UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS - ICEX
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

EXTRAÇÃO DE DADOS EM REDES DE NÓS SENSORES SEM FIO UTILIZANDO ROBÔS MÓVEIS

MARCELO BORGHETTI SOARES

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

Belo Horizonte

Dezembro, 2004

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Extração de Dados em Redes de Nós Sensores sem Fio Utilizando Robôs Móveis

MARCELO BORGHETTI SOARES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Dr. Mário Fernando Montenegro Campos - Orientador
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Dr. Guilherme Augusto Silva Pereira - Co-orientador
Departamento de Engenharia Elétrica – PPGEE – UFMG

Prof. Dr. Geraldo Robson Mateus
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Dr. Diógenes Cecilio da Silva Jr.
Departamento de Engenharia Elétrica – PPGEE – UFMG

Belo Horizonte, 6 de Dezembro de 2004.

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS - ICEX
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DATA EXTRACTION IN WIRELESS SENSOR NETWORKS USING MOBILE ROBOTS

MARCELO BORGHETTI SOARES

Dissertation presented to the Graduate Program in Computer Science at Federal University of Minas Gerais, as partial requirement to obtain the degree of Master in Computer Science

Belo Horizonte
December, 2004

Resumo

O objetivo deste trabalho é propor métodos para realizar a *extração de dados em redes de nós sensores sem fio* por meio da utilização de robôs móveis. A abordagem a ser apresentada representa uma alternativa aos algoritmos de roteamento em redes de sensores que tentam minimizar o custo em energia associado à transmissão de dados. Este trabalho é motivado por aplicações como, por exemplo, monitoramento do ambiente e exploração de ambientes inóspitos. Além da tarefa de extração de dados, o robô pode ser usado para deposição e recalibração de nós sensores, fusão de dados e manutenção da cobertura e conectividade da rede. Aspectos referentes à navegação do robô e exploração do ambiente também são endereçados. Foram implementados dois métodos de extração de dados: i) Método Reativo, no qual a quantidade de dados coletados pelos sensores é representada por meio de *funções de potencial* cujo gradiente atrai o robô e ii) Método Híbrido, que inclui em sua metodologia heurísticas para resolução do *Problema do Roteamento de Veículos (VRP)*, onde o objetivo é distribuir um conjunto de *nós clientes* entre um conjunto de *veículos* e gerar rotas otimizadas em cada agrupamento formado. Esses dois métodos tentam minimizar variáveis como trajetória percorrida, tempo de realização da tarefa, quantidade de dados perdidos, etc. Ambos foram avaliados em um ambiente simulado com 60 nós sensores e com até 10 robôs. Ao final deste texto, resultados relativos à trajetória do robô, eficiência, perdas de dados na rede e quantidade de dados coletados pelo robô, são comparados e discutidos.

Abstract

The goal of this work is to propose methods for data extraction in wireless sensor networks using mobile robots. This approach represents an alternative to the routing algorithms used in sensor networks that try to minimize energy consumption associated with data transmission. The methods developed here are motivated by applications such as environment monitoring and inhospitable environment exploration. In addition to the data extraction task, the robots can be used for node deployment and re-calibration, data fusion and coverage and connectivity repair. Aspects related to robot navigation and environment exploration are addressed as well. Two methods for data extraction were implemented: i) Reactive Method in which the data collected in the sensor node is modelled as a potential function whose gradient attracts the robot and ii) Hybrid Method, that includes heuristics for the resolution of *Vehicle Routing Problem (VRP)*, where the objective is to distribute a *set of clients* among a *set of vehicles* and create optimized routes in every group. These methods try to minimize variables as path trajectory, total task time, data loss, robot's energy, etc. Both were evaluated in a simulated environment with 60 sensor nodes and up to 10 robots. Finally, numerical results related to path trajectory, efficiency, data loss and data extracted by the robot are compared and discussed.

Agradecimentos

Agora que mais uma etapa de minha vida se conclui, é hora de olhar para trás e relembrar os momentos de aprendizado e amadurecimento conquistados nesse período de dois anos que estive na UFMG. Impossível, portanto, não pensar nas pessoas especiais que sempre me acompanharam e nas muitas pessoas que daqui em diante farão parte da minha história.

