam e (iii) remover as proposições que podem ser derivadas dela. O problema torna-se mais complexo quando existem várias bases de conhecimento que necessitam manter uma consistência lógica local e global. Neste caso, tratamos o problema como Revisão de Crenças Multi-Agente ou Revisão Distribuída de Crenças [35] [49]. Note que essa definição é notadamente diferente do conceito de Revisão de Crenças Distribuídas, onde é possível utilizar um algoritmo de revisão de crenças centralizado como o Backtraking Dirigido por Dependência para unificar as diversas bases de conhecimento. Por outro lado, técnicas de Revisão Distribuída de Crenças, são baseadas em algoritmos de coordenação, comunicação, modelos de reputação e combinação de informação incerta.

Conflitos entre agentes também podem ser resolvidos utilizando técnicas de satisfação de restrições. Uma técnica de satisfação de restrições é utilizada quando se deseja encontrar valores para um conjunto de variáveis que devem satisfazer determinadas restrições. A princípio um conjunto de restrições é fornecido para o sistema que utiliza algum algoritmo de busca para encontrar uma determinada configuração para as variáveis que satisfaça todas as restrições pré-determinadas. Ao longo da busca, novas restrições podem ser derivadas (usando um provador de teoremas) e estados candidatos encontrados. Como o espaço de busca pode ser

bastante grande, técnicas heurísticas de poda e escolha de candidatos podem ser utilizadas. Um problema de Satisfação Distribuída de Restrições consiste na descoberta de uma solução que satisfaz as restrições de um conjunto de agentes. Entretanto, cada agente possui um conjunto privado de restrições (base de conhecimento) sobre uma determinada variável. Diversos algoritmos síncronos e assíncronos foram propostos para solucionar esse problema. Dentre os mais conhecidos podemos citar Synchronous Branch and Bound [32] e Adopt [51].

Algoritmo 1 Pseudo-código do A*

Require: noAlvo

Require: noInicial

lista.Adiciona(*noInicial*, 0) // adiciona o nó inicial na lista e o custo 0

loop

cMC \Leftarrow *selecionaCMC*(*lista*) // CMC = caminho de menor custo

noCorrente \Leftarrow *selecionaUltimoNo*(*cMC*)

if *noCorrente* = *noAlvo* **then**

 QUIT

end if

lista.Apaga(*cMC*) // apaga o caminho de menor custo da lista

for *noCorrente* \Rightarrow *noSucessor* **do**

G = *custo*(*noInicial*, *noSucessor*) // Custo do nó inicial até o nó corrente

H = *heurística*(*noSucessor*, *noAlvo*) // Custo estimado do nó sucessor até o nó alvo

caminho \Leftarrow *cMC* + *noSucessor*

custo \Leftarrow *G* + *H*

lista.Adiciona(*caminho*, *custo*) // adiciona a lista os novos caminhos candidatos e o custo deles

end for

end loop

Neste trabalho pretende-se desenvolver uma técnica baseada em resolução distribuída de problema, que tem como base um algoritmo de busca bastante conhecido em Inteligência Artificial, o algoritmo A*. A seguir descrevemos os conceitos fundamentais deste algoritmo.

O algoritmo A* (Algoritmo 1) busca minimizar o custo total de uma busca, sendo atualmente um dos algoritmos de busca mais conhecidos [60]. Este tipo de busca avalia o custo de cada nó até seu vizinho (conhecido como $g(n)$) e do nó vizinho até o nó objetivo com uma

função heurística (conhecido como $h(n)$). O $g(n)$ é dado pelo custo de um nó inicial até o nó n e $h(n)$ é o custo de n até o nó meta. Dessa forma o A* utiliza como função de qualidade de um nó aquele que minimiza a equação

$$f(n) = g(n) + h(n)$$

onde $f(n)$ é o custo estimado da solução. Este algoritmo é de solução ótima com base na heurística definida.

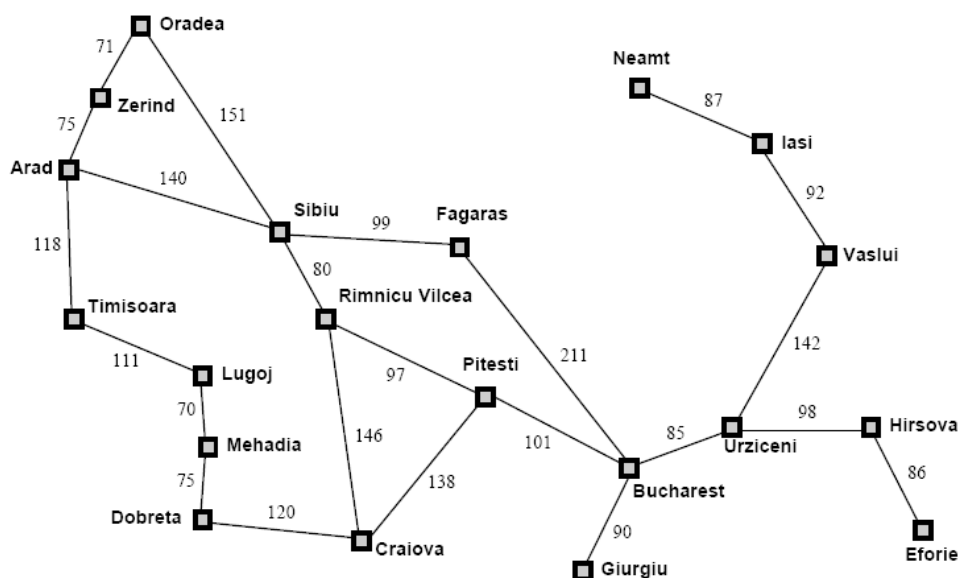


Figura 2.1: Mapa rodoviário simplificado da Romênia [60]

Considerando o mapa rodoviário simplificado da Romênia (Figura 2.1) será exemplificado o funcionamento do algoritmo A* conforme exemplo utilizado em [60]. Imagine alguém na cidade de Arad, na Romênia, aproveitando suas férias visitando os pontos turísticos da cidade e tudo mais, porém ele tem uma passagem não reembonsável para sair de Bucharest no dia seguinte, então ele toma como um objetivo chegar a Bucharest, porém não é de seu conhecimento como chegar a Bucharest o mais rápido possível.

O algoritmo A* para Bucharest é executado utilizando os valores de $g(n)$ com base nos dados da Figura 2.1 e o $h(n)$ na Tabela 2.1. O algoritmo A* expande o Arad exibindo $f(\text{Sibiu})=140 + 253= 393$, $f(\text{Timisoara})=118 + 329= 447$ e $f(\text{Zerind})=75 + 374= 449$. Sibiu é a cidade escolhida pois é a que possui o menor $f(n)$. Agora Sibiu é expandida obtendo as seguintes cidades com seus respectivos $f(n)$: $f(\text{Arad})=280 + 366= 646$, $f(\text{Fagaras})=239 + 178= 417$, $f(\text{Oradea})=291 + 380= 671$, $f(\text{Rimnicu Vilcea})=220 + 193= 413$. Estas cidades se juntam

Tabela 2.1: Distância em linha reta até Bucharest (estimativa heurística)

Cidade	Distância	Cidade	Distância
Arad	366	Bucharest	0
Craiova	160	Dobreta	242
Eforie	161	Fagaras	178
Giurgiu	77	Hirsova	151
Iasi	226	Lugoj	244
Mehadia	241	Neamt	234
Oradea	380	Pitesti	98
Rimnicu Vilcea	193	Sibiu	253
Timisoara	329	Urziceni	80
Vaslui	199	Zerind	374

a Timisoara e Zerind, para a próxima escolha, sendo escolhida a cidade de $f(\text{Rimnicu Vilcea})=413$. Com a expansão de Rimnicu Vilcea as cidades de $f(\text{Craiova})=366 + 160= 526$, $f(\text{Pitesti})=317 + 98= 415$ são adicionadas e $f(\text{Sibiu})=300 + 253= 553$ aparece novamente, porém, com outro valor. Nessa expansão dentre as cidades foi escolhida a cidade de Fagaras por apresentar o menor $f(n)$, ($f(\text{Fagaras})=415$) obtido na expansão anterior. Com a expansão de Fagaras a cidade Sibiu aparece novamente com o valor de $f(\text{Sibiu})=338+253=591$ e a cidade destino Bucharest aparece com o valor de $f(\text{Bucharest})=450 + 0= 450$, porém, dentre as cidades que podem ser escolhidas a que possui o menor valor de $f(n)$ é Pitesti com o valor de $f(\text{Pitesti})=415$. Expandindo Pitesti aparecem novamente as seguintes cidades com os valores $f(\text{Bucharest})=(140+80+97+101) + 0= 418$, $f(\text{Craiova})=(140+80+97+138) + 160= 615$ e $f(\text{Rimnicu Vilcea})=(140+80+97+97) + 193= 607$. Dentre essas cidades presentes agora a próxima escolhida é Bucharest, cujo $f(\text{Bucharest})=418$ é o menor dentre todas, chegando assim ao destino.

Quando o A^* termina sua busca, diz-se por definição que ele encontrou um caminho cuja custo atual é menor que o custo de qualquer outro caminho. Com base nisto, o caminho gerado pelo algoritmo A^* no exemplo acima é utilizado pelo viajante para percorrer o caminho com a menor distância dentre todos os caminhos de Arad até Bucharest.

2.6 Considerações Finais

Neste capítulo foram descritos os principais conceitos sobre Inteligência Artificial Distribuída, tratando da abordagem de Sistemas Multi-Agente e Resolução Distribuída de Problemas. No próximo capítulo introduzimos conceitos sobre redes de sensores, sendo apresentado um estudo sobre trabalhos que utilizam técnicas de IAD para solucionar alguns problemas encontrados neste tipo de rede.

Capítulo 3

Redes de Sensores

3.1 Considerações Iniciais

Anteriormente foi apresentado quais são os objetivos deste trabalho e a motivação para utilizar uma abordagem baseada em Inteligência Artificial Distribuída. Também foi apresentado os conceitos sobre IA, IAD e explicado o funcionamento do algoritmo de busca A*. Neste capítulo são apresentados alguns conceitos, as possíveis aplicações, e os desafios críticos enfrentados quando uma aplicação é desenvolvida com uma RSSF. Além disso estudos que integram IAD e RSSF são discutidos.

3.2 Definições

Sensores em RSSF são pequenos componentes que combinam capacidade de processamento e comunicação sem fio. Cada sensor pode ser visto como uma entidade computacional autônoma [24], que possui pequena capacidade de processamento, memória, e principalmente, energia e banda de comunicação [50]. Estes sensor (ou nós) coletam dados ou atuam no ambiente, processam localmente ou coordenadamente entre vizinhos podendo enviar ou receber a informação da estação base.

Os principais componentes de um nó sensor consiste (Figura 3.1) de uma unidade de sensoriamento, unidade de energia, unidade de processamento e unidade de transmissão [39] [1] [57]. Alguns tipos de sensores possuem algumas características adicionais, tais como, um gerador de energia, sistema de movimentação e sistema de localização.

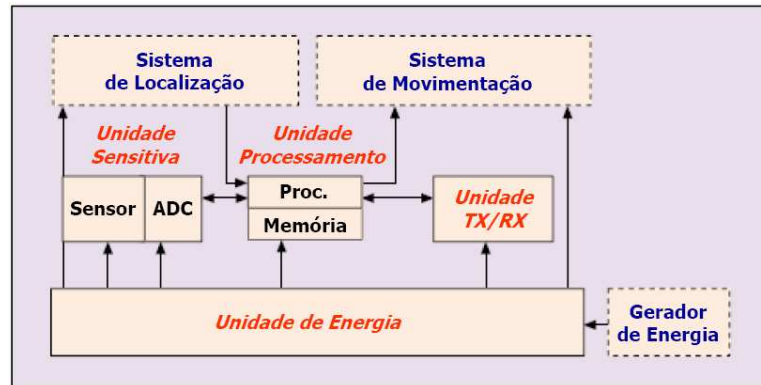


Figura 3.1: Componentes de um nó sensor [1]

A unidade de sensoriamento é composta pelas sub-unidades de sensoriamento e um conversor analógico-digital. Uma vez convertido, o sinal é então entregue para a unidade de processamento, a qual geralmente está associada a uma pequena unidade de armazenamento, gerenciando a colaboração com outros nós em busca do alvo. A unidade de transmissão representa todo o sistema de transmissão e recepção que conecta o nó na rede, podendo ser via infravermelho, radio frequência ou bluetooth. A unidade de energia é o armazenador de energia, que tem capacidade finita. Contudo em alguns casos é possível recarregar a energia quando o armazenador estiver ligado a um gerador de energia (ex.: células solares).



Figura 3.2: Mica Motes - MICA2 [67]

Existem diversos tipos de micro-sensores que apresentam os mais variados tamanhos, programados para um tipo de sensoriamento específico ou vários tipos de sensoriamento em



Figura 3.3: Rockwell: WINS

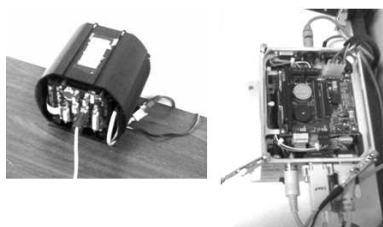


Figura 3.4: SensorWeb

um mesmo sensor. Existem diversos modelos de sensores resultantes de diversos projetos de pesquisa, como por exemplo, o Projeto Mica Motes [31] da Universidade de Berkeley, onde foi desenvolvido três famílias de nós sensores: MICA, MICA2 [30] (Figura 3.2) e o MICA2DOT. Outros nós desenvolvidos são o WINS Rockwell (Figura 3.3) desenvolvido pelo Rockwell Science Center com a Universidade da Califórnia e o SensorWeb (Figura 3.4) desenvolvido pelo Instituto de Tecnologia da Califórnia.

Redes de sensores sem fios (RSSFs) [53] são formadas por dezenas à milhares de sensores depositados em um ambiente para monitorar e transmitir alguma característica física do ambiente. São um conjunto de nós que operam de modo individual, mas que podem formar uma rede com o objetivo transmitir informações locais para uma determinada estação base. A comunicação feita entre os nós vizinhos acontece quando algum tipo de interação é necessária entre os nós e esse tipo de comunicação é chamada de cooperação. As três principais funções destes sensores são: coletar, controlar e atuar.

A estação base serve como meio de comunicação entre a redes de sensores e o usuário final. Como os sensores só conseguem se comunicar com seus vizinhos, não possuindo acesso

direto à estação base, estes vizinhos devem fazer o papel de roteadores para que a mensagem de determinado sensor alcance a base ou vice-versa.

Este tipo de rede provavelmente [27] será muito utilizada devido à grande habilidade para controlar e monitorar ambientes físicos de uma localização remota. Ela permite uma visão multi-dimensional do ambiente e a atuação sobre este ambiente de maneira remota. Cada sensor possui independência entre si, de maneira que o mesmo só possui informação local sobre o ambiente e somente conhece seus vizinhos que estão dentro do alcance de sua faixa limitada de transmissão.

As RS possuem uma ampla variedade de aplicações [1]: comerciais, militares, ambientais, saúde. Podem ser utilizadas para diagnosticar e detectar falhas em máquinas, detectar intrusões (vídeo sensores) [26], em tarefas especializadas como vigilância e segurança, monitoração ambiental, transporte, indústria e cuidados médicos [57], monitorar veículos em movimento, presença de certos tipos de objetos, velocidade/direção/tamanho de um objeto. Na área militar essas redes podem ser utilizadas para comando, controle, comunicação, vigilância, inteligência, reconhecimento, mira inteligente, monitoração de tropas amigas para saber as condições e a disponibilidade de equipamentos e munições no campo de batalha. As RSSFs podem ser aplicadas para reconhecimento e detecção de ataques biológicos e nuclear, onde, os sensores podem ser depositados em uma ampla área para que detectem agentes biológicos ou de radiação comunicando imediatamente, evitando catástrofes. Podem também ser utilizadas após uma explosão nuclear, para reconhecimento da área sem a necessidade de expor soldados a radiação. Em aplicações ambientais podem ser utilizadas para detecção de incêndio em florestas, isto é, os sensores podem avisar de determinado foco de incêndio antes que o mesmo se espalhe. Podem também ser utilizadas para detectar agentes biológicos no ar e na água. Também é possível utilizar na agricultura para observar dados como temperatura, umidade, nível de água no solo, nível de erosão para auxiliar na aplicação de insumos. Na saúde as RSSFs podem ser utilizadas para monitoramento de médicos e pacientes dentro do hospital, coletando informações como temperatura do corpo, pressão do paciente, localização do médico dentro do hospital. Na automação de residências nós sensores poderiam atuar sobre os eletrodomésticos interagindo com uma rede externa via internet ou satélite. No futuro, as redes de sensores se tornarão parte de nosso dia a dia, mais comuns que o computador pessoal.

Geralmente se utiliza este tipo de rede para coletar, disseminar informações em ambientes irregulares de difícil acesso [24]. O controle neste tipo de rede não é centralizado, cada sensor opera de maneira autônoma tomando suas próprias decisões a partir de informação

local, e o fluxo de dados é predominantemente unidirecional do sensor de origem até o sensor alvo. A vantagem é que uma RSSF pode ser utilizada em quase todos os tipos de terreno com um ambiente hostil onde não é possível o uso de redes tradicionais.

Em redes tracionais o consumo de energia não é tão relevante, já que geralmente uma rede elétrica alimenta os nós da rede e baterias podem ser trocadas ou recarregadas conforme a necessidade. Porém devido à dificuldade de acesso o principal objetivo de uma RS passa a ser prolongar o tempo de monitoramento por meio de um eficiente gerenciamento de energia [57]. Portanto, a quantidade de dados que trafega nestas redes deve ser pequena, e as operações realizadas devem ser feitas de maneira eficiente.