Agradeço à minha família, meus pais e irmãos que nos momentos felizes e tristes moldaram a pessoa que sou hoje. Em especial a minha mãe, por sempre ter ensinado, mesmo indiretamente, que a cultura vale mais do que qualquer coisa. Obrigado mãe!

Ao meu orientador Mário que confiou em mim durante todo mestrado. Sempre saí de sua sala motivado, mesmo em ocasiões em que eu merecia "levar aquele puxão de orelhas". Hoje, admiro ainda mais a sua sabedoria.

Ao meu co-orientador Guilherme que acompanhou de perto as minhas dificuldades. Graças a sua qualidade técnica e duras críticas aprendi muito mais do que imaginava.

Aos meus colegas do VeRLab, pela momentos de convivência que tivemos ao longo desse ano. Troquei idéias com alguns e ri com outros. Posso dizer que, junto com eles, visto com orgulho a camisa do VeRLab.

À secretaria do DCC pela eficiência, em especial à Cida e Sônia, que prepararam a minha viagem para o Japão. Agradeço também aos coordenadores do *Projeto Sensornet* que viabilizaram financeiramente a ida à terra do sol nascente. Nossa, foi tudo tão corrido! Mas como valeu a pena!

A todos os meus amigos que tive a felicidade de conhecer em Belo Horizonte. Meu amadurecimento pessoal se deve em grande parte a essas pessoas, que me mostraram como a vida pode ser realmente boa de ser vivida. Seja no apoio dado, nas inumeráveis sessões no Cine Belas Artes ou nas conversas sérias (ou nada sérias) pelas ruas de BH.

A todas essas pessoas, e já emocionado, eu agradeço de coração!

Sumário

Lista de Figuras

Capítulo 1

Introdução

Este trabalho tem por objetivo propor soluções para o problema da *extração de dados em redes de nós sensores sem fio* por meio da utilização de robôs móveis. Os dados coletados nessas redes podem conter informações do ambiente como temperatura, umidade, pressão, luminosidade, etc. e são obtidos por meio de pequenos dispositivos, chamados de *nós sensores*. Por serem dispositivos com capacidades de armazenamento e processamento limitadas, a informação coletada não pode permanecer indefinidamente nos mesmos, de forma que é necessário transmití-la para uma *unidade central* (*sink*) que fará o processamento e a análise dos dados.

A transmissão dos dados nessas redes é uma operação que consome energia e, visto que o tempo de vida dos nós sensores está relacionado ao uso de baterias, isso torna-se um fator crítico. Nesse contexto, um agente móvel (por exemplo, robô) poderia monitorar regiões da rede e planejar a melhor maneira de coletar os dados adquiridos pelos nós, evitando, assim, a comunicação freqüente entre os sensores. Complementarmente, a utilização de robôs móveis irá atacar o problema do consumo de energia. Porém, para esse texto é importante se ter em mente que o objetivo é *apresentar métodos adequados para realizar a extração dos dados*, utilizando-se robôs móveis, partindo-se do pressuposto de que tal estratégia representa uma alternativa *viável* à diminuição da comunicação freqüente entre os nós sensores e, consequentemente, ao problema do consumo de energia. Como esse problema é crítico no projeto e utilização de redes de nós sensores, muito estudo (por exemplo, roteamento, mapas de energia) é realizado para tratá-lo. Assim, não serão apresentados resultados comparativos entre soluções tipicamente utilizadas e os métodos apresentados neste texto. Os resultados desse trabalho serão obtidos a partir da comparação entre os métodos de extração de dados aqui mostrados.