A implantação de uma RSSF deve, ainda, levar em consideração diversos fatores, tais como:

- Tolerância a Falha: alguns sensores podem falhar como consequência de falta de energia ou mesmo interferência do ambiente. Contudo é preciso garantir que mesmo com a queda de alguns sensores a rede continue funcionando;
- Escalabilidade: a quantidade de sensores depositados em um ambiente pode variar de dezenas a milhares ou ainda chegar ao extremo de milhões;
- Custo: o custo de uma rede de sensores deve ser menor que o de uma rede tradicional. Portanto, o custo de cada nó deve ser baixo, embora este custo seja variável conforme as funcionalidades que o sensor possua;
- Limitações de Hardware: devido à restrição de memória, processamento e principalmente de energia, o desenvolvimento de RSs confiáveis e eficientes depende da solução de grandes desafios na criação de algoritmos de roteamento, protocolos de rede, softwares e serviços que utilizam estes recursos.

3.3 Tipos de redes

As RSSFs podem ser classificadas por meio de diversas características [52] segundo a sua configuração. Essas características variam quanto a *composição*, *organização* e *distribuição dos sensores* e a *dinâmica da rede*. A composição da rede pode ser homogênea quando todos os sensores possuem as mesmas características e heterogênea quando as características (capacidade) de um sensor em relação ao outro é diferenciada. A organização da rede é considerada

plana quando não existe nenhum agrupamento entre os sensores e hierárquica quando os sensores são organizados em grupos, particionando o ambiente em regiões de interação chamadas setores. Quanto à distribuição, uma rede é considerada regular quando os sensores são estrategicamente distribuídos em uma grade, seja esta distribuição feita manualmente ou por um robô, e irregular quando estes sensores são distribuídos (esparrramados) pelo ambiente de modo aleatório.

Quanto à dinâmica (mobilidade) da rede, ela pode ser classificada de dois modos: redes de sensores estáticas e redes de sensores móveis. As RSSFs estáticas podem ter qualquer uma das características quanto a composição, organização e distribuição dos sensores, porém, os sensores são estáticos e se comunicam sempre com os sensores que são seus vizinhos, não havendo mobilidade entre estes sensores que estão se comunicando. Já as RSSFs móveis também podem ter qualquer uma das características quanto a composição, organização e distribuição, porém, como o próprio nome enfatiza, os sensores se movimentam em relação aos demais sensores e estação base. Neste tipo de rede a qualquer momento a rota construída para atingir determinado objetivo pode falhar, havendo a necessidade de construir uma nova rota.

Quanto à transmissão de dados em uma RSSFs podemos classificá-la como:

- Simplex: Os nós sensores permitem apenas a transmissão de informação;
- Half-Duplex: Os nós permitem transmitir ou receber informações em um determinado instante;
- Full-duplex: Os nós permitem transmitir e receber informações ao mesmo instante.

Outra característica em RSSFs é o endereçamento, cujo propósito é identificar os nós que compõe a rede para auxiliar no estabelecimento de rotas. Em redes tradicionais geralmente se utiliza um endereçamento global, onde é atribuído um endereço IP para cada ponto, sendo que quando é necessário contactar determinado ponto basta conhecer o IP deste. As mensagens que trafegam na rede possuem o endereço de quem esta enviando e para quem está indo, portanto estes endereços ocupam certo espaço na mensagem. As RSSFs geralmente são compostas por uma grande quantidade de pontos (nós), o que ocasiona um maior número de endereçamento. Com um identificador de endereço maior, maior será o tamanho de cada mensagem trafegada, pois o endereço da origem e do destino precisam estar inseridos nesta mensagem. Como em RS os recursos são limitados, se faz necessário sempre procurar trafegar o mínimo de informação, evitando o tráfego de dados desnecessários.

Algumas das alternativas para endereçamento em RSSFs [59]:

- Espacial: neste modelo de endereçamento é utilizado as coordenadas geográficas como identificar um nó individual. Os protocolos que utilizam este tipo de endereçamento também são conhecidos como algoritmos geográficos. Dependendo da quantidade de nós que estão depositados no ambiente, o endereço pode torna-se muito grande em relação aos dados trafegado.
- Baseado em atributos: diferente dos outros tipos de endereçamentos, o baseado em atributo tem como base atributos relevantes, que são utilizados como chave distribuídas pela estação base. Protocolos como o SPIN (Sensor Protocols for Information via Negotiation) e DD (Directed Difusion) utilizam este tipo de endereçamento.
- De transações: os nós selecionam probabilisticamente identificador único para cada transação. O RETRI (Random Ephemeral TRansaction Identifiers), atribui um identificador aleatoriamente para cada pacote [17]. Todos os fragmentos do pacote recebem o mesmo identificar do pacote, permitindo deste modo sua reconstrução para quem receber esses fragmentos.

3.4 Tipo de rede escolhido

A pesquisa foi desenvolvida considerando uma rede de sensores de composição homogênea, organização plana, distribuição irregular e sensores estáticos. Como se está trabalhando com o objetivo de aumentar o tempo de vida de uma rede de sensores com um eficiente gerenciamento de energia, nada melhor do que uma rede homogênea para poder melhor avaliar os resultados da pesquisa, já que em uma rede heterogênea pode haver sensores com capacidade de energia diferenciada dificultando a avaliação dos resultados. Quanto à organização da rede adotamos a característica plana porque nosso foco é o gerenciamento de energia sem perda de eficiência do roteamento e em uma rede hierárquica a disposição dos setores pode variar muito de um ambiente para outro, além de ter um impacto direto no consumo da energia da rede, também dificultando a avaliação dos resultados.

A adoção de uma rede com as características citadas acima se deve também ao fato de a integração entre IA e RS ser um assunto recente e ser necessário uma topologia de rede simples para que o potencial da técnica possa ser observado. Entretanto, o modelo de rede escolhido

não está distante do que é utilizado em aplicações reais, pois redes de organização plana e distribuição irregular são bastante comuns. A escolha de rede homogênea permitirá melhor avaliar o tempo de vida de uma rede de sensores considerando o gerenciamento de energia. Optamos por sensores com característica estática devido à complexidade de se trabalhar com sensores móveis. A comunicação entre os nós é feita utilizando o modo de transmissão Full-duplex.

3.5 Arquitetura de Comunicação

O plano de gerenciamento de tarefas tem como foco gerenciar trabalho que os sensores devem executar na sua região, tais como, sensoriamento e atuação sobre ambiente. O plano de gerenciamento de mobilidade tem a função de administrar os movimentos do sensor, manter atualizada as informações de quem são seus vizinhos e o nível de energia que estes possuem. O plano de gerenciamento de energia administra como o sensor vai utilizar este recurso.

O sensor poderia, por exemplo, permanecer no estado mínimo de consumo e somente ser ativado quando algum outro sensor o notificar enviando uma mensagem. Outra tarefa deste plano poderia ser notificar seus vizinhos quando o sensor estiver com seu nível de energia baixo, para que os vizinhos não o utilizem como rota, deixando os poucos recursos restantes para sensoriamento. Estes planos gerenciam como o sensor vai gastar seus recursos. Portanto, para que um roteamento seja eficiente, se faz necessário que estes planos trabalhem juntos.

Combinando com os planos de gerenciamento, a arquitetura de comunicação em RSSFs possui um pilha de protocolo (Figura 3.5) semelhante à arquitetura sugerida pelo TCP/IP, que basicamente consiste da camada de aplicação, camada de transporte, camada de rede, camada de enlace (dados) e camada física [1].

3.5.1 Camada de Aplicação

O objetivo desta camada é tornar o *hardware* e o *software* das outras camadas transparentes para as aplicações que manipulam as redes de sensores. Com base nas tarefas de sensoriamento, o desenvolvimento de diferentes tipos de aplicações podem ser feitas e utilizadas nesta camada. Em [1] foram citadas três aplicações: *Sensor Management Protocol* (SMP), *Task Assignment and Data Advertisement Protocol* (TADAP), e *Sensor Query and Data Dissemination Protocol* (SQDDP).

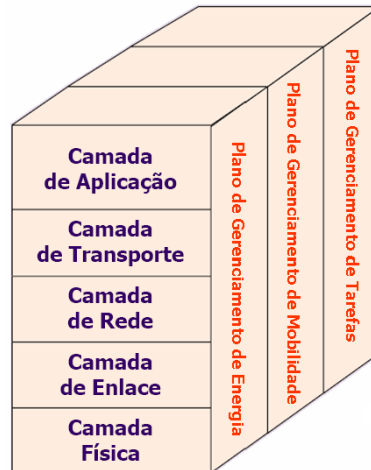


Figura 3.5: Pilha de protocolo [1]

3.5.2 Camada de Transporte

A natureza colaborativa das RSSF trazem diversas vantagens em relação ao sensoriamento tradicional, incluindo a maior precisão e extensão da área de cobertura. Para que estas potencialidades sejam exploradas, torna-se necessária a comunicação eficiente e confiável entre os nós da rede. Para isso, é preciso desenvolver um protocolo de transporte confiável. Os principais objetivos da camada de transporte estão listados a seguir:

- Interconectar as camadas de aplicação e de rede através de técnicas de multiplexação e demultiplexação;
- Prover um serviço de entrega de dados com um mecanismo de controle de erro especificado pelo nível de confiabilidade exigido pela camada de aplicação;
- Regular o volume de tráfego injetado na rede através de mecanismos de controle de fluxo e congestionamento.

3.5.3 Camada de Rede

Os nós sensores estão espalhados densamente sobre uma região de sensoriamento. Protocolos especiais de roteamento com suporte a vários saltos entre os nós sensores são necessários. As técnicas de roteamento em redes ad hoc já propostas na literatura normalmente não se encaixam nos requerimentos de uma rede de sensores. A camada de rede de uma rede de sensores possui alguns princípios que precisam ser considerados. Em primeiro lugar, a

eficiência quanto ao gasto de energia é sempre um fator importante. O segundo princípio diz que as RSSF têm, na sua maioria, roteamento baseado no conteúdo. Além disso, a agregação de dados é útil apenas quando ela não atrapalha o esforço conjunto dos nós sensores.

3.5.4 Camada de Enlace

Esta camada tem como responsabilidade a multiplexação dos fluxos de dados, detecção dos pacotes, acesso ao meio e controle de erro. Ela garante uma comunicação ponto a ponto e ponto a multipontos em uma rede de comunicação. O controle de acesso ao meio em uma RSSF deve atingir dois objetivos: *(i)* o primeiro é a criação de uma infraestrutura - isto é necessário pois o controle de acesso ao meio deve estabelecer comunicação salto a salto e fornecer à rede a habilidade de se auto-organizar; *(ii)* o segundo objetivo é a divisão justa e eficiente dos meios de comunicação entre os nós sensores.

3.5.5 Camada Física

Essa camada tem como responsabilidade a seleção de frequências, geração da frequência portadora, detecção de sinal, modulação e codificação. Comunicações sem fio a longas distâncias podem ser despendiosas, tanto em termos de energia quanto em complexidade de implementação. Quando se projeta uma camada física para uma rede de sensores, a minimização de energia se mostra de alta importância, acima inclusive dos problemas tradicionais de uma comunicação sem fio como reflexão, sombreamento etc. Logo, o projetista deve ter estes problemas e restrições em mente para melhor projetar a camada física.

3.6 O problema do roteamento

Além das restrições inerentes a uma RSSF refletido na complexidade de implementação da arquitetura de comunicação, outros fatores afetam diretamente o processo de roteamento de mensagens, tais como [2] [53]:

- A distribuição dos sensores: a densidade de distribuição e posicionamento pode alterar a quantidade de sensores visitados e o trajeto da rota;

- O consumo de energia: afeta diretamente o tempo de vida do sensor, provocando falhas em rotas estabelecidas previamente e também alterando dinamicamente o trajeto do roteamento;
- Nós (sensores) heterogêneos: sensores podem ter um papel diferente ou possuir capacidade diferente em termos de computação, comunicação e energia. Na mesma rede pode haver sensores especiais com diferentes funcionalidades. Por exemplo, alguns sensores podem ser responsáveis pela captura de imagens enquanto outros podem somente fazer a leitura de temperatura;
- Tolerância à falha: sensores podem falhar perdendo sua capacidade de processar, coletar ou transmitir suas informações devido a falta de energia ou interferência do ambiente. A falha de um sensor não deve afetar a rede, pois outra rota deve ser definida sem o sensor que falhou para que a meta (destino) seja alcançada;
- Quantidade de sensores: geralmente às RSSF são compostas de centenas às milhares de nós, mas o roteamento deve possuir o mesmo comportamento independente do tamanho da rede;
- Mobilidade: quando os sensores são móveis é gerado um grande tráfego de informações, já que para cada alteração de movimento se faz necessário atualizar as informações no sensor;
- Tamanho da área de cobertura: como cada sensor só possui informações locais e se comunica apenas com seus nós adjacentes (vizinhos), a área de cobertura que se deseja é um importante parâmetro;
- Densidade de distribuição: em algumas situações vários sensores que estão próximos uns dos outros acabam por reportar as mesmas informações para a base, gerando assim uma redundância nas informações que acaba por consumir recursos da rede sem necessidade;
- Privacidade: deve-se garantir que se indivíduos externos não autorizados têm acesso aos dados transmitidos, estes estejam criptografados, sendo necessário prever recursos de processamento e energia para essa tarefa.

Uma das principais preocupações do roteamento em redes de sensores é fazer com que o tempo de vida da rede seja degradado o mínimo possível, por isto, é necessário que o algoritmo de roteamento empregue um mecanismo de comunicação eficiente e robusto em termos de

gerenciamento de energia. O foco de um protocolo em redes de sensores deve ser economizar em processamento e comunicação.

Devido ao grande número de sensores, existe um grande número de interações ocasionando também um alto consumo de energia que deve ser otimizado por um algoritmo de roteamento eficiente.

Um protocolo de roteamento eficiente é composto basicamente por três fases [24]:

- **Descoberta da rota:** esta fase visa criar uma rota de um determinado sensor origem até o sensor alvo (destino) geralmente passando por diversos sensores.
- **Atualização de informação:** nesta fase os pacotes de dados solicitados são enviados para o sensor que os requisitou pela rota criada anteriormente.
- **Manutenção da rota:** uma vez descoberta esta rota se faz necessário monitorar a rota descoberta, para que ela continue válida mesmo quando um sensor que pertence à rota falha, descobrindo outros sensores substitutos.

Os protocolos de roteamento são classificados em pró-ativos, reativos e híbridos. Pró-ativos exigem que todos os nós conheçam as possíveis rotas, de tal maneira que quando houver necessidade de transmitir algum pacote de informação, a rota seja conhecida para uso imediato. Por isto, este tipo de protocolo mantém uma tabela de roteamento e troca mensagens para mantê-la sempre atualizada. Protocolos reativos descobrem a rota somente quando precisam enviar pacotes de dados para um determinado destino, buscando o gerenciamento eficiente de recursos de energia e banda. Protocolos híbridos são compostos pelas duas abordagens citadas anteriormente, onde um conjunto de nós realiza atualizações em sua tabela de roteamento enquanto outros nós apenas traçam esta rota quando houver necessidade de transmitir pacotes de informações.

Nesta seção foram discutidos alguns problemas (situações) envolvidos que devem ser estudados e levados em consideração ao se trabalhar com redes de sensores. Nas próximas seções serão apresentadas algumas técnicas e projetos relacionados com roteamento em redes de sensores que foram desenvolvidos ou estão em desenvolvimento, uma vez que a contribuição deste trabalho está direcionada principalmente com o problema de descoberta de rotas.

3.7 Algoritmos de roteamento

Existem inúmeros algoritmos de roteamento desenvolvidos para redes de sensores. Nesta seção são descritos alguns deles: DD (*Directed Diffusion*), MCFA (*Minimum Cost Forwarding Algorithm*), EAR (*Energy Aware Routing*), GBR (*Gradiente-Based Routing*), BVR (*Beacon Vector Routing*), DSR (*Dynamic Source Routing*), AODV (*Ad-hoc On-demand Distance Vector*) e NADV (*Normalized Advance*).

Na DD (difusão direta) cada sensor nomeia os dados que ele gerou usando um ou mais atributos [2] [37] [57] [7] [36]. A estação base pode solicitar dados e disseminar seu interesse, que os nós intermediários se encarregam de propagar. Interesses estabelecem gradientes de dados em direção a estação base que expressou este interesse. O sensor que receber o interesse propagado ativa seu sensoriamento para coletar informações referente à aquele interesse. Quando coletadas estas informações ele envia os dados pelo caminho inverso da propagação de interesse que está armazenado em cache. Uma importante característica da DD é que a propagação de dados, interesse e agregação são determinados pela troca de mensagem entre sensores vizinhos.

Já com o algoritmo de roteamento de MCFA [2] [57] explora-se o fato de que o caminho do sensor até a estação base é conhecido, não havendo necessidade de manter tabelas de roteamento na estação base. Cada sensor conhece o caminho de menor custo estimado de si próprio até a estação base. Portanto quando um nó deseja enviar uma mensagem para a base, ele faz um broadcast para seus vizinhos. Quando o nó recebe a mensagem ele verifica se seu custo é menor do que aquele de quem recebeu a mensagem. Se ele estiver no caminho de menor custo, então ele faz um broadcast da mensagem para seus vizinhos, senão ele a descarta. Isso se repete sucessivamente até a mensagem alcançar a estação base. Cada sensor deve conhecer o caminho de menor custo de si mesmo até a estação base. Isto é obtido por meio de uma mensagem de broadcast enviada pela estação base que se propaga para todos os sensores. Cada nó recebe a mensagem de broadcast enviada pela estação base e checa se o caminho pelo qual recebeu é de menor custo que o caminho que já tinha armazenado em si próprio. Em caso positivo ele armazena o novo caminho, senão ele desconsidera a mensagem, sendo feito este teste a cada broadcast recebido da estação base. Cada nó pode fazer vários *updates* em seu caminho armazenado até receber o caminho de menor custo.