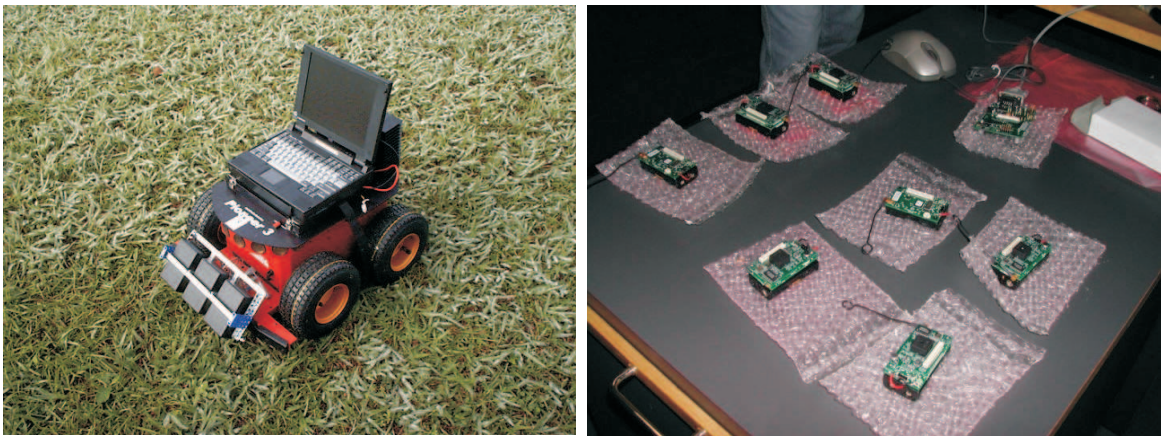

Figura 1.1: Robô *Pioneer* e nós sensores do *Projeto Sensornet* [Sensornet, 2004]. O robô, na imagem à esquerda, está preparado para realizar uma tarefa de deposição de nós sensores (caixas pretas) em posições específicas. Na imagem à direita pode-se ver 8 nós sensores utilizados no projeto para coletar dados de temperatura do ambiente.

1.1 Motivação

A tarefa específica de extração de dados em redes de nós sensores sem fio por meio de robôs móveis endereça diversos aspectos como, por exemplo, navegação, exploração e planejamento de trajetória. Em muitas aplicações, a maneira com a qual o robô lida com esses aspectos é dependente dos dados coletados pela rede, seja em termos quantitativos (quantidade de dados) ou qualitativos (informação contida nos dados). Assim, podem-se citar os seguintes exemplos, que servem de motivação para muitos trabalhos encontrados na literatura:

Monitoramento do ambiente. Uma das aplicações largamente citadas que fazem uso de redes de nós sensores é o monitoramento do ambiente. A Figura 1.1 mostra um robô *Pioneer* [ActivMedia Robotics, 2004] e um grupo de sensores usados no *Projeto Sensornet* [Sensornet, 2004]. Os nós sensores podem ser depositados, por exemplo, em uma floresta ou próximo a um vulcão e coletar informações como temperatura, umidade, qualidade do ar, etc. Tais medições podem ser úteis na previsão do tempo, detecção de incêndios, tremores sísmicos, erupções vulcânicas, etc. Os métodos propostos neste texto apresentam alternativas interessantes para realizar a extração de dados. Entre essas alternativas pode-se citar uma metodologia diferenciada para geração das rotas a serem seguidas pelos robôs. Os métodos podem ainda ser empregados, com as devidas modificações, em tarefas de deposição, recalibração ou recarregamento (baterias) dos nós sensores.

Cabe ainda ressaltar que a utilização de robôs e nós sensores é mais interessante do que o emprego de robôs móveis apenas. Os nós sensores podem coletar dados do ambiente constantemente, ao passo que um robô não pode estar em todas as posições de interesse ao mesmo tempo. Além disso, já que a comunicação entre os nós diminuiria consideravelmente, os sensores poderiam ser depositados em posições afastadas umas das outras, de forma a abranger uma região maior.

Exploração. A exploração de ambientes remotos realizada pela NASA desde 1997, [NASA, 2004], pode criar uma grande gama de aplicações que se utilizam da junção das duas áreas subjacentes. Em ambientes totalmente desconhecidos e inóspitos, como *Marte*, o emprego de nós sensores espalhados pelo ambiente pode ser valioso pois, a partir das informações coletadas por esses nós, um robô móvel poderia navegar de forma mais eficiente pela região. Os nós sensores poderiam informar ao robô sobre a presença de obstáculos ou a ocorrência de eventos de interesse na região. Da mesma forma, a deposição dos nós poderia ser feita pelos próprios robôs, levando-se em consideração, por exemplo, regiões do ambiente ainda não visitadas.

Resgate de vítimas em acidentes. O resgate de vítimas em acidentes, tais como incêndios, desabamentos, etc., serve de motivação para alguns trabalhos, [Peterson and Rus, 2004], [Kantor et al., 2003]. A idéia aqui é usar o robô para guiar as vítimas para locais seguros. Como os nós sensores encontram-se espalhados por todo o ambiente, em uma situação de incêndio, por exemplo, locais com temperatura muito alta poderiam ser evitados pelos robôs à medida que estes navegassem em busca de uma região segura para as vítimas.

1.2 Definição do Problema

O problema tratado neste texto pode ser apresentado da seguinte maneira:

Definição 1 *Dado um conjunto* $\mathcal{S} = \{s_1, s_2, ... \ s_n\}$ *de* n *nós sensores, espalhados em um ambiente* $\mathcal{W} \in \mathbb{R}^2$*, com quantidade de dados* $d_i(t)$ *em um instante* t *para* $1 \leq i \leq n$*, extrair os dados coletados pela rede usando um conjunto* $\mathbb{R} = \{R_1, R_2, ... \ R_m\}$ *de* m *robôs com capacidade de armazenamento de dados* Q_i *para* $1 \leq i \leq m$*, de forma que o custo associado à realização dessa tarefa, para os elementos de* $\mathcal{S}$ *e* $\mathcal{R}$*, seja minimizado.*

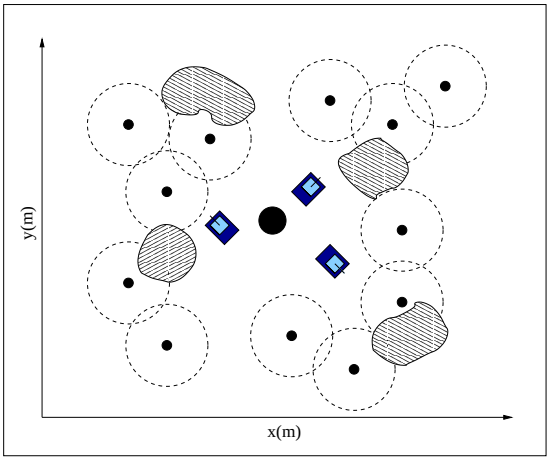

Figura 1.2: Ambiente de exemplo. Três robôs (retângulos) estão posicionados próximos à unidade central (círculo preto) de onde irão partir para coletar as informações dos nós sensores (pontos pretos), desviando-se de obstáculos (figuras hachuradas) presentes no ambiente.

O problema proposto acima deve levar em conta diversas considerações, especialmente sobre a rede de nós sensores. Conforme pode ser visto na Figura 1.2, existe apenas uma unidade central por meio da qual os robôs são comandados. Para este trabalho escolheu-se uma representação mais simples e situações nas quais tenha-se múltiplas unidades centrais serão abordadas em trabalho futuros. O robô é representado pela configuração $q_R(t) = (x_R, y_R)$ no seu espaço de configurações $\mathcal{C}$. O ambiente representado por $\mathcal{W}$ pode incluir ainda uma região de obstáculos $\mathcal{O} \subseteq \mathcal{W}$ [Latombe, 1991], [LaValle, 2004]. O problema tratado neste trabalho foi resolvido em ambientes com a presença de obstáculos, porém os resultados numéricos a serem apresentados no Capítulo 5 foram obtidos a partir de ambientes sem obstáculos.

O *custo* mencionado na Definição 1 pode levar em conta diversas variáveis: trajetória percorrida, tempo de realização das tarefas (eficiência), energia dos robôs, perda de dados na rede, etc. Dessa forma, a proposta deste trabalho é implementar métodos por meio dos quais seja possível realizar a extração de dados em redes de nós sensores de forma a minimizar as variáveis citadas anteriormente. Além disso, é interessante estimar a quantidade de dados coletados pelo robô, já que essa é a razão pela qual os métodos foram designados.