Diferentemente, no GBR (*Gradiente-Base Routing*) [2] [37] [57] [7] cada sensor memoriza a quantidade de sensores (pulos) necessários para traçar uma rota de um determinado

sensor até a a estação base de maneira autônoma. Cada sensor tem um parâmetro chamado altura (*height*) que é o número mínimo possível de pulsos (sensores) que a mensagem pode percorrer até alcançar a estação base. A diferença entre a altura do sensor e de seus vizinhos é considerada como o gradiente desta ligação. Os pacotes de dados são roteados para os sensores com mais alto gradiente na ligação. No GBR dois esquemas para a disseminação de dados podem ser utilizados para balancear o tráfego na rede, aumentando seu tempo de vida. No esquema estocástico de disseminação de dados o sensor escolhe um gradiente aleatoriamente quando dois ou mais sensores têm o mesmo gradiente. No esquema baseado em energia o sensor aumenta sua altura quando a energia baixa de certo limiar, desestimulando o envio de mensagens para este sensor. Já no esquema de disseminação de dados baseado em fluxo, os novos fluxos não são roteados através de sensores que já fazem parte de algum caminho de outro fluxo.

Outra técnica que foi desenvolvida para roteamento em RSSF é o EAR [2] [57], cuja sobrevivência da rede é o foco principal. No EAR é mantido um conjunto de caminhos no lugar de obrigar a utilização de um caminho ótimo por várias vezes consumindo seus recursos por uso excessivo do mesmo caminho. Estes caminhos são escolhidos por meio de uma probabilidade, sendo que o valor desta probabilidade depende de como o baixo consumo de energia de cada caminho pode ser obtido. Como caminhos diferentes são escolhidos a cada vez, a degradação da rede é mais equilibrada, não degradando um caminho único rapidamente, proporcionando um balanceamento entre os nós e melhorando o tempo de vida da rede.

Existe também uma outra técnica de roteamento chamada BVR [22] [13] que utiliza a abordagem da mínima distância. Na rede são estabelecidos vários nós sinalizados (*Beacons*). É atribuído um vetor de distância para cada nó da rede, identificando a distância deste até os nós sinalizados. Os nós sinalizados transmitem de tempos em tempos a informação de sua localização para todos os nós da rede. Para o nó de origem S1 enviar dados para o nó destino S9, é necessário que S1 escolha um vizinho que minimize a função de distância dada por:

$$o(k, n, q) = \sum_{i \in c(k, q)} w_i(n, q) * |B_i(n) - B_i(q)|, \text{ onde:}$$

- $k \rightarrow$ é o número de nós sinalizadores;
- $n \rightarrow$ é o nó vizinho;
- $q \rightarrow$ é o destino que se pretende alcançar;
- $B_i(n) \rightarrow$ é a distância do nó vizinho até o nó sinalizador mais próximo deste;

- $B_i(q) \rightarrow$ é a distância do nó alvo até o nó sinalizador mais próximo deste destino;
- $w_i(n, q) \rightarrow$ é um peso atribuído conforme a seguinte regra: se $B_i(n) > B_i(q)$ então atribui 10 senão atribui 1.

Quando acontecer do vizinho escolhido falhar, a mensagem será re-transmitida para próximo vizinho que seria o segundo melhor, caso este também falhe, então será feito um broadcast da mensagem para todos os vizinhos.

Outra técnica que também foi desenvolvida recentemente é o DSR [34] sendo considerado um protocolo de roteamento em-demanda, que possui dois mecanismos que trabalham juntos:

- **Descoberta da rota:** é o mecanismo no qual um nó que deseja enviar dados para outro nó precisa obter a rota para o tráfego destes dados. Este mecanismo só é utilizado se o nó origem não conhece a rota para o nó destino.
- **Manutenção da rota:** este mecanismo é responsável por detectar qualquer alteração que ocorra na rota, tais como falha de nós que faziam parte da rota ou outra qualquer alteração na topologia da rede.

Para estabelecer uma rota utilizando o protocolo DSR, o sensor origem (S) transmite uma mensagem solicitando a rota para o sensor destino (D), a qual é recebida por todos os vizinhos de S. Cada nó que receber esta mensagem, acrescenta seu próprio endereço dentro da mensagem e a re-transmite para seus vizinhos. Se a mensagem já contém o endereço do nó ou uma outra cópia da mensagem já passou por este nó, então este ignora a mensagem. O processo continua até a mensagem de solicitação de rota alcançar D. Quando a mensagem de solicitação de rota alcançar D, este nó emite uma outra mensagem de retorno em direção a S indicando o caminho pelo qual S deve enviar os dados.

Para a manutenção da rota no DSR, o sensor S transmite uma mensagem de checagem para seu vizinho solicitando sua confirmação se este recebeu a mensagem. O vizinho também emite uma mensagem de checagem para o próximo nó solicitando sua confirmação. O processo é repetido pelos outros sensores da rota até a mensagem alcançar D. Se algum dos sensores da rota não confirmar, o nó que detectou esta falha de seu vizinho emite uma mensagem em direção a S notificando o erro na rota. Quando S é notificado da falha da rota, este pode solicitar uma nova descoberta de rota.

Assim como o DSR, o AODV [66] [54] [69] [23] também é um protocolo de roteamento em-demanda, isto é, ele só estabelece a rota para o destino quando há necessidade de trafegar alguma informação para este destino. Quando um nó (S) deseja trafegar dados para um determinado destino (D), este transmite uma mensagem fazendo um broadcast para seus vizinhos solicitando uma rota até D. Esta mensagem possui um identificador único, permitindo deste modo que os nós detectem e apaguem mensagens duplicadas. Quando o nó vizinho recebe a solicitação da rota, este grava em sua tabela de roteamento o caminho inverso para nó anterior de onde foi recebido esta mensagem e as informações do nó origem da solicitação da rota. O cache de roteamento que cada nó possui, tem um tempo de vida que é definido dentro da mensagem de solicitação da rota. Quando a mensagem de solicitação de rota alcançou D, este transmite uma nova mensagem de resposta fazendo um *unicast* para o vizinho por onde chegou a solicitação da rota. Essa mensagem de resposta vai seguindo o caminho inverso até alcançar S. Quando alcançar a origem S, a rota foi estabelecida, podendo enviar os dados. Toda vez que os dados trafegam por esta rota, o contador de expiração da vida do cache do caminho inverso de cada nó que faz parte da rota é reinicializado. Se determinada rota ficar sem tráfego de dados durante um tempo maior do que aquele definido como tempo de vida do cache, será necessário uma nova solicitação para o estabelecimento de rota.

O protocolo NADV [42] também utiliza informações da localização dos sensores para a entrega de pacotes de dados. A estratégia para o tipo de roteamento geográfico é simplesmente encaminhar os pacotes de dados para o vizinho que estiver mais próximo do destino. Contudo, no NADV a escolha do vizinho para o qual vai rotear a mensagem, considera a proximidade deste vizinho até o destino do roteamento e também a qualidade do link para este vizinho. O custo do link pode ser de vários tipos, por exemplo, as mensagens com alta prioridade podem ser roteadas para o link que oferece a menor latência, e as mensagens com baixa prioridade podem ser roteadas para o link que minimiza o consumo de energia aumentando o tempo de vida da rede, ou ainda, pode ser considerado o grau de ruído do link. Se o sensor S deseja alcançar o destino G, então S deve escolher o valor de n que maximiza a fórmula: $NADV(n) = \frac{ADV(n)}{Cost(n)}$, onde n é o vizinho candidato para o envio da mensagem e, ADV(n) é dado por $ADV(n) = D(S) - D(n)$, onde D(s) é a distância em saltos de S até G e D(n) é a distância em saltos de n até G. A estratégia busca maximizar o ADV(n) para o próximo salto. O Cost(n) pode ser considerado uma variedade de tipos de custos individuais ou fazendo uma combinação entre estes. Neste protocolo a seleção do próximo sensor a ser enviada a mensagem será sempre aquele que maximizar o NADV.

3.8 Redes de Sensores, RDP e SMA

Uma das primeiras áreas exploradas com técnicas de resolução distribuída de problemas baseadas em Inteligência Artificial foi o monitoramento de dispositivos em ambientes dinâmicos. Trabalhos como *Contract-Net*, *Multistage* e *DVMT* utilizam termos e conceitos comuns a redes de sensores, embora tenham objetivos específicos e sejam considerados trabalhos pioneiros para a Inteligência Artificial Distribuída (IAD).

Um desses trabalhos, o *Contract-Net* [63], é um protocolo de interação que facilita o controle distribuído na execução cooperativa de tarefas entre os nós distribuídos. Os nós devem estar interconectados e a comunicação entre eles é feita pelo envio de mensagens, não havendo o compartilhamento de memória. Este protocolo estabelece uma estruturação de alto nível para as interações entre os nós utilizando a negociação entre eles como mecanismo. Este protocolo deixa de ser interessante quando os recursos disponíveis para comunicação entre os nós é limitado, pois para o contract-net cada negociação é feita através de muitas trocas de mensagens.

Outro trabalho desenvolvido foi *Distributed Vehicle Monitoring Testbed (DVMT)* [16] que simula uma rede de tráfego rodoviário, onde cada nó é responsável por uma região dentro de determinada área. Após o nó detectar a presença de um veículo, ele se comunica com os outros para resolver ambiguidades, estimar com mais precisão a localização do veículo, evitar dados ruidosos e eliminar veículos fantasmas.

Quando um veículo entra na área coberta por mais de um nó, estes cooperam entre si para confirmar ou anular a detecção do veículo. Este é um problema natural para uma abordagem de resolução distribuída de problema [44], já que os nós sensores estão localizados em uma ampla área geográfica, interação entre si para eliminar possíveis inconsistências e cada nó somente pode se comunicar com outro que esteja próximo (ao alcance).

Com base nestes trabalhos pioneiros da IAD é possível verificar que em uma rede distribuída cada agente deve ter uma visão topológica local devido à limitação de banda e o gasto excessivo provocado pela troca de informações necessárias para manutenção de uma topologia global. Isso exige mecanismos distribuídos de coordenação.

O protocolo de negociação *multistage* é utilizado para planejamento em um ambiente distribuído com controle descentralizado e comunicação limitada entre os agentes [11]. Quando a satisfação de um objetivo requer o compromisso de recursos, pode surgir competição entre os objetivos pelos recursos limitados. Este protocolo provê um mecanismo para alcançar um

consenso entre os agentes com objetivos contraditórios acerca de uma solução admissível. O objetivo deste protocolo é permitir que os nós troquem informações de forma que interações úteis sejam detectadas e manipuladas de maneira razoável.

Existem também alguns trabalhos utilizando agentes que tentam otimizar o uso de recursos em RSSF com uma abordagem de agentes móveis, onde um agente contém a mensagem transmitida e informações complementares de roteamento [55]. Nesta abordagem um ou mais agentes transitam por entre os sensores em direção ao alvo. Este agente móvel é formado pelos atributos identificação, buffer, itinerário e método.

Em contrapartida, a abordagem apresentada neste trabalho parte do princípio que cada sensor é um agente desenvolvido para resolver um problema específico, sendo todos homogêneos e a interação entre estes agentes se dá por meio de simples trocas de mensagens entre vizinhos. A escolha da ação de cada agente sensor é baseada apenas no seu conhecimento local que engloba informações sobre seus vizinhos e nas informações recebidas com as mensagens. Com base nestas informações o agente escolhe o agente mais apropriado para interagir em busca da solução.

Dessa forma, podemos visualizar essas redes como uma sociedade de pequenos agentes com capacidade limitada, que não possuem habilidade individual de encontrar a solução apropriada para o problema que se pretende resolver. Contudo, a interação entre esses agentes pode alcançar a solução desejada.

Um outro trabalho desenvolvido para RSSF é uma plataforma utilizada com agentes móveis chamada *ActorNet* [41]. Este framework é uma implementação do modelo de atores da computação distribuída. Como implementar agentes diretamente para redes de sensores pode ser um pouco complicado devido às limitações que o ambiente destas RSSF possuem, o *ActorNet* visa facilitar à criação de agentes que utilizam um modelo de comunicação assíncrono e suporte para coordenação entre estes agentes. Utilizando este framework baseado em agentes pode-se facilmente criar uma rede de sensores e experimentar técnicas que buscam solucionar problemas encontrados neste tipo de redes com recursos limitados.

Um outra abordagem também baseada em agentes é o *Distributed, Autonomic, Reconfigurable Wireless Network (DARWIN)* [67] que é muito similar ao modo de trabalho do protocolo de negociação *Contract-Net*. Um agente é apontado para ser o coordenador do algoritmo. Este coordenador envia requisições para seus agentes e verifica a redundância no retorno do resultado que estes agentes enviam. Com base nestas informações de retorno e na distância entre estes agentes o coordenador decide se coloca ou não o agente para hibernar

(desativa). Periodicamente o coordenador irá acordar os agentes e novamente proceder a uma verificação de redundância entre os agentes.

Outro trabalho que também utiliza a abordagem de SMA é o *Multi-Sensor Agent-Based Intrusion Detection System (MSABIDS)* [70] que busca detectar intrusos na rede. Os agentes agem como sensores que coletam dados do ambiente no qual estão operando, raciocinam encima destes dados coletados interagindo uns com os outros e sugerem ações para que a invasão possa ser evitada.

Em [25] também é introduzido o conceito de *Agent-Organized Network (AON)* como uma topologia de interação entre os agentes que tomam decisões individuais com base em informações locais para formarem um time. A formação de um time (também chamado de rede social) consiste de inúmeros agentes que devem possuir uma única conexão com seu time. Por exemplo, os agentes A, B, C e D antes de formarem um time possuem diversas ligações sociais como pode ser visto na Figura 3.6.

Porém, observe que após formar um time (Figura 3.7) estes agentes só possuem um ligação social com o time. Qualquer agente poderá ser adicionado ou retirado do time desde que este possua uma ligação com qualquer outro agente que já esteja no time.

A formação de um time ocorre para a execução de uma determinada tarefa, portanto a estratégia considera aqueles agentes que tenham a habilidade necessária para o trabalho. Um agente está disponível quando seu status está "UNCOMMITTED" e muda seu status para COMMITED quando ele foi selecionado para o time, mas este time ainda não está totalmente formado. Quando o time estiver totalmente formado então começa a execução da tarefa para a qual o time foi formado e o status dos agentes passa a ser "ACTIVE".

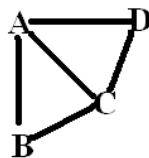


Figura 3.6: Agentes antes de formarem o time

Nesses trabalhos relacionados com RSSF foi possível observar que geralmente os agentes são reativos, isto é, para se alcançar determinado sensor a partir da estação base, esta emite um estímulo (mensagem) para o sensor vizinho que tenha o melhor custo, e este agente

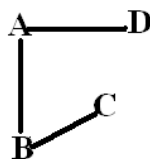


Figura 3.7: Agentes após formarem um time

reage ao estímulo processando a mensagem recebida e em seguida retransmite este estímulo para o agente seguinte. Este processo se repete até alcançar a solução (alvo).

Os agentes utilizados para formar uma rede, geralmente são criados para um propósito específico. Portanto sua inteligência está codificada dentro dele próprio, por isso nossa abordagem é desenvolvida a partir de uma perspectiva da Resolução Distribuída de Problema. Não optamos por uma abordagem focada em SMA, devido à complexidade necessária e por se tratar de agentes com recursos limitados, sem muitos recursos computacionais disponíveis e com comunicação restrita entre eles [45].

Como as mensagens não podem ser transmitidas e recebidas simultaneamente, e dois agentes não podem transmitir em um mesmo canal ao mesmo tempo sem causar interferência [45]. Além disso, há necessidade de coordenação, o que caracteriza mais ainda esses sensores como sendo agentes.

Existem várias razões para se adotar a abordagem de resolução distribuída de problema em problemas encontrados em RSSF, tais como: velocidade, confiabilidade, extensibilidade e principalmente a habilidade de resolver problemas que têm um espaço natural de distribuição [63].

3.9 Considerações Finais

Neste capítulo foram apresentados diversos conceitos sobre Redes de Sensores, além de uma discussão sobre as interações entre RSSF e IAD. Na próxima seção é apresentada a contribuição principal desta pesquisa: a proposta de um novo algoritmo para roteamento em redes de sensores baseado em Inteligência Artificial Distribuída.

Capítulo 4

Algoritmo AD*

4.1 Considerações Iniciais

Nos capítulos anteriores foram discutidos alguns conceitos pertinentes a RSSF, tais como tipos de redes de sensores, arquitetura de comunicação e a relação que existe entre Redes de Sensores, Sistemas Multi-Agentes e Resolução Distribuída de Problemas. Verificou-se também que um algoritmo de roteamento deve considerar características como distância entre os sensores e a energia restante do sensor, com o objetivo de balancear o consumo de energia na rede, proporcionando um tempo de vida mais longo para a RSSF. Neste capítulo é proposto o algoritmo de roteamento AD* que tem como base o tradicional algoritmo de busca em grafo A* tradicionalmente utilizado na computação não distribuída. Porém o AD* trabalha de forma distribuída, uma vez que os sensores não são distribuídos e possuem apenas informações locais.

4.2 Princípios do AD*

O nome do algoritmo, AD*, foi escolhido porque ele é baseado em uma implementação do A* (A estrela), de maneira distribuída (D). Este algoritmo tem como papel principal descobrir a melhor trajetória para uma mensagem com destino definido trabalhando com nós autônômos.

Para este trabalho é assumido que o sensor de origem (SO) também chamado de *estação base (EB)* conhece as coordenadas de todos os sensores na rede, porém os demais sensores da rede só conhecem as coordenadas de seus vizinhos e da EB. O sensor destino (SD) pode ser qualquer sensor da rede para quem o SO solicitou determinada informação a ser coletada ou

para uma solicitação de execução de alguma tarefa. Cada sensor utilizado em nossa rede de sensores possui um identificador único (endereço).

A estação base serve como meio de comunicação entre a rede de sensores e o usuário final. Como os sensores só conseguem se comunicar com seus vizinhos, não possuindo acesso direto à estação base, estes vizinhos devem fazer o papel de roteadores para que a mensagem de determinado sensor alcance a base ou a mensagem da estação base alcance determinado sensor.

O objetivo deste algoritmo é estabelecer a rota de menor custo levando em conta a distância e a energia restante dos sensores. Quando o algoritmo está descobrindo uma rota e esta deixa de ser a melhor, é então feito um re-planejamento (re-roteamento) para que a informação seja transmitida do sensor atual até o sensor que possui a melhor rota. Após a chegada da informação neste sensor, o algoritmo continua com seu roteamento normalmente, com os sensores transmitindo as informações de um vizinho para outro até alcançar o seu destino. O algoritmo proposto mantém uma lista de caminhos candidatos enquanto estiver traçando a rota.

Como geralmente as RSSF são compostas de milhares de sensores, os caminhos candidatos podem formar uma lista grande, as mensagens que trafegam pela rede pode consumir muito recurso. Para evitar o problema de trafegar mensagens na rede por dezenas/centenas de caminhos candidatos, optou-se por aplicar uma técnica chamada Busca Dirigida (*Beam Search - BS*).

BS é um método heurístico para a resolução de problemas de otimização combinatorial [68] [5]. O procedimento consiste em avaliar nós de cada nível da árvore de busca e selecionar somente aqueles nós que prometem melhor ramificação, enquanto o restante dos nós são cortados permanentemente da árvore. Como grande parte da árvore é podada e somente poucos nós ficam em cada nível, o tempo corrente é polinomial em relação ao tamanho do problema.

A abordagem clássica do BS consiste em podar o ramo e deixar somente os N melhores nós em cada nível, onde N é chamado largura da busca. Os outros nós são completamente descartados, já que o objetivo desta técnica é a busca rápida. O BS não garante encontrar uma solução ótima, pois se os nós principais para a melhor solução são descartados durante o processo de poda, não existirá meio de alcançar uma solução ótima posteriormente. A abordagem *beam search* minimiza este perigo selecionando um número de caminhos que aparentemente prometem ser de boa qualidade. Quanto maior for N , maior será a segurança em encontrar uma solução ótima, porém isso aumenta o esforço computacional.

O processo de avaliação é o ponto chave da técnica *Beam Search*. São dois os tipos de avaliação:

- Funções de avaliação de prioridade: esta função calcula as prioridades (urgências). Esta função possui somente uma visão local do problema, já que considera somente a próxima decisão a ser feita.
- Funções de avaliação do custo total: calcula uma estimativa do custo total mínimo da melhor solução que pode ser obtido. Esta função possui uma visão mais global, pois projeta da solução parcial corrente e estima o custo total.

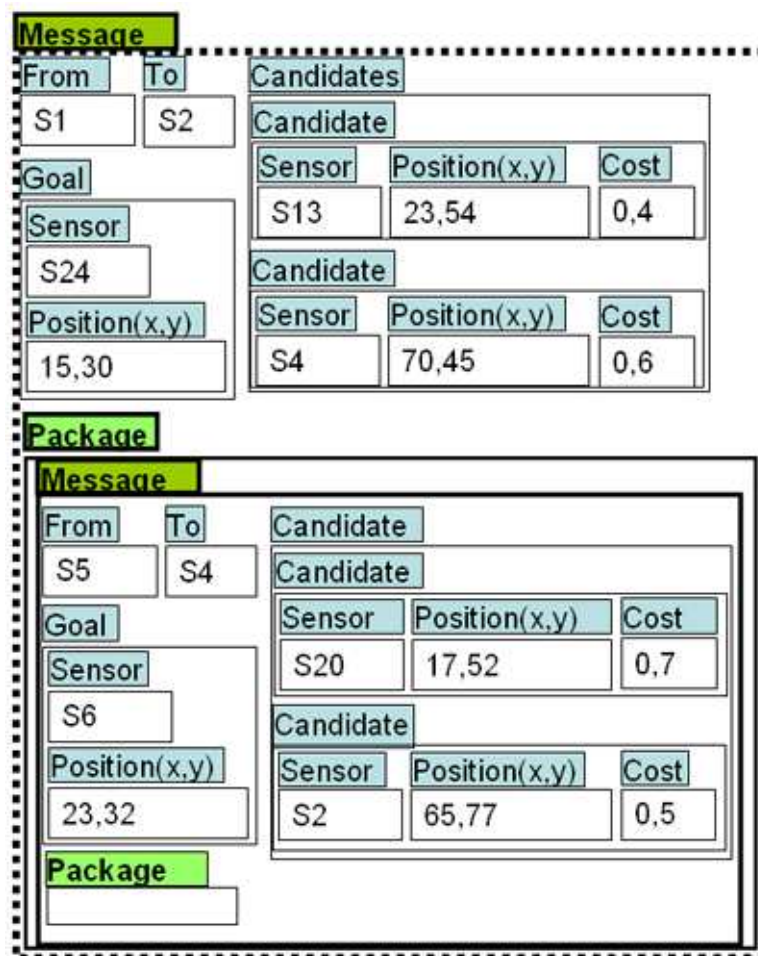


Figura 4.1: Estrutura da Mensagem

No AD*, após adicionar os vizinhos de um sensor para a lista de caminhos candidatos (expansão de um sensor), é aplicado o BS com avaliação do custo total, para deixar na lista de candidatos somente os N caminhos candidatos que tenham os menores custos de roteamento. Portanto, o AD* sempre carrega consigo informações dos N melhores caminhos até alcançar seu alvo. Os experimentos foram realizados utilizando $N=1$ e $N=2$. Todas as mensagens trocadas

entre os agentes sensores possuem o mesmo formato (Figura 4.1), de modo que é compreendida por qualquer sensor que venha recebê-la.

Conforme mostrado pela Figura 4.1, a mensagem é composta pelo identificador (From) de quem está enviando a mensagem e o identificador (To) para quem ela está sendo enviada. Também possui informações do sensor alvo (Goal), tais como, identificador do sensor alvo e a posição geográfica dele na rede. As informações dos caminhos candidatos que também faz parte da mensagem contém a identificação do último sensor do caminho com sua posição geográfica e a distância percorrida (Cust) por este caminho candidato desde a EB. A quantidade de caminhos candidatos que farão parte da mensagem poderá ser no máximo igual ao N definido como valor de poda pelo BS. O pacote (package) só vai conter algum valor quando no estabelecimento da rota acontecer um re-roteamento da mensagem de um caminho candidato que deixou de ser considerado o de melhor custo para o outro caminho que assumiu este status como sendo aquele de melhor custo.

4.3 Modelo de Consumo de Energia

O consumo de energia de um agente sensor só é computado para receber e enviar a mensagem. Neste trabalho, não está sendo considerado o consumo no processamento da mensagem. O consumo de energia para receber a mensagem é menor do que para enviá-la, pois para enviar a mensagem o sinal de transmissão do sensor é amplificado para ter uma alcance maior e evitar a perda dos dados durante a transmissão. Portanto, o tamanho da mensagem tem impacto direto sobre o consumo de energia.

Em nosso trabalho assumimos que o modelo de consumo de energia utilizado é o mesmo usado em [26], cujo modelo dissipa $E_{elec} = 50nJ/bit$ de energia para enviar ou receber uma mensagem e $E_{amp} = 100pJ/bit/m^2$ para uma transmissão amplificada. Portanto, para transmitir uma mensagem do agente sensor S_i até o S_j , a energia consumida por S_i para enviar a mensagem é:

$$E_{S_i}(k, d) = E_{elec} * k + E_{amp} * k * d^2, \quad (4.1)$$

e a energia consumida pelo sensor S_j para receber a mensagem é:

$$E_{S_j}(k, d) = E_{elec} * k, \quad (4.2)$$

onde k representa o tamanho da mensagem em $k-bit$, e d^2 é a distância ao quadrado [24]. Para este trabalho foi definido que cada sensor possui uma bateria de 0.5J assim como em [26].

4.4 Funcionamento do Algoritmo

No início do AD* é simulado o consumo de energia (Algoritmo 2 - linha 2) para receber a mensagem. Em seguida, verifica-se se o sensor que possui a mensagem é o sensor destinatário da mensagem (Algoritmo 2 - linha 6). Se o sensor atual for o sensor destino, então o algoritmo verifica se existe algum pacote de mensagem (Algoritmo 2 - linha 7), pois, neste caso o destino encontrado é de um re-roteamento (explicado nos parágrafos seguintes), sendo necessário descompactar a mensagem (Algoritmo 2 - linha 12-14) e continuar o roteamento (Algoritmo 2 - linha 17) a partir deste sensor. Caso o pacote de mensagem esteja vazio (Algoritmo 2 - linha 7) quando o sensor atual for o destino, então a rota foi traçada completamente (Algoritmo 2 - linha 8), concluindo o roteamento.

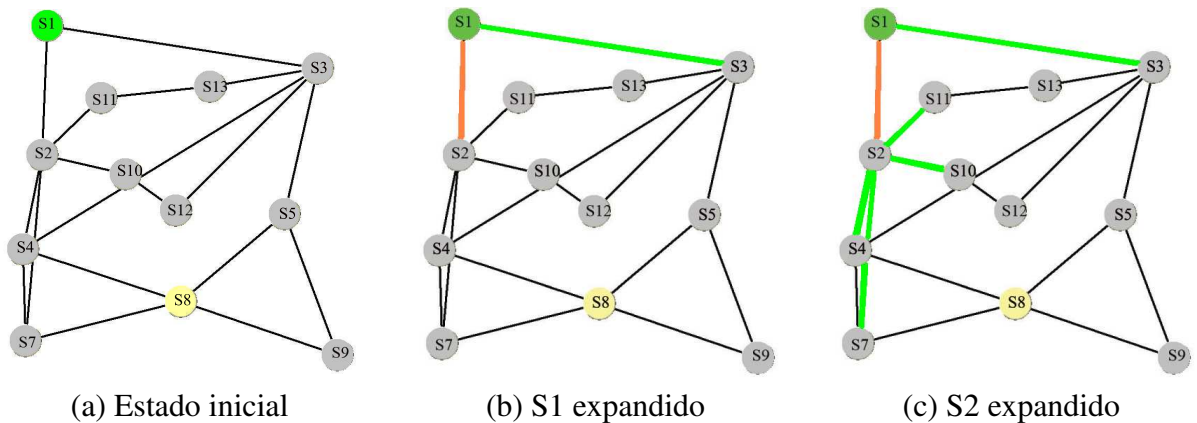


Figura 4.2: Rede de sensores

Quando o sensor recebe a mensagem, é necessário expandí-lo (Algoritmo 2 - linha 17) para saber quais são os possíveis caminhos a seguir. Por exemplo, observe a Figura 4.2a, e imagine que se deseje estabelecer uma rota do agente sensor S1 (EB) até o sensor S8. O sensor S1 é expandido (Figura 4.2b) para saber quais são os caminhos candidatos, que neste caso são: [s1,s3] e [s1,s2]. Quando o sensor expande, ele atribui um custo para cada um de seus possíveis caminhos, conforme a seguinte fórmula:

$$custo = \left(\frac{Dacum}{MaxD} + \frac{D}{MaxD} \right) * \alpha + CEestim + Econs. \quad (4.3)$$

A seguir é detalhado o cálculo deste custo usando o caminho [s1,s2], pois o cálculo é feito do mesmo modo para os outros caminhos. Detalhando a fórmula do custo, temos o *Dacum* que é a distância acumulada entre todos os sensores que compõe o caminho, neste caso por exemplo, o *Dacum* para [s1,s2] é a distância entre eles, pois eles são adjacentes. O *CEestim* é a

estimativa da energia que seria consumida para enviar uma mensagem de S2 até S8 (destino) em linha reta. A distância (D) é a distância estimada de S2 até S8. O $MaxD$ é distância estimada da estação base (S1) até o sensor alvo (S8), que é utilizada para normalizar as distâncias D_{acum} e D . Para este trabalho foi adotado 0.5 como o peso (α) que é atribuído às distâncias para que esta não seja predominante sobre a energia consumida. E_{cons} é a quantidade de energia já consumida do sensor S2.

Algoritmo 2 Pseudo-código do AD* (Parte I)

```

1: Rotear( msg )
2: ConsumptionReceive( sizeofMessageBit( msg ) )
3: goal = msg.goal
4: listOfPathCandidate = msg.listOfPathCandidate
5: packageMessage = msg.packageMessage
6: if goal.name = Myself then
7:   if packageMessage == null then
8:     betterPath = catchBetterPath( listOfPathCandidate )
9:     print betterPath
10:    STOP
11:  else
12:    goal = packageMessage.goal
13:    listOfPathCandidate = packageMessage.listOfPathCandidate
14:    packageMessage = packageMessage.message
15:  end if
16: end if
17: pathCandidate = Expand(myself, goal, listOfPathCandidate, myNeighbors)
18: pathCandidate = BeamSearch(pathCandidate)
19: if pathCandidate.count > 1 then
20:   betterPath = catchPathBetter(pathCandidate)
21: else
22:   betterPath = pathCandidate
23: end if
24: lastSensor = catchLastNode(betterPath)
25: route = catchPenultimateNode(betterPath)

```

Algoritmo 3 Pseudo-código do AD* (Parte II)

```

1: if route != myself and lastSensor is not myneighbor then
2:   package.goal = goal
3:   package.listOfPathCandidate = pathCandidate
4:   package.message = packageMessage
5:   goal = lastSensor
6:   pathCandidate = expand(myself, goal, null, myNeighbors)
7:   pathCandidate = BeamSearch(pathCandidate)
8:   betterPath = catchBetterPath(pathCandidate)
9:   lastSensor = catchLastNode(PathBetter)
10:  msg = makeMessage(goal, pathCandidate, package)
11: else
12:  msg = makeMessage(goal, pathCandidate, packageMessage)
13: end if
14: Send msg to lastSensor
15: consumptionSend(sizeofMessagebit(msg))

```

Supondo que o custo de $[s1,s3]$ seja 0.77 e o custo de $[s1,s2]$ seja 0.70, o melhor caminho é aquele com o menor custo, neste caso seria o $[s1,s2]$. Observe, porém, que antes do sensor S1 escolher qual é o caminho a seguir, é necessário aplicar o BS (Algoritmo 2 - linha 18). Após a poda (BS), o sensor S1 deve então escolher o melhor caminho para seguir com a rota, neste caso, o caminho $[s1,s2]$ será o escolhido (Algoritmo 2 - linha 19-23).

Uma vez escolhido o caminho para seguir com o roteamento, o sensor S1, verifica se o sensor escolhido é seu vizinho para continuar o roteamento (Algoritmo 3 - linha 1), pois se o sensor escolhido for seu vizinho a mensagem pode ser enviada diretamente para ele (Algoritmo 3 - linha 12,14) que vai dar prosseguimento à rota. Caso o sensor que deverá receber a mensagem não seja seu vizinho, então se faz necessário re-rotear a mensagem do sensor atual para o sensor que passou a ser a melhor rota (Algoritmo 3 - linha 1-10), ou seja, descobrir um roteamento temporário e intermediário entre sensores que não são nem a origem nem o destino da mensagem inicial. Observe que neste caso, como o S2 é vizinho, S1 envia a mensagem diretamente para S2 (Algoritmo 3 - linha 12,14) e simula o consumo de energia para transmitir a mensagem (Algoritmo 3 - linha 15).

Neste momento o sensor atual é S2, que ao receber a mensagem simula o consumo de energia (Algoritmo 2 - linha 2). Como este sensor não é o sensor destino (Algoritmo 2 - linha 6) a expansão é feita diretamente (Algoritmo 2 - linha 17). A mensagem que S2 recebeu contém os caminhos candidatos [s1,s3] com custo 0.77 e [s1,s2] cujo custo é 0.70. Expandindo o sensor atual S2 (Figura 4.2c) são encontrados os candidatos [s1,s3] com custo 0.77 que veio com a mensagem e os novos candidatos que surgem com a expansão de S2 que são [s1,s2,s11]-0.82, [s1,s2,s10]-0.81, [s1,s2,s4]-0.78 e [s1,s2,s7]-0.79. O caminho de menor custo neste caso passou a ser [s1,s3] (Algoritmo 2 - linha 19-22).

Como o sensor atual é o S2 e a rota passou a ser melhor pelo S3 que não é vizinho de S2, se faz necessário rotear a mensagem de S2 para S3. Essa rota intermediária é chamada de re-roteamento (Algoritmo 3 - linha 1-10). Como a EB precisa alcançar o sensor S8 e, portanto, este sensor é o objetivo (alvo) original, ele é empacotado na mensagem (Algoritmo 3 - linha 2-4) e uma nova rota é traçada, onde a origem é S2 e o destino a ser alcançado é S3 (Figura 4.3a).