Para este trabalho foram propostos dois métodos. O primeiro deles (*Método Reativo*), proposto no Capítulo 3, tenta minimizar a quantidade de dados perdidos na rede (pode ser impossível garantir que não ocorram perdas de dados, mas, dependendo

da aplicação nas quais as redes de sensores são empregadas, essas perdas podem ser toleradas). Nesse esquema, a 'urgência' com a qual um determinado nó sensor deve ser atendido é modelada por meio de *funções de potencial* [Latombe, 1991] diretamente relacionadas a quantidade de dados $d_i(t)$. O segundo método (*Método Híbrido*) tenta minimizar diretamente a trajetória percorrida pelo robô e, indiretamente, pode-se esperar, embora não se possa garantir, que o tempo de execução, a quantidade de dados na rede e a energia do robô, sejam minimizadas também por meio desse método. Nesse método foram empregadas soluções para o *Problema de Roteamento de Veículos* (VRP), [Dantzig and Ramser, 1959], [Bodin, 1983], [Hjorring, 1995], [Aronson, 1996], [Laporte and Semet, 1999], [Díaz, 2003], oriundo das áreas de *Otimização Combinatória* e *Pesquisa Operacional*. Esse problema combina outros dois problemas, o *Problema do Caixeiro Viajante* (*Travelling Salesman Problem* - TSP) e o *Bin Packing*, [Cormem, 2001]. A quantidade de dados $d_i(t)$ (demanda no instante t) dos nós sensores e a capacidade de armazenamento de dados Q_i dos robôs são variáveis importantes no Método Híbrido, uma vez que a distribuição da rede de sensores entre os robôs será feita com base no valor dessas variáveis. Entretanto, nesse trabalho, para fins de simplificação do problema, a capacidade máxima de cada nó sensor foi utilizada como sua respectiva demanda.

Ambos os métodos podem ser vistos como heurísticas, já que em nenhum dos casos é possível garantir que se possa alcançar a solução ótima.

1.3 Contribuições

As contribuições do presente trabalho são mostradas a seguir:

1. Uma proposta alternativa para diminuição do consumo de energia em redes de nós sensores sem fio. Nessa abordagem os dados coletados pela rede são extraídos por meio de robôs móveis. Dessa forma, a comunicação necessária para realizar o roteamento dos dados é eliminada ou pelo menos minimizada, ecomizando energia. Dois métodos foram propostos para a extração de dados da rede:

   (a) Método Reativo;

   (b) Método Híbrido;

2. Revisão bibliográfica aprofundada sobre trabalhos que empregam robôs móveis em redes de nós sensores sem fio;

3. Revisão bibliográfica aprofundada sobre trabalhos relacionados ao Problema do Roteamento de Veículos.

1.4 Organização da Dissertação

Este texto está organizado da seguinte maneira: o Capítulo 2 apresenta os principais trabalhos relacionados à área de robótica e redes de sensores sem fio. Embora o objetivo desse trabalho seja apresentar alternativas para um problema específico (extração de dados em redes de nós sensores), foi possível relacionar alguns pontos comuns com os trabalhos do Capítulo 2. Além disso, também são mostrados trabalhos referentes a soluções heurísticas para o Problema do Roteamento de veículos (VRP), superficialmente introduzido neste capítulo. O Capítulo 3 apresenta o primeiro método empregado (Reativo), no qual o principal objetivo é extrair os dados da rede minimizando a perda de dados. Para tanto, a quantidade de pacotes presentes nos nós foi modelada por meio de campos de potencial. O Capítulo 4 apresenta o segundo método implementado (Híbrido), que utiliza soluções para o VRP. As duas metodologias são comparadas no Capítulo 5 e são apresentados resultados obtidos por meio de simulações. Finalizando, no Capítulo 6 tem-se a conclusão e as direções futuras de pesquisa.