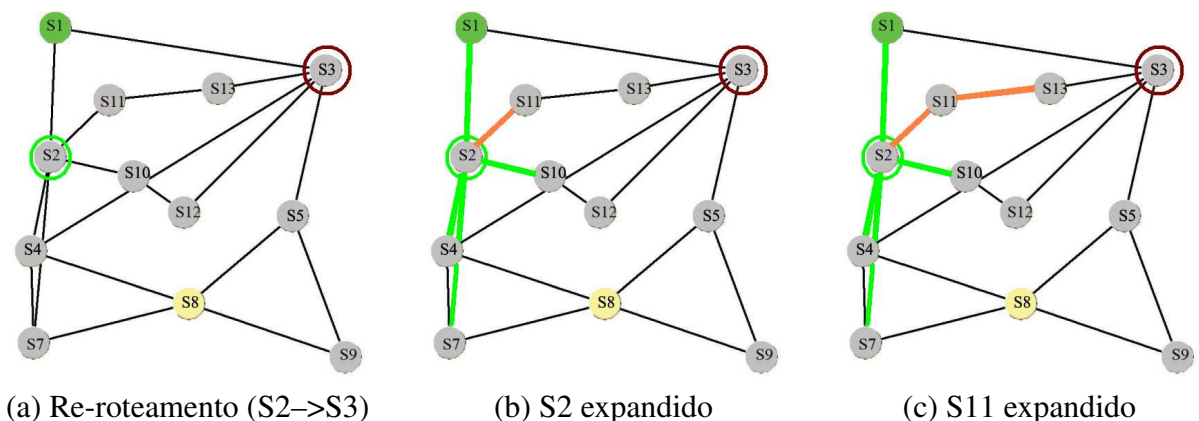


Figura 4.3: Rede de sensores

O S2 é expandido (Algoritmo 3 - linha 6), encontrando os seguintes caminhos candidatos (Figura 4.3b): [s2,s1]- 0.75, [s2,s11] - 0.74, [s2,s10]-0.77, [s2,s4]- 0.80 e [s2,s7]- 0.83, sendo que o melhor caminho escolhido (Algoritmo 3 - linha 8-10) é S11. Portanto o sensor S2 envia mensagem para S11 (Algoritmo 3 - linha 10,14). Quando S11 recebe a mensagem ele verifica se é o destino, como ele não o é, ele é expandido, e os seguintes caminhos candidatos são descobertos (Figura 4.3c): [s2,s1]- 0.75, [s2,s11,s13] - 0.744, [s2,s10]-0.77, [s2,s4]- 0.80 e [s2,s7]- 0.83, onde [s2,s11,s13] é o melhor caminho.

Por isto S11 envia a mensagem para S13 que passa a ser o sensor atual, que ao receber a mensagem se expande e passa a ter os seguintes caminhos candidatos (Figura 4.4a): [s2,s1]-

0.75, $[s2,s11,s13,s3]$ - 0.748, $[s2,s10]$ -0.77, $[s2,s4]$ - 0.80 e $[s2,s7]$ - 0.83, onde $[s2,s11,s13,s3]$ é o melhor caminho. Dessa forma, S13 envia a mensagem para S3 que ao receber verifica se ele é o destino (Algoritmo 2 - linha 6). Neste caso, o S3 é o destino. Portanto ele verifica se é o destino final do roteamento ou simplesmente destino de um re-roteamento (Algoritmo 2 - linha 7). Como existe pacote de mensagem, ele é o destino de um re-roteamento S3 identifica o pacote de mensagem que foi lhe enviado com os caminhos candidatos (Figura 4.4b) que são $[s1,s3]$ - 0.77, $[s1,s2,s11]$ - 0.82, $[s1,s2,s10]$ -0.81, $[s1,s2,s4]$ - 0.78 e $[s1,s2,s7]$ - 0.79 e continua fazendo sua rota expandindo o sensor atual S3 (Figura 4.4c).

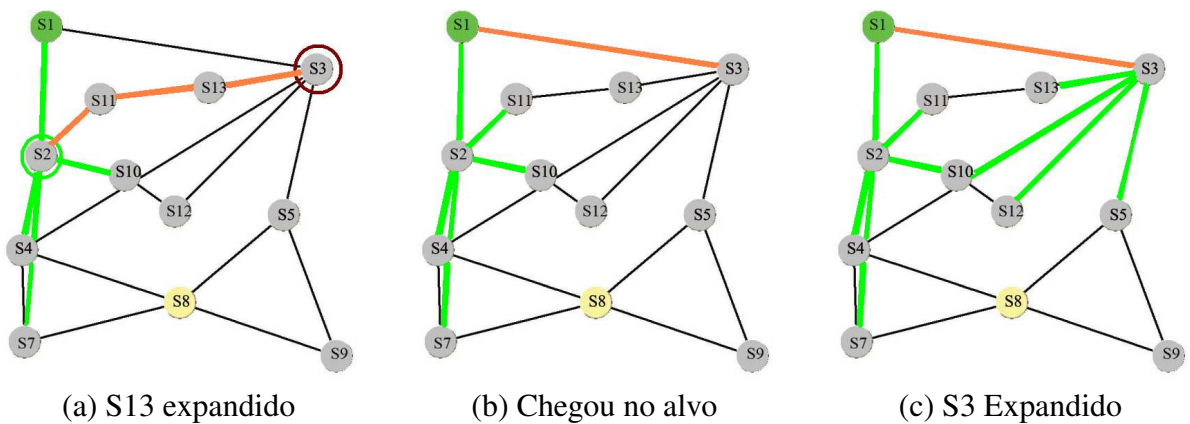


Figura 4.4: Rede de sensores

Este processo se repete até alcançar a estação base. Observe que para $N=2$ pode haver vários re-roteamentos dentro da rede até que a rota solicitada seja estabelecida. Já para $N=1$ não haverá re-roteamento em nenhum momento.

Os nós só conhecem seus vizinhos e conhecem a posição da EB. Portanto, no início do roteamento o sensor S2 deixou de ser o melhor caminho, que passou a ser o S3. Como S3 não é vizinho de S2, o único modo de enviar a mensagem que estava no S2 para o S3 seria fazer um roteamento do S2 até o S3. No entanto para calcular os custos de caminhos, é necessário conhecer a posição do sensor origem e do sensor destino, sendo que S2 não é a EB. Ele é simplesmente um sensor intermediário como o S3. Como o sensor atual é S2 então sua posição é conhecida, e a posição do S3 está no caminho candidato que está na mensagem recebida por S2. Portanto, todo caminho candidato é composto pelo último nó e a posição do mesmo, facilitando assim o cálculo do custo quando houver re-roteamento.

4.5 Arquitetura do Nó Sensor

A representação do nó sensor com AD* utilizado neste trabalho pode ser observada na Figura 4.5. A arquitetura nó é composta basicamente por:

- Gerenciador de mensagem: é responsável por receber e/ou enviar mensagens. Tem como tarefa reconhecer mensagens enviadas para o sensor e decodificar a mesma. Para o envio de mensagens, ele codifica os dados em um formato reconhecido pelos outros sensores da rede. Depois de recebida a mensagem e feita a devida decodificação, então esses dados são enviados para o kernel.
- Gerenciador de energia: é responsável por atualizar e fornecer informações sobre a energia do sensor. A cada movimentação de mensagem, tanto para receber como para enviar é simulado o consumo da energia com base no tamanho da mensagem.
- Gerenciador de Configuração: tem a função de emitir uma mensagem broadcast para detectar quem são os outros sensores que estão ao seu alcance. Os sensores que respondem são adicionados à lista como seu vizinho. Outra função deste gerenciador é emitir de tempos em tempos uma mensagem diretamente para os vizinhos que estão na lista, solicitando informações como sua posição e capacidade de energia.
- Gerenciador de caminhos candidatos: tem como função expandir o sensor, o acrescentar na lista dos outros já existentes, e aplicar a poda (*beam-search*).
- Lista de vizinhos e seus status: armazena quem são seus vizinhos e qual é o status da energia de cada um deles e a posição em que estes estão localizados.
- Kernel: é responsável por tomar as decisões sobre o que fazer com os dados recebidos do gerenciador de mensagem. O kernel recebe a lista de vizinhos e envia para o gerenciador de caminhos candidatos processá-la. Este retorna a lista dos candidatos. Em seguida, o kernel seleciona o melhor dentre a lista de candidatos e prepara os dados enviando-os para o gerenciador de mensagem encaminhar a mensagem para o sensor escolhido.

Neste trabalho só foi considerado o consumo de energia com a troca de mensagens para roteamento entre os nós. Portanto, não é considerado o consumo de energia para atualizações de informações referente aos sensores vizinhos e nem o consumo de energia com o processamento feito pelo nó sensor. Também não faz parte do escopo deste trabalho as fases de manutenção da

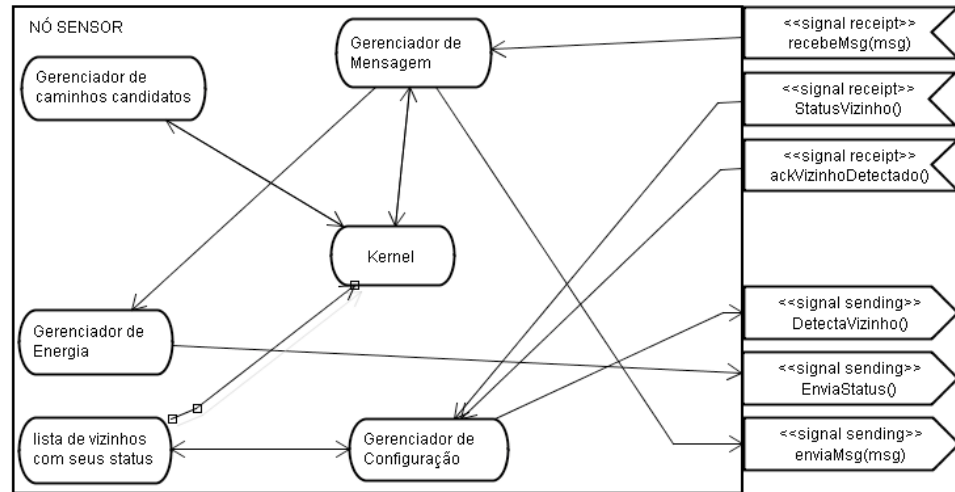


Figura 4.5: Modelo do Sensor

rota, nem a atualização de informação. Somente está sendo considerada, neste trabalho, apenas a fase de descoberta da rota.

4.6 Considerações Finais

Neste capítulo foram descritos os princípios do AD*, explicando o modelo de energia utilizado e o funcionamento deste algoritmo. Para exemplificar seu funcionamento foi utilizado uma pequena rede de sensor, por meio da qual foi explicado a estratégia utilizada pelo algoritmo para o estabelecimento da rota solicitada. No próximo capítulo será mostrado e avaliado os resultados dos experimentos realizados com o AD* e outros algoritmos conhecidos na literatura que abordam a mesma problemática.

Capítulo 5

Experimentos

5.1 Considerações Iniciais

Neste capítulo é descrito a metodologia utilizada para avaliar o AD* e alguns algoritmos conhecidos na literatura que foram escolhidos para comparação. São apresentados também os resultados obtidos por estes experimentos seguidos pelos comentários pertinentes sobre estes resultados.

5.2 Metodologia de Avaliação

Para avaliar o desempenho do algoritmo de roteamento AD*, utilizamos várias medidas de desempenho conforme apresentado em [24]. Todos os experimentos foram realizados considerando a energia consumida para a descoberta das rotas e o consumo com tráfego de dados. Antes de descrever as medidas de avaliação, é necessário explicar algumas das notações utilizadas nas fórmulas:

- n é o número total de nós existente na rede;
- $Node_i$ é o nó i , por exemplo, se a rede possuir 100 nós o primeiro nó é $Node_1$ e o último é $Node_{100}$;
- $Node_i.Energy$ é a energia do nó i , por exemplo, a energia do nó 50 é denotada por $Node_{50}.Energy$;
- $Node_i.AmountMsg$ é a quantidade de mensagens que este nó trocou com seus vizinhos, semelhante ao item anterior da energia;

- x é o número de rotas estabelecidas;
- $Route_x$ é a rota x , portanto para saber o caminho da primeira rota estabelecida utiliza-se $Route_1$;
- $Count(Route_x)$ é a quantidade de nós que compõem a $Route_x$;
- $COUNT_{m=1}^{\infty} Node_m$ é a quantidade de nós que trocaram mensagem durante o estabelecimento das rotas;
- $Fail(Node_i)$ é uma função que retorna 0 se o $Node_i$ não falhar e 1 se o $Node_i$ falhar;

As medidas de avaliação utilizadas para medir o desempenho do AD* são [24]:

- Média de energia restante na rede (AvgRE): com esta medida é possível avaliar o tempo de vida da rede, pois geralmente quanto maior for a média de energia restante, maior será o tempo de vida da rede. A fórmula utilizada é $AvgRE = \frac{\sum_{i=1}^n Node_i.Energy}{n}$.
- Desvio Padrão (StdDev): é desejável que o desvio padrão da energia restante dos nós seja baixo. Quanto menor for o desvio padrão da energia restante, maior será a eficiência do algoritmo para manter o equilíbrio no consumo de energia da rede. A fórmula utilizada é $StdDev = \sqrt{\frac{\sum_{i=1}^n (Node_i.Energy - AvgRE)^2}{n}}$.
- Mínima Energia Restante (MinRE): esta medida mostra a energia restante do sensor que tem a menor energia de toda a rede. Quanto mais próxima for da média de energia restante, maior será o equilíbrio de consumo de energia. A fórmula utilizada é $MinRe = \min_{i=1}^n Node_i.Energy$.
- Média de Energia Consumida (AvgEpa): é a média da energia consumida pelos agentes sensores para estabelecerem a rota solicitada. Quanto menor for a energia consumida, maior será a eficiência do algoritmo em se tratando de economia de energia. A fórmula utilizada é $AvgEpa = 0,5 - AvgRE$, onde 0,5 é a energia inicial do nó.
- Média de Mensagens (AvgMSG): número médio de mensagens trocadas entre os agentes sensores para o estabelecimento das rotas. A quantidade de mensagens trocadas entre os agentes sensores tem impacto direto sobre o consumo de energia. A fórmula utilizada é $AvgMsg = \frac{\sum_{i=1}^n Node_i.AmountMsg}{COUNT_{m=1}^{\infty} Node_m}$.

- Média de Nós da Rota (AvgHop): é o número médio de nós que compoem as rotas estabelecidas. A fórmula utilizada é $AvgHop = \frac{\sum_{x=1}^k Count(Route_x)}{k}$.
- Sensores que Falharam (SensorFail): é o número de nós que falharam por falta de energia durante o estabelecimento das rotas ou no tráfego de mensagens de dados. É desejável que a quantidade de sensores falhados seja a menor possível. A fórmula utilizada é $SensorFail = \sum_{i=1}^n Fail(Node_i)$.
- Número de Rotas até a queda da rede (NumRoute): é o número de rotas que foi possível estabelecer enquanto se consegue alcançar os sensores alvo. Quanto maior for o número de rotas possível de estabelecer, maior será a eficácia do algoritmo de roteamento. A fórmula utilizada é $NumRoute = COUNT_{y=1}^{\infty} Route_y$.

Para avaliar a performance do AD* foram escolhidos alguns algoritmos encontrados na literatura. A escolha destes algoritmos ocorreu em função da simplicidade para sua implementação, facilidade em extrair as medidas dos experimentos e principalmente pela semelhança que esses possuem com AD*, pois ambos utilizam como estratégia a energia restante e/ou a distância entres os nós. Portanto, os resultados dos experimentos com o algoritmo AD* são comparados com os resultados de outros quatro algoritmos:

- Estocástico [61] [24]: quando o agente sensor recebe uma mensagem ele escolhe aleatoriamente para qual de seus vizinhos enviar a mensagem recebida.
- Baseado no Máximo de Energia (MaxEnergy) [33] [24]: o agente sensor envia a mensagem para o vizinho que tem mais energia. Neste algoritmo a rota sempre irá perseguir os sensores com maior energia restante.
- Baseado em Gradiente Estocástico (GBRE) [61] [2]: No algoritmo baseado em gradiente o sensor memoriza o número de pulos até a estação base. Quando um nó precisa enviar alguma informação, procura minimizar o número de pulos até a estação base. O gradiente é a diferença de pulos que o nó memorizou com os pulos memorizados por seu vizinho. O vizinho que é escolhido para rotear a mensagem é aquele que proporcionar o maior gradiente. Quando dois ou mais vizinhos possuem o mesmo gradiente, então a escolha entre eles é feita aleatoriamente.
- Baseado em Gradiente Máximo de Energia (GBRMaxEnergy) [61] [2]: trabalha de maneira semelhante ao anterior, porém quando dois ou mais vizinhos possuem o

mesmo gradiente, a mensagem é roteada para aquele vizinho que possuir maior energia disponível.

5.3 Resultados

O cenário para os experimentos simula uma RSSF com 1000 sensores (Figura 5.1) homogêneos espalhados de modo irregular em uma área de 1053m x 714m, cuja rota deve ser estabelecida da estação base S1 até os sensores-alvo (S119, S84, S24, S17, S112, S153, S75, S135, S136, S148, S73, S234, S108, S156, S225, S133, S143, S134, S6, S230). Estes sensores se alternam para cada rota estabelecida. Todos os algoritmos foram executados sobre este mesmo cenário.

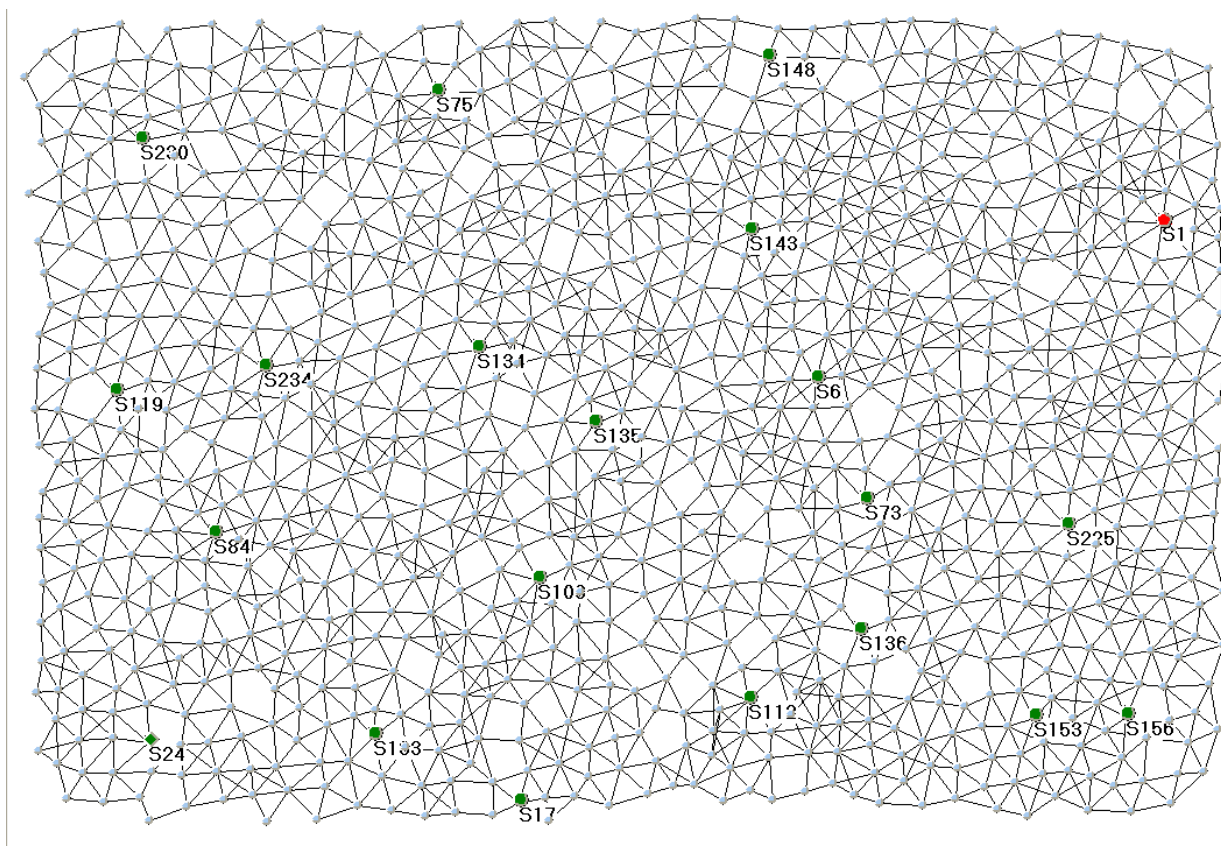


Figura 5.1: Rede de sensores com 1000 nós

As medidas dos experimentos foram extraídas após a geração de 100 rotas, 200 rotas, 300 rotas e assim sucessivamente até completar 1000 rotas. Depois que uma rota é estabelecida, são trafegadas por ela 100 mensagens, com 64 bits cada uma, para simular o tráfego de dados.

Portanto, quando estamos nos referindo a dados de 100 rotas estabelecidas, estamos também considerando o tráfego de 10000 mensagens de 64 bits cada, totalizando 6.4Mbits.

No Algoritmo 4 é possível visualizar o pseudo-código do algoritmo utilizado para simular o tráfego de dados entre os sensores depois de estabelecida a rota. A Estação Base envia 100 mensagens de 64bits por cada rota estabelecida (Algoritmo 4 - Linha 1-7). O sensor próximo à EB que recebe a mensagem de dados simula o consumo de energia (Algoritmo 4 - Linha 12) para o recebimento. Se o sensor que recebeu a mensagem é o destino (Algoritmo 4 - Linha 13-15) então a mensagem alcançou o destino. Porém, se o sensor que recebeu a mensagem de dados não é o destino, então este envia para o vizinho que faz parte da rota e simula o consumo da energia que consumiu com a transmissão desta mensagem para o vizinho (Algoritmo 4 - Linha 16-19). Este processo se repete para cada mensagem de dados até alcançar o destino.

Algoritmo 4 Pseudo-código para simular tráfego de dados

```

1: startTrafficData( route ) // eb é a estação base
2:  $i \leftarrow 0$ ;
3: while  $i < 100$  do
4:    $msg \leftarrow route + 64$ 
5:   eb.trafficData(msg)
6:    $i \leftarrow i + 1$ 
7: end while
8:
9: trafficData( msg )
10:  $route \leftarrow msg[0]$ 
11:  $size \leftarrow msg[1]$ 
12: consumptionReceive(size)
13: if empty(next) then
14:   STOP;
15: end if
16:  $next \leftarrow route[myself + 1]$ 
17:  $msg \leftarrow route + size$ 
18: send(msg,next)
19: consumptionSend(size)

```

Com o algoritmo Estocástico não foi possível extrair as medidas dos experimentos, já que com apenas 50 rotas estabelecidas e a estação base (S1), e alguns de seus sensores vizinhos ficaram completamente isolados do restante da rede devido à queda de vários sensores por falta de energia. Portanto, nos gráficos que mostraremos não foi possível incluir dados deste algoritmo devido a seu desempenho extremamente ruim.

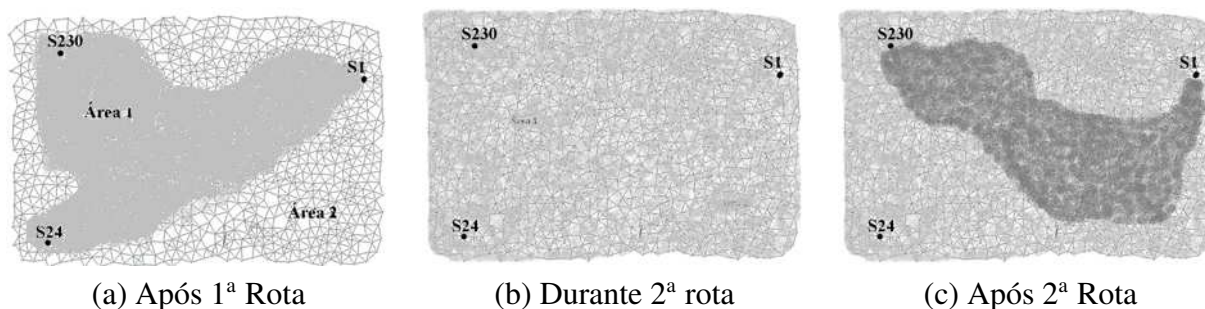


Figura 5.2: Rede de sensores utilizando o MAXEnergy para roteamento

Com o algoritmo baseado no Máximo de Energia obtivemos um problema muito semelhante ao Estocástico. Com 70 rotas estabelecidas a energia da rede estava degradada completamente, não sendo mais possível a estação base alcançar os sensores-alvo. Na rede da Figura 5.2 a primeira rota estabelecida foi de S1 até S24, sendo que a área mais escura da figura (Figura 5.2 - a "Área 1") denota os sensores visitados para o estabelecimento desta rota. A 2ª rota foi estabelecida de S1 até S230. É possível observar que na Área 2 onde não houve consumo de energia para estabelecer a 1ª rota, portanto o roteamento de S1 iniciará pelos sensores com mais energia. Ocorre que a Estação Base deseja uma rota para um nó que está dentro de uma área (Área 1) que já teve energia consumida parcialmente pelo estabelecimento da primeira rota e o tráfego de dados. O que ocorre é que o algoritmo MAXEnergy permanece consumindo a energia da área (Área 2) com mais energia até que a energia chegue ao nível dos sensores (Figura 5.2 - b) que sofreram desgaste na primeira rota (Área 1), para então conseguir alcançar o sensor S230 (Figura 5.2 - c).

Nos experimentos com o $AD^*(N \rightarrow 1)$ ¹ (Tabela 5.1) vemos que não houve nenhuma falha de sensores durante o estabelecimento das rotas e tráfego de dados. O consumo médio de energia para este algoritmo se mostrou bastante eficiente em relação aos demais algoritmos comparados. Já com o estabelecimento de 1000 rotas e o tráfego de 6.4Mbit de dados, a energia média consumida foi de apenas 0,0291J. Outra característica interessante é o fato de que o

¹N é o número de caminhos candidatos. $N \rightarrow 1$ significa que o BS efetuou a poda deixando somente o melhor caminho candidato. $N \rightarrow 2$ significa que o BS efetuou a poda deixando os dois melhores caminhos candidatos.

Tabela 5.1: Dados dos Experimentos com o AD*(N->1)

Rotas	AvgRE	StdDev	MinRE	AvgEpa	AvgAME	AvgHop	SensorFail
100	0,4974	0,0036	0,4736	0,0026	23,1900	23,1900	0,0000
200	0,4947	0,0063	0,4573	0,0053	23,7750	23,7750	0,0000
300	0,4919	0,0088	0,4419	0,0081	24,2767	24,2767	0,0000
400	0,4890	0,0113	0,4291	0,0110	24,6950	24,6950	0,0000
500	0,4860	0,0137	0,4176	0,0140	25,0180	25,0180	0,0000
600	0,4831	0,0161	0,4066	0,0169	25,2567	25,2567	0,0000
700	0,4801	0,0187	0,3928	0,0199	25,4843	25,4843	0,0000
800	0,4771	0,0212	0,3803	0,0229	25,6438	25,6438	0,0000
900	0,4740	0,0237	0,3689	0,0260	25,8011	25,8011	0,0000
1000	0,4709	0,0262	0,3575	0,0291	25,9670	25,9670	0,0000

sensor da rede que tem a menor energia estar com 0.3575J, sendo que a média de energia restante foi de 0.4709, mostrando assim que houve um equilíbrio significativo no consumo. O número médio de mensagens trocadas entre os sensores para o estabelecimento das rotas foi igual ao número médio de saltos para os caminhos estabelecidos, mostrando deste modo que as trocas de mensagens para o estabelecimento das rotas está bem otimizada. É possível observar que o valor do desvio padrão (AvgDev) da energia restante mesmo aumentando a quantidade de rotas estabelecidas, se manteve entre 0.0024 à 0.0027. Por exemplo, entre 400 e 500 rotas a variação é de 0,0024 (0,0137-0,0113).

Com o AD*(N->2) (Tabela 5.2) é possível observar que não houve nenhuma falha de sensores durante o estabelecimento das rotas e tráfego de dados. O consumo médio de energia para este algoritmo não se mostrou tão eficiente quanto o anterior, mas conseguiu uma média de consumo aceitável, já que com o estabelecimento de 1000 rotas e o tráfego de dados de 6.4Mbit, a energia média consumida foi de apenas 0,0311J. Porém o sensor da rede que tem a menor energia possui 0.3460J, sendo que a média de energia restante foi de 0.4689, mostrando assim que houve um certo equilíbrio no consumo de energia. Porém, o desvio padrão da energia restante foi maior que AD*(N->1) e o consumo de energia também foi maior. O número médio de mensagens trocadas entre os sensores para o estabelecimento das rotas foi maior que o número médio de saltos para os caminhos estabelecidos, mostrando deste modo que as trocas de mensagens para o estabelecimento das rotas tiveram impacto direto sobre o consumo de

Tabela 5.2: Dados dos Experimentos com o AD*(N->2)

Rotas	AvgRE	StdDev	MinRE	AvgEpa	AvgAME	AvgHop	SensorFail
100	0,4971	0,0041	0,4697	0,0029	60,0700	34,3800	0,0000
200	0,4942	0,0072	0,4527	0,0058	57,1500	35,2850	0,0000
300	0,4911	0,0098	0,4374	0,0089	54,9800	35,5433	0,0000
400	0,4880	0,0123	0,4243	0,0120	54,2925	36,1400	0,0000
500	0,4849	0,0149	0,4109	0,0151	53,6640	36,6200	0,0000
600	0,4817	0,0176	0,3968	0,0183	53,0000	36,8200	0,0000
700	0,4785	0,0202	0,3849	0,0215	52,8914	37,1886	0,0000
800	0,4753	0,0228	0,3720	0,0247	52,6450	37,3162	0,0000
900	0,4721	0,0255	0,3591	0,0279	52,3656	37,4367	0,0000
1000	0,4689	0,0282	0,3460	0,0311	52,1180	37,5270	0,0000

energia. Observe também que o acréscimo no desvio padrão (AvgDev) da energia restante com o aumento da quantidade de rotas estabelecidas obteve uma variação de 0.0026 à 0.0031. Por exemplo, entre 700 e 800 rotas a variação é de 0,0026 (0,0228-0,0202).

Tabela 5.3: Dados dos Experimentos com o GBRE

Rotas	AvgRE	StdDev	MinRE	AvgEpa	AvgAME	AvgHop	SensorFail
100	0,4973	0,0066	0,4380	0,0027	20,3500	20,3500	0,0000
200	0,4947	0,0134	0,3635	0,0053	20,3500	20,3500	0,0000
300	0,4922	0,0195	0,3074	0,0078	20,3500	20,3500	0,0000
400	0,4896	0,0254	0,2467	0,0104	20,3500	20,3500	0,0000
500	0,4870	0,0321	0,1731	0,0130	20,3500	20,3500	0,0000
600	0,4845	0,0380	0,1177	0,0155	20,3500	20,3500	0,0000
700	0,4819	0,0446	0,0474	0,0181	20,3500	20,3500	0,0000
800	0,4793	0,0507	0,0355	0,0207	20,3500	20,3500	0,0000
900	0,4767	0,0523	0,0000	0,0233	20,3700	20,3700	2,0000
1000	0,4742	0,0572	0,0000	0,0258	20,3860	20,3860	2,0000

Os experimentos realizados com GBRE (Tabela 5.3) nos mostraram sensores falhados ao se estabelecer 900 e 1000 rotas com tráfego de dados. Contudo, o consumo médio de energia

se mostrou extremamente eficiente, sendo que com o estabelecimento de 1000 rotas e o tráfego de dados, a energia média consumida foi de apenas 0.258J. Porém, o desvio padrão (Figura 5.8) da energia restante foi o mais alto de todos os algoritmos que utilizamos para os experimentos, nos mostrando que esta técnica provoca um desequilíbrio no consumo da energia da rede, provocando áreas sem energia, enquanto outras permanecem com bastante energia. O número médio de mensagens trocadas entre os agentes sensores coincidiram exatamente com a quantidade média de pulso para as rotas estabelecidas. No entanto, a variação do desvio padrão (StdDev) mediante o aumento da quantidade de rotas estabelecidas ficou entre 0.0049 e 0.0068. Por exemplo, entre 100 e 200 rotas a variação é de 0,0068 (0,0134-0,0066).

Tabela 5.4: Dados dos Experimentos com o GBRMaxEnergy

Rotas	AvgRE	StdDev	MinRE	AvgEpa	AvgAME	AvgHop	SensorFail
100	0,4973	0,0054	0,4495	0,0027	20,3500	20,3500	0,0000
200	0,4948	0,0107	0,3991	0,0052	20,3500	20,3500	0,0000
300	0,4922	0,0159	0,3454	0,0078	20,3500	20,3500	0,0000
400	0,4897	0,0212	0,2949	0,0103	20,3500	20,3500	0,0000
500	0,4871	0,0264	0,2494	0,0129	20,3500	20,3500	0,0000
600	0,4846	0,0316	0,2022	0,0154	20,3500	20,3500	0,0000
700	0,4820	0,0368	0,1518	0,0180	20,3500	20,3500	0,0000
800	0,4795	0,0421	0,1046	0,0205	20,3500	20,3500	0,0000
900	0,4769	0,0473	0,0542	0,0231	20,3500	20,3500	0,0000
1000	0,4744	0,0525	0,0054	0,0256	20,3500	20,3500	0,0000

O algoritmo GBRMaxEnergy (Tabela 5.4), semelhante ao que acontece com GBRE, mostrou bastante eficiência em relação ao consumo médio de energia após o estabelecimento de 1000 rotas com o tráfego de 6.4Mbit de dados. A energia média consumida foi de 0.0256, possuindo portanto o menor consumo de energia em relação aos outros algoritmos utilizados neste trabalho. Contudo, diferentemente do que aconteceu com o GBRE, com o GBRMaxEnergy não ocorreu queda de sensores por falta de energia, mas o sensor que ficou com a menor energia possui 0.0054J. Conseqüentemente, apesar de não haver falha nos sensores, fica claro que alguns sensores ficaram com a energia restante muito baixa. Verificando o desvio padrão da energia restante, podemos concluir que apesar do resultado ter sido um pouco melhor que o GBRE, ainda ocorre um desequilíbrio bastante significativo com o consumo de

energia, embora menor que o GBRE. Contudo, também observamos que o número médio de mensagens trocadas foi exatamente igual ao número médio de pulos para as rotas estabelecidas. A variação que ocorreu com o desvio padrão (stdDev) com o aumento na quantidade de rotas estabelecidas ficou entre 0.0052J e 0.0053J. Por exemplo, entre 300 e 400 rotas a variação é de 0,0053 (0,0212-0,0159).