Capítulo 2

Trabalhos Relacionados

A extração de dados em redes de nós sensores, tipicamente, é realizada por meio de roteamento. Cada nó sensor da rede transmite a informação coletada para nós vizinhos, e esse processo é repetido até que o dado chegue à unidade central. Esse esquema de transmissão de dados é conhecido como *multi-hop* e é uma das principais formas de economia de energia. Como a energia requerida na comunicação entre dois nós sensores é dependente da distância entre eles, a transmissão de dados de um nó sensor diretamente para a unidade central (esquema conhecido como *single hop*) pode ser muito mais custosa do que a transmissão *multi-hop*. Os algoritmos de roteamento trabalham de várias formas, por exemplo, criando fluxos de dados em direção à unidade central [Intanagonwiwat et al., 2000], ou elegendo líderes responsáveis pelas áreas de cobertura previamente determinadas e pela comunicação direta com a unidade central [Heinzelman et al., 2000]. Conforme foi dito anteriormente, extração de dados por meio de roteamento está fora do escopo deste texto e, portanto, nenhum tabalho nessa área foi investigado.

A proposta desse trabalho é utilizar robôs móveis para realizar a extração de dados. Porém, muitas aplicações empregam robôs móveis com diferentes propósitos. Por exemplo, problemas que surgem em função do funcionamento da rede a longo prazo (nós sem energia, descalibrados, congestionamento da rede, etc.), manutenção da *cobertura* e *conectividade* da rede, *navegação* de um agente móvel e *exploração* do ambiente, encontram soluções interessantes a partir do emprego de robôs em redes de nós sensores. Em [LaMarca et al., 2002] os robôs móveis são utilizados para recalibrar nós sensores. Essa tarefa é apresentada no âmbito do sistema *PlantCare*, no qual um grupo de robôs autônomos deve monitorar plantas de residências. Além dessa tarefa, são levantadas outras situações nas quais as duas áreas podem ser utilizadas conjuntamente. Essas

situações partem do princípio de que a rede de sensores é um repositório de informação sem nenhuma capacidade de atuação no ambiente, ao passo que robôs são agentes com capacidade sensorial limitada. A Seção 2.1 apresenta os principais trabalhos nessa área que de alguma forma relacionam-se com o problema aqui tratado. Nesse texto, a extração de dados relaciona-se também com a navegação do robô pela rede de nós sensores, de forma que os trabalhos apresentados na próxima seção referem-se a planejamento de rotas e coleta de informações. Nestes trabalhos, basicamente, as principais tarefas dos robôs móveis podem ser classificadas da seguinte forma:

1. Deposição de nós sensores por robôs móveis: endereça alguns aspectos como conectividade e cobertura em redes de nós sensores.

2. Navegação e exploração por robôs móveis: o robô móvel pode utilizar a informação local aos nós sensores para navegação e exploração do ambiente, resgate de vítimas em desastres, rastreamento de alvos móveis. Complementarmente, tarefas como a extração e fusão de dados podem ser realizadas à medida que o robô navega pela rede.

Como se pode supor, os assuntos não são excludentes. Embora os trabalhos foquem, basicamente, na solução de um dos problemas, existem vários aspectos que não podem ser tratados de forma isolada. Por exemplo, os robôs móveis durante a navegação podem ser auxiliados pelos nós sensores, de forma que a deposição desses deva ser cuidadosamente planejada.

O problema aqui tratado foi definido na seção 1.2 e, conforme mencionado, um dos métodos empregados em sua resolução pode ser facilmente relacionado a um problema de otimização combinatória e pesquisa operacional (VRP). Assim sendo, algumas soluções heurísticas para esse problema foram investigadas na seção 2.2 e servirão de base para a construção do método apresentado no Capítulo 4. O estudo destas soluções heurísticas tem por objetivo encontrar caminhos para o robô que se aproximem da solução ótima, minimizando custos associados a trajetória, tempo, energia, etc.