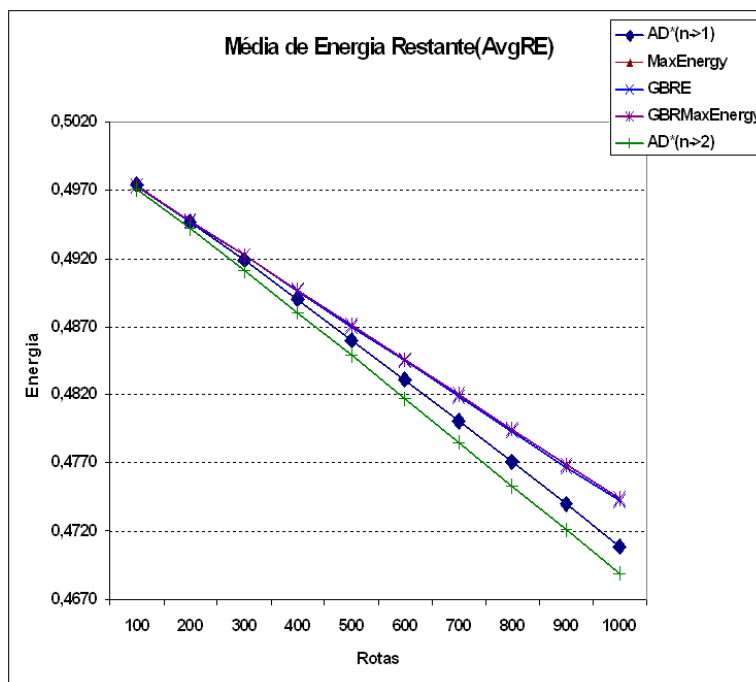


Figura 5.3: Média de Energia Restante

Observa-se (Figura 5.3) que após traçar 1000 rotas e trafegar 6.4Mbit de dados, a média restante de energia em ordem decrescente é: GBRMaxEnergy de 0,4744J, o GBRE de 0,4742J, o AD*(N->1) de 0,4709J e o AD*(N->2) de 0,4689J.

A energia consumida (Figura 5.4) segue o mesmo modelo da energia restante (Figura 5.3) de modo que a energia consumida+energia restante obrigatoriamente resulta em 0.5J. É possível observar que os algoritmos GBRE e o GBRMaxEnergy possuem o mesma média de energia consumida, devido ao fato da estratégia destes algoritmos considerarem o gradiente como o fator mais preponderante na escolha do vizinho.

É possível observar (Figura 5.5) que o GBRE teve queda de 2 sensores (SensorFail=2), e o GBRMaxEnergia ficou com alguns sensores com pouca energia restante, o que não aconteceu com os algoritmos AD*(N->1) e AD*(N->2) que conseguiram manter todos os sensores com elevada carga de energia mantendo assim o consumo equilibrado na rede.

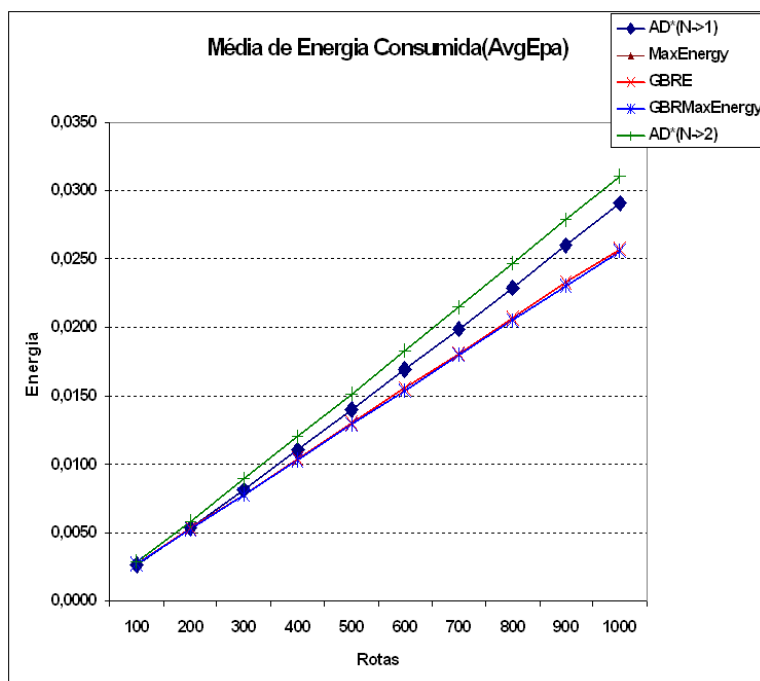


Figura 5.4: Média de Energia Consumida

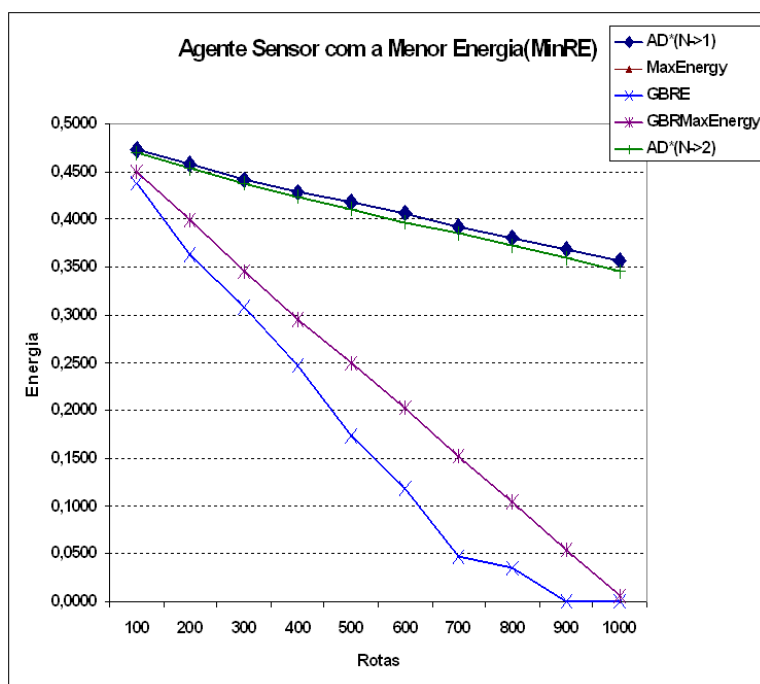


Figura 5.5: Sensor com Menor Energia

O AD*(N->2) precisou efetuar um número maior de troca de mensagens (Figura 5.6) em relação ao AD*(N->1) para a descoberta de rotas, comprovando deste modo o que dissemos anteriormente: a quantidade de trocas de mensagens entre os agentes sensores possui impacto

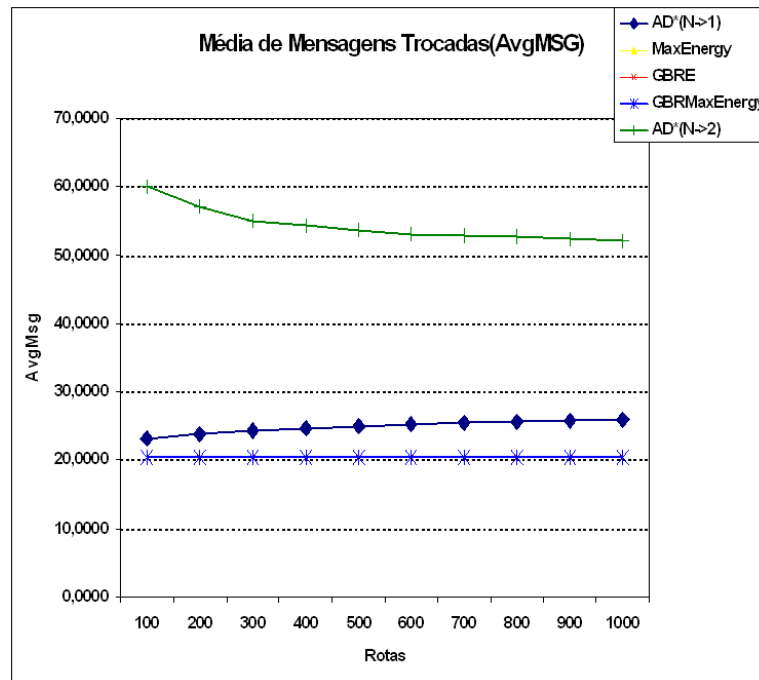


Figura 5.6: Média de Mensagens Trocadas Entre os Nós

direto sobre o consumo de energia da rede. No $AD^*(N->1)$ a quantidade de troca de mensagens oscila proporcionalmente ao número de nós que compoem a rota estabelecida. Já com o $AD^*(N->2)$ a quantidade de mensagens é proporcional ao número de nós que compoem a rota estabelecida acrescido dos re-roteamentos que foram necessários fazer.

Com o algoritmo MaxEnergy a primeira rota estabelecida foi composta por 450 nós, já com o Estocástico a quantidade de nós que formou a primeira rota estabelecida foi de 780. Isto nos mostra um dos motivos desses dois algoritmos consumirem tanta energia para o tráfego de mensagens. Contudo, observa-se na Figura 5.7 que os algoritmos GBRE, GBRMaxEnergy, $AD^*(N->1)$ tiveram uma excelente média de nós que formaram as rotas estabelecidas. O $AD^*(N->2)$ obteve uma média de nós aceitável para a rota, porém bem maior que $AD^*(N->1)$. Esta diferença se deve pelo fato do $AD^*(N->2)$ consumir energia com os re-roteamentos. No $AD^*(N->2)$ durante a descoberta da rota, quando o sensor atual deixa de ser a melhor rota, é necessário retransmitir as informações do roteamento que está sendo feito para o sensor que passou a ser a melhor rota, e este re-roteamento pode ocorrer diversas vezes até se conseguir estabelecer a rota da estação base até o sensor alvo, ocasionando deste modo um consumo maior de energia em comparação com o $AD^*(N->1)$.

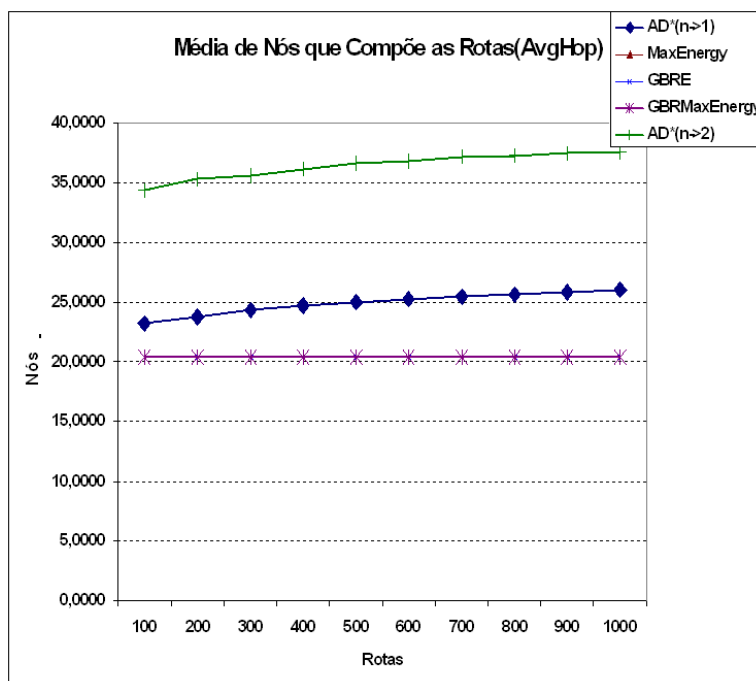


Figura 5.7: Média de Nós que Compõem a Rota

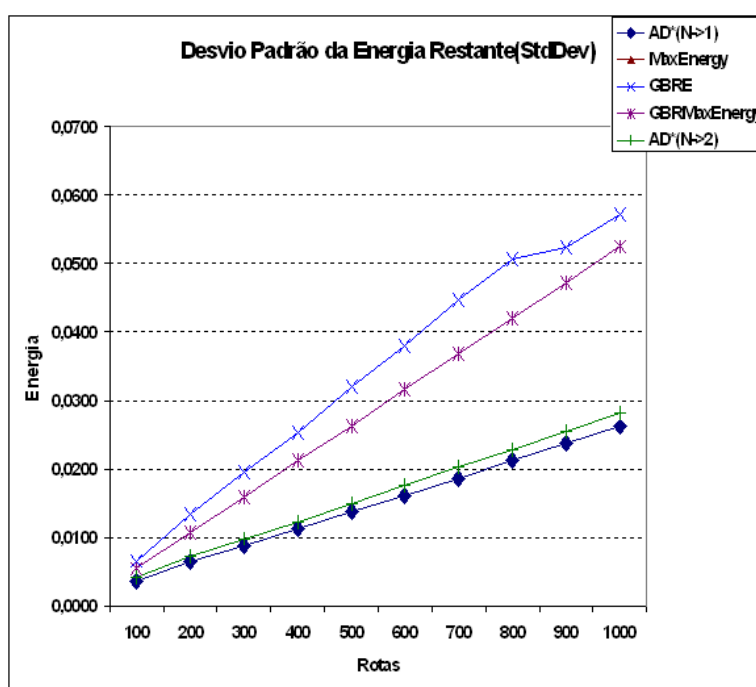


Figura 5.8: Desvio Padrão da Energia Restante

Conforme demonstrado pela Figura 5.8, podemos afirmar que o algoritmo que possui o maior equilíbrio no consumo de energia para o estabelecimento das rotas é o AD*(N->1) seguido pelo AD*(N->2), GBRMaxEnergy, GBRE.

O $AD^*(N->1)$ além de manter um equilíbrio de consumo na rede, garante que a energia consumida é bem menor em relação ao MAXEnergy. Com o GBRE e o GBRMaxEnergy mesmo consumindo pouca energia para o estabelecimento de rotas, estes dois algoritmos tendem a criar um vale de sensores sem energia, não havendo um consumo equilibrado, podendo ocasionar áreas densamente falhas na rede de sensores, impedindo assim que determinados sensores consigam alcançar a estação base.

Tabela 5.5: Número de Rotas até a queda da rede (NumRoute)

	Estocástico	MaxEnergy	GBRE	GBRMaxEnergy	AD(N->1)	AD(N->2)
Rotas	50	70	3925	3940	5157	4419

O número de rotas com tráfego de dados que cada sensor conseguiu estabelecer antes que a energia da rede ficasse degradada ² pode ser observado na Tabela 5.5. O algoritmo Estocástico e o MaxEnergy tiveram rapidamente a estação base isolada do restante da rede devido à falha dos sensores por falta de energia. O GBRE conseguiu estabelecer 3925 rotas antes que todos os alvos ficassem totalmente isolados da Estação Base. O GBRMaxEnergy conseguiu estabelecer 3940 rotas antes da queda completa dos sensores em volta da Estação Base. Com o $AD(N->1)$ foi possível alcançar o melhor resultado com o estabelecimento de 5157 rotas antes que a rede se degradasse completamente, sendo este o melhor resultado dentre todos os algoritmos.

Em algum momento foi possível observar que o $AD(N->1)$ consumia a energia de uma pequena área até conseguir baixar o nível para continuar estabelecendo a rota. Isto ocorreu porque na heurística utilizada pelo AD^* o peso da energia restante era muito maior do que aquele da distância. Contudo, o peso utilizado foi considerado ideal para alcançar um equilíbrio de energia da rede, pois, conforme se aumenta o peso da distância maior será o desvio padrão, mostrando deste modo um maior desequilíbrio no consumo de energia.

Observando o $AD(N->2)$, verifica-se que embora ele tenha conseguido o segundo melhor resultado quanto ao número de rotas estabelecidas antes que a energia da rede ficasse degradada, ele apresentou um resultado bem inferior ao do $AD(N->1)$.

²Considera como queda da rede (energia degradada) quando a EB não consegue alcançar nenhum dos sensores-alvo

Para avaliar estatisticamente os algoritmos, aplicamos o Teste de Friedman [14]. Este é um teste estatístico não-paramétrico desenvolvido pelo economista norte-americano Milton Friedman com o objetivo de detectar a magnitude das variações entre os grupos testados.

Para realizar o Teste de Friedman, os dados devem ser dispostos em uma tabela com n linhas (variáveis) e n colunas (algoritmos). Foram consideradas como variáveis as medidas AvgRE, StdDev, MinRe e os algoritmos comparados com o teste foram AD*(N->2), AD*(N->1), GBRE e GBRMaxEnergy.

Na Tabela 5.6 e 5.7 é ilustrado a aplicação do Teste de Friedman para o estabelecimento de 100 rotas com tráfego de dados. Foi escolhido 100 rotas apenas para exemplificar, pois o mesmo modelo é aplicado para 200, 300 e assim sucessivamente até 1000 rotas. É possível observar (Tabela 5.6) as medidas originais de cada variável para o respectivo algoritmo. A partir destas medidas originais (dados crus) é feito um ranqueamento (Tabela 5.7), onde se classifica como 1° o algoritmo que possui a melhor medida para a variável, sendo o 2° aquele com a segunda melhor medida, e assim sucessivamente. Uma vez classificados (ranqueados) os algoritmos para todas as variáveis, é calculado a média desta classificação para cada um dos algoritmos testados.

Tabela 5.6: Aplicação do Teste de Friedman (Medidas Originais)

	AD*(n-2)	AD*(n-1)	GBRE	GBRMaxEnergy
AvgRE	0,4971	0,4974	0,4973	0,4973
StdDev	0,0041	0,0036	0,0066	0,0054
MinRE	0,4697	0,4736	0,4380	0,4495

Tabela 5.7: Aplicação do Teste de Friedman (Medidas Classificadas)

	AD*(n-2)	AD*(n-1)	GBRE	GBRMaxEnergy
AvgRE	4	1	2,5	2,5
StdDev	2	1	4	3
MinRE	2	1	4	3
Média	2,7	1	3,5	2,8

Com 4 algoritmos e 3 variáveis, o F de distribuição se dá com $4-1=3$ e $(4-1) \times (3-1)=6$ graus de liberdade. Portanto, o valor crítico de $F(3,6)$ para um nível de significância de 5% é de 4.757, então rejeita-se algoritmos que fiquem com a média classificatória acima disto.

Após ter sido estabelecida a média do ranqueamento para cada um dos algoritmos é possível considerar que o algoritmo com a menor média de classificação é o algoritmo considerado com sendo aquele que alcançou o melhor resultado segundo o Teste de Friedman, e aquele que obtiver a maior média de classificação é considerado como sendo aquele com o pior resultado.