2.1 Robótica e redes de sensores sem fio

[Li et al., 2003] apresentam um algoritmo distribuído para guiar agentes autônomos (que podem ser pessoas) em uma rede de sensores sem fio, por regiões consideradas

seguras. Por exemplo, os nós podem estar espalhados em uma área florestal para monitoramento de incêndios, e as regiões que detectassem eventos do tipo *temperatura alta* seriam consideradas inseguras e estariam excluídas do caminho gerado pelo algoritmo. A técnica utilizada consiste em modelar as regiões inseguras por meio de *campos de potencial* repulsivos. Dessa forma, o algoritmo precisa planejar rotas na rede de nós sensores que levem o agente da posição atual até uma posição alvo, desviando de regiões críticas. É particularmente interessante para redes de nós sensores voltadas a *eventos*, onde espera-se que a solicitação de determinados nós aconteçam. Entretanto, requer constante comunicação entre o agente móvel e a rede de nós sensores, o que pode trazer uma série de problemas, tais como perda de dados, congestionamento da rede (já que *broadcast* é constante), conexão assimétrica entre os nós. O algoritmo empregado é *guloso* e consegue obter a rota ótima entre dois pontos, visto que os sensores da rede estão uniformemente dispostos no ambiente. Porém, a solução ótima entre entre dois pontos pode estar muito aquém da solução ótima que considere um conjunto de pontos pré-estabelecidos. Neste trabalho, o Método Híbrido computa a rota levando em conta n sensores (n alvos). Esse método utiliza heurísticas, visando obter caminhos otimizados. Além disso, admite-se a distribuição aleatória dos sensores pelo ambiente.

[Corke et al., 2003] estendem o trabalho de [Li et al., 2003] para o caso de um robô autônomo aéreo (helicóptero). O algoritmo tem as seguintes características: i) cada nó da rede precisa saber sua localização (não necessariamente exata) para prosseguir com a geração do caminho que o robô móvel seguirá; ii) cada nó da rede tem que ter a capacidade de computar, alterar e armazenar o caminho que o robô móvel seguirá (quando o nó fizer parte do caminho); iii) o robô móvel tem que ter a capacidade de interagir com os nós da rede de forma a tomar conhecimento do caminho gerado e de eventuais alterações. Como esse trabalho é apenas uma extensão de [Li et al., 2003], os mesmos problemas podem ser aqui encontrados.

[Corke et al., 2004] estendem seu próprio trabalho em [Corke et al., 2003] para a deposição de nós sensores e a manutenção da conectividade em uma rede de nós sensores sem fio usando um robô autônomo aéreo. A etapa de deposição é feita da seguinte maneira: dada uma *topologia* específica de rede e uma *escala de deposição*, constrói-se um plano que os robôs devem seguir para depositar os nós. Cada nó sensor torna-se um vértice de um grafo na topologia pré-determinada: *estrela*, *grade* ou *aleatória*. A escala na qual os nós são depositados depende do alcance dos mesmos. A segunda fase consiste em testar a conectividade da rede. Um nó especial é configurado com a função de enviar pacotes a todos os nós. Esse nó aguarda a resposta de todos os

outros, de forma a certificar-se de que a rede esteja totalmente conectada. Se for detectado alguma ruptura na rede, alguns pontos de deposição são gerados de forma a restaurar a conectividade. A maneira como o robô irá alcançar as *posições-alvo* para depositar os nós é feita da seguinte forma: i) para cada ponto (x_i, y_i) dos nós sensores da rede, primeiro ordenam-se os pontos pela coordenada x e depois pela coordenada y; ii) depositam-se os nós na ordem estabelecida. Com esse método, a rota gerada pode estar muito aquém da rota ótima que contivesse todos os pontos, acarretando um custo maior (energia, combustível, etc.) ao robô aéreo que realizasse a deposição. Diferentemente, no Método Híbrido proposto neste trabalho, o robô móvel consegue chegar aos n sensores utilizando rotas que minimizam os custos associados às variáveis anteriormente citadas.

[Peterson and Rus, 2004] apresentam um dispositivo de *hardware* chamado de *Flashligth* que interage com a rede de nós sensores exatamente como foi descrito em [Li et al., 2003]. As funções realizadas pelo dispositivo são: i) ativar ou desativar uma área específica da rede (pois pode ser interessante manter certos nós *adormecidos* para diminuir o gasto em energia); ii) detectar a ocorrência de eventos na rede, tais como aumento de temperatura; iii) guiar o agente móvel ou robô para locais seguros em atividades de resgate de vítimas de incêndios florestais, terremotos, etc. Ainda utilizando a mesma técnica de campos de potencial para geração de caminhos, [Kantor et al., 2003] usam *Flashligth* para gerar rotas através da rede de nós sensores em um cenário de procura e resgate de vítimas de incêndios. Nesse trabalho e nos trabalhos citados anteriormente a rede de sensores pode ser vista como um repositório distribuído de informação. Cada sensor consegue estimar sua distância até o alvo e áreas de risco, após receber estimativas dos nós sensores vizinhos. Dessa forma, a comunicação entre os sensores é algo indispensável. No trabalho aqui proposto, ao contrário, o robô consegue estimar a quantidade de dados coletados pelos nós sensores usando uma equação de predição: a quantidade de dados presentes nos nós sensores irá aumentar determinísticamente de acordo com essa equação. Assim, a transmissão de dados entre os sensores da rede não é necessária.

[Tang et al., 2004] descrevem um algoritmo para deposição de nós sensores em ambientes fechados. A metodologia empregada nesse trabalho, utiliza um planejador centralizado para geração das posições dos nós que irão compor a rede. Primeiramente é gerado um mapa de ocupação [Elfes, 1987], [Moravec, 1988] em que estará representado o ambiente no qual o robô irá atuar, incluindo corredores, portas e obstáculos. Em um segundo momento, são geradas as posições nas quais os nós serão depositados,

que levam em conta vários aspectos como: manter a comunicação entre os nós, evitar que as posições geradas sejam próximas demais a ponto de dificultar a deposição, evitar posições nas quais existam portas ou obstáculos, etc. Essas posições são geradas em grupos de n sensores que, em um último momento, são atribuídos a robôs líderes responsáveis por guiar os nós (já que se tratam de nós sensores móveis) até a suas posições de destino. Esse método, implementado em um grupo de 70 nós sensores móveis, mostra-se interessante para ambientes fechados, nos quais as restrições devido à portas e obstáculos produz uma cobertura uniforme sobre todo o ambiente. No Método Híbrido aqui proposto, a rota seguida pelos robôs também é gerada por meio de um planejador centralizado. Porém, nesse trabalho, a extração de dados da rede de nós sensores é realizada em ambientes abertos e não há conhecimento das posições de obstáculos a priori. A rota gerada pelo planejador pode, então, ser utilizada para outros propósitos tais como recalibração ou deposição de nós sensores.

[Dhariwal et al., 2004] propõem um método para detecção e rastreamento de eventos em redes de nós sensores sem fio, baseado na quimiotaxia (atração ou repulsão entre células vivas e alguma substância química) de bactérias. O sistema encontra aplicação em tarefas de monitoramento de eventos por robôs móveis. Os eventos são representados por meio de campos de potencial e o controle do robô é feito utilizando *passeio aleatório induzido* (*biased random-walk*). Dessa forma, o robô mantém-se em uma determinada direção ao perceber acréscimo no gradiente, ao passo que com *passeio aleatório simples*, o robô estaria constantemente mudando sua orientação. O algoritmo foi testado em simulação e em robôs reais [Sibley et al., 2002] no Laboratório de Robótica da Universidade da Califórnia [USC Robotics Research Lab, 2004]. Com múltiplos robôs, o método mostrou-se bastante eficaz para ambientes nos quais exista a geração concorrente de eventos, com os robôs alcançando todas as fontes. A eficiência (tempo para que o robô alcance a origem do evento) é inferior à obtida pelo método do *gradiente descendente simples*. No método do gradiente descendente simples, porém, os robôs sentiam-se atraídos para poucas fontes, sendo que algumas não eram rastreadas. No Método Reativo proposto neste trabalho, o robô segue o gradiente de funções de potencial e, portanto, a performance é comparável à obtida pelo gradiente descendente simples. Complementarmente, a distribuição dos