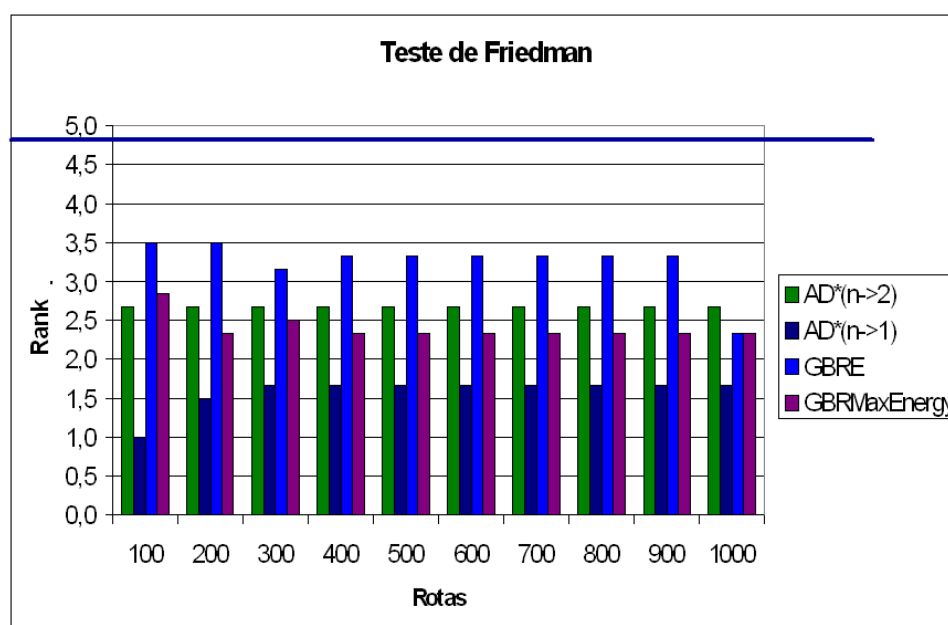


Figura 5.9: Resultado do Teste de Friedman

Após ter sido aplicado o Teste de Friedman, o resultado é demonstrado na Figura 5.9, com qual é possível notar que o algoritmo $AD^*(N \rightarrow 1)$ apresentou um melhor resultado classificatório em relação aos algoritmos GBRMaxEnergy, GBRE e $AD^*(N \rightarrow 2)$. Foi verificado também, que nenhum dos algoritmos ficou acima do valor crítico de distribuição não sendo possível fazer nenhuma distinção estatisticamente significante entre os algoritmos.

5.4 Considerações Finais

Neste capítulo foi apresentada a metodologia de avaliação utilizada para avaliar as técnicas estudadas e os resultados dos experimentos realizados foram analisados. Discutiu-se os

resultados para cada uma das técnicas considerando as métricas utilizadas para as comparações. Observou-se que o algoritmo AD* conseguiu alcançar um menor consumo médio de energia e foi também aquele que melhor equilíbrio conseguiu no consumo da energia. No próximo capítulo serão apresentadas as conclusões resultantes deste trabalho com algumas das possíveis direções futuras desta pesquisa.

Capítulo 6

Conclusões

As RSSF podem ser empregadas em uma ampla variedade de aplicações, tais como, monitoração de ambientes, campos de batalha, ciências biomédicas etc. Além disto, estas redes possuem um custo de implantação infinitamente menor do que aquele necessário em uma rede tradicional.

Neste trabalho foram analisadas algumas das dificuldades envolvidas quando se trabalha com redes de sensores. Focamos, principalmente, o consumo de energia que foi o objetivo principal deste trabalho ao propor um novo algoritmo de roteamento para RSSF baseado em Resolução Distribuída de Problemas, com o objetivo de manter o tempo de vida da rede o maior possível.

Embora o objetivo do trabalho tenha sido o estabelecimento das rotas sem estar preocupado com o tráfego de dados e a manutenção destas rotas, foi possível notar que novos mecanismos se fazem necessários em futuros trabalhos para melhorar o algoritmo. Contudo, para o estabelecimento das rotas considerando o balanceamento da energia disponível dos sensores, os resultados alcançados com o AD* se mostraram bastante satisfatórios.

Considera-se que ao se referir à RSSF, geralmente estamos falando de uma grande quantidade de nós interconectados uns aos outros, sem uma coordenação centralizada, onde cada nó só possui uma visão local do ambiente e a solução do problema se dá pela cooperação que estes nós têm uns com os outros. A nossa abordagem de utilizar uma técnica da Resolução Distribuída de Problemas para resolver o problema de roteamento em redes de sensores demonstrou ser bastante eficiente.

Foi possível também observar que a RDP tem sido amplamente utilizada para solucionar diversos problemas em redes de sensores e trabalhos como Contract-Net [63], DVMT [16]

[44], Multistage [11], etc., podem ser utilizados em problemas com algumas semelhanças ao roteamento em RSSF. Isso nos motivou a fazer uso de técnicas de IAD para resolver o problema de roteamento em RS alcançando significativos resultados com os experimentos.

Considerando os experimentos realizados, é possível concluir que o $AD^*(N->1)$ alcançou baixo consumo de energia para estabelecer os roteamentos, além do consumo balanceado de energia. O MaxEnergy embora mantenha o consumo balanceado, se mostrou extremamente ruim, consumindo muita energia desnecessariamente para o estabelecimento de novas rotas, mostrando um comportamento semelhante ao Estocástico.

O GBRE e o GBRMaxEnergy alcançaram um consumo médio razoável, porém provocaram áreas densamente desgastadas em determinados pontos da rede ocasionadas pelo não balanceamento do consumo. Isto resulta em áreas com muita energia enquanto outras permanecem sem energia. Isto ocasiona divisões na rede, dificultando o alcance em determinadas áreas que ficaram isoladas devido ao vale de sensores falhados por falta de energia.

O algoritmo estocástico mostrou-se extremamente fraco para roteamento neste tipo de rede, pois, antes de conseguir estabelecer 50 rotas deixou áreas sem energia de modo que não foi mais possível alcançar os sensores desejados.

Conforme os experimentos demonstraram, o $AD^*(N->1)$ se mostrou bastante eficiente para o estabelecimento de rotas em RSSF. Pretendemos continuar fazendo outros experimentos para buscar mais resultados que comprovem de maneira mais enfática os resultados alcançados até agora.

Pretendemos em trabalhos futuros, adicionar comparações com outros algoritmos de roteamento, tais como, DSR [34] [46], AODV [69] [54] [66] [23], NADV [42]. Estes algoritmos, semelhantes ao AD^* , buscam por meio de um gerenciamento eficiente da energia e o balanceamento do tráfego aumentar o tempo de vida dos nós, e conseqüentemente aumentar também o tempo de vida da rede. De qualquer forma, acreditamos que os resultados alcançados pelo AD^* , sejam semelhantes aos experimentos apresentados neste trabalho.

Considerando trabalhos futuros, é possível incluir mecanismos de alteração dinâmica de rota que permitam ao algoritmo alterar a rota descoberta quando a energia dos sensores diminuir devido à transmissão de dados. Desta forma, sensores com baixa energia podem solicitar a descoberta de uma nova rota para continuar trafegando as informações, procurando deste modo manter o equilíbrio da energia disponível dos nós.

Como geralmente as RSSF estão depositadas em ambientes hostis, podendo ocorrer falhas nos nós sensores em qualquer momento por diversos fatores, é necessário que a rede

seja tolerante a falhas. Esta é uma direção futura de pesquisa bastante interessante, sendo que mecanismos de tolerância a falhas poderiam ser adicionados ao AD* para que quando um determinado sensor da rota estabelecida falhar por algum motivo, os sensores vizinhos de onde ocorreu a falha cooperassem entre si para restabelecer aquele ponto da rota.

Outra futura direção para o AD*, seria aplicar o algoritmo em RSSF hierárquicas (também chamadas de "clusters"), onde um agrupamento de nós possui um líder ao qual estes nós se reportam e que coopera com os líderes dos outros agrupamentos para estabelecer o caminho até a estação base.

Trabalhos futuros também poderiam acrescentar mecanismos no AD* para trabalhar com redes de sensores móveis, onde os sensores podem se movimentar continuamente provocando falhas em rotas pré-estabelecidas, necessitando que esses nós executem operações necessárias para a recuperação do caminho.

Referências Bibliográficas

- [1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Comput. Networks*, 38(4):393–422, 2002.
- [2] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, 2004.
- [3] Ronald Ashri, Michael Luck, and Mark d’Inverno. On identifying and managing relationships in multi-agent systems. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 743–748. Morgan Kaufmann, 2003.
- [4] Avron Barr and Edward A. Feigenbaum. *The handbook of artificial intelligence, 4 vols.* Addison-Wesley, 1981.
- [5] Chuda Basnet, Guochun Tang, and Tadashi Yamaguchi. A beam search heuristic for multi-mode single resource constrained project scheduling. In *Working Papers*, 2001.
- [6] Nourredine Bensaïd and Philippe Mathieu. A hybrid architecture for hierarchical agents. In *International Conference on Computational Intelligence and Multimedia Applications, ICCIMA 97*, pages 91–95, GRIFFITH UNIVERSITY, Gold-Coast, Australia, February 1997.
- [7] David Braginsky and Deborah Estrin. Rumor routing algorithm for sensor networks. In *International Conference on Distributed Computing Systems (ICDCS-22)*, 2002.
- [8] Rodney A. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988]*, 2(1):14–23, 1986.
- [9] Rodney A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*

- (IJCAI-91), pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.
- [10] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*. Addison Wesley, 1985.
- [11] Susan E. Conry, Robert A. Meyer, and Victor Lesser. *Multistage negotiation in distributed planning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [12] Daniel D. Corkill and Victor Lesser. The use of meta-level control for coordination in a distributed problem solving network. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, August 1983.
- [13] Luke Demoracski and Dimiter R. Avresky. Performance analysis of fault-tolerant beacon vector routing for wireless sensor networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 40–44, New York, NY, USA, 2005. ACM.
- [14] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- [15] Edmund H. Durfee. *Distributed problem solving and planning*. MIT Press, Cambridge, MA, USA, 1999.
- [16] Edmund H. Durfee and Victor Lesser. *Using partial global plans to coordinate distributed problem solvers*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [17] Jeremy Elson and Deborah Estrin. Random, ephemeral transaction identifiers in dynamic sensor networks. In *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, page 459, Washington, DC, USA, 2001. IEEE Computer Society.
- [18] Cora Beatriz Excelente-toledo and Nicholas R. Jennings. The dynamic selection of coordination mechanisms. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):55–85, 2004.
- [19] Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24:159–182, 1998.

- [20] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. An agenda-based framework for multi-issue negotiation. *Artificial Intelligence*, 152(1):1–45, 2004.
- [21] Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [22] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng T. Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensor networks. In *NSDI '05*, 2005.
- [23] Mounir Frikha and Jamila Ben Slimane. Conception and simulation of energy-efficient aodv protocol in ad hoc networks. In *Mobility '06: Proceedings of the 3rd international conference on Mobile technology, applications & systems*, page 10, New York, NY, USA, 2006. ACM Press.
- [24] Long Gan, Jiming Liu, and Xiaolong Jin. Agent-based, energy efficient routing in sensor networks. *aamas*, 01:472–479, 2004.
- [25] Matthew E. Gaston and Marie desJardins. Agent-organized networks for dynamic team formation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 230–237, New York, NY, USA, 2005. ACM.
- [26] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.
- [27] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MOBICOM*, pages 174–185, Seattle, USA, 1999.
- [28] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, 1977.
- [29] Carl Hewitt and Jeff Inman. *DAI betwixt and between: from intelligent agents to open systems science*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.
- [30] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Communications ACM*, 47(6):41–46, 2004.

- [31] Jason Lester Hill. *System architecture for wireless sensor networks*. PhD thesis, 2003. Adviser-David E. Culler.
- [32] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *Principles and Practice of Constraint Programming*, pages 222–236, 1997.
- [33] Xiaoyan Hong, M. Gerla, Hanbiao Wang, and L. Clare. Load balanced, energy-aware communications for mars sensor networks. *IEEE Aerospace Conference Proceedings*, 3:3–1109 – 3–1115, March 2002.
- [34] Yih-Chun Hu and David B. Johnson. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 231–242, New York, NY, USA, 2000. ACM Press.
- [35] Michael N. Huhns and David M. Bridgeland. Multiagent truth maintenance. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1437–1445, 1991.
- [36] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.
- [37] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.
- [38] Nicholas R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.
- [39] I. Khemapech, I. Duncan, and A. Miller. A survey of wireless sensor networks technology. *Proceedings of the 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, June 2005.
- [40] Raymond Kurzweil. *The age of intelligent machines*. MIT Press, 1990.
- [41] YoungMin Kwon, Sameer Sundresh, Kirill Mechitov, and Gul Agha. Actornet: an actor platform for wireless sensor networks. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1297–1300, New York, NY, USA, 2006. ACM.

- [42] Seungjoon Lee, Bobby Bhattacharjee, and Suman Banerjee. Efficient geographic routing in multihop wireless networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 230–241, New York, NY, USA, 2005. ACM Press.
- [43] Victor Lesser. An overview of dai: Viewing distributed ai as distributed search. *Journal of Japanese Society for Artificial Intelligence*, 5(4), 1990.
- [44] Victor Lesser and Daniel D. Corkill. *The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1988.
- [45] Victor Lesser, Charles Ortiz, and Milind Tambe, editors. *Distributed Sensor Networks: a multiagent perspective*. Kluwer Publishing, 2003.
- [46] Yuhong Luo, Jianxin Wang, and Songqiao Chen. An energy-efficient dsr routing protocol based on mobility prediction. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 444–446, Washington, DC, USA, 2006. IEEE Computer Society.
- [47] Pattie Maes. Artificial life meets entertainment: lifelike autonomous agents. *Communications ACM*, 38(11):108–114, 1995.
- [48] Roger Mailler, Victor Lesser, and Bryan Horling. Cooperative negotiation for soft real-time distributed resource allocation. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 576–583, New York, NY, USA, 2003. ACM.
- [49] Benedita Malheiro and Eugenio Oliveira. Solving conflicting beliefs with a distributed belief revision approach. In *IBERAMIA-SBIA '00: Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI*, pages 146–155, London, UK, 2000. Springer-Verlag.
- [50] Raquel A. F. Mini, Daniel Ludovico Guidoni, and Max do V. Machado. Treedc: um algoritmo de coleta de dados ciente da energia para redes de sensores sem fio. *REIC*, 5(III), 2005.

- [51] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2003. ACM.
- [52] Fabíola Guerra Nakamura. Planejamento dinâmico para controle de cobertura e conectividade em redes de sensores sem fio planas. In *Master's thesis*, Belo Horizonte, MG, BR, 2003.
- [53] Cláudio L. de e Castro Maria Clícia Stelling de Pereira, Marluce R. e Amorim. Tutorial sobre redes de sensores. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*, Maio 2004.
- [54] Asad Amir Pirzada, Marius Portmann, and Jadwiga Indulska. Evaluation of multi-radio extensions to aodv for wireless mesh networks. In *MobiWac '06: Proceedings of the international workshop on Mobility management and wireless access*, pages 45–51, New York, NY, USA, 2006. ACM Press.
- [55] H. Qi, S. S. Iyengar, and K. Chakrabarty. Multi-resolution data integration using mobile agents in distributed sensor networks. In *IEEE Transactions on Systems, Man and Cybernetics (Part C): Applications and Reviews*, volume 31, pages 383–391, August 2001.
- [56] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
- [57] P. Rentala, R. Musunuri, S. Gandham, and U. Saxena. Survey on sensor networks. *International Conference on Mobile Computing and Networking*, 2001.
- [58] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw-Hill Higher Education, 1990.
- [59] Correia L. Vieira L. Ruiz, L. Arquiteturas para redes de sensores sem fio. In *22o Simpósio Brasileiro de Redes de Computadores*, pages 167–218, 2004.
- [60] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

- [61] C. Schurgers and M. Srivastava. Energy efficient routing in wireless sensor networks. in *the MILCOM Proceedings on Communications for Network-Centric Operations: Creating the information Force, McLean, VA, 2001.*
- [62] Zhengnan Shi and Pradip K. Srimani. An efficient distributed protocol for online gossiping problem. In *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 599–602, Washington, DC, USA, 2005. IEEE Computer Society.
- [63] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers*, 29(12):1104–1113, 1980.
- [64] Anthony Stentz. The focussed d* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.
- [65] Katia Sycara. Multiagent systems. *AI Magazine*, 10(2):79–93, 1998.
- [66] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt. A specification-based intrusion detection system for aodv. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 125–134, New York, NY, USA, 2003. ACM Press.
- [67] Richard Tynan, David Marsh, Donal O’Kane, and G. M. P. O’Hare. Intelligent agents for wireless sensor networks. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1179–1180, New York, NY, USA, 2005. ACM Press.
- [68] Jorge M. S. Valente and Rui A. F. S. Alves. Beam search algorithms for the early/tardy scheduling problem with release dates. FEP Working Papers 143, Universidade do Porto, Faculdade de Economia do Porto, April 2004. available at <http://ideas.repec.org/p/por/fepwps/143.html>.
- [69] Wooi-Ghee Wang, Takahiro Hara, Masahiko Tsukamoto, and Shojiro Nishio. Aodv compatible routing with extensive use of cache information in ad-hoc networks. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 852–859, New York, NY, USA, 2002. ACM Press.

- [70] Richard A. Wasniowski. Multi-sensor agent-based intrusion detection system. In *InfoSecCD '05: Proceedings of the 2nd annual conference on Information security curriculum development*, pages 100–103, New York, NY, USA, 2005. ACM Press.
- [71] Patrick Henry Winston. *Perspective*. MIT Press, Cambridge, MA, 1984.
- [72] Patrick Henry Winston. *Artificial intelligence (3rd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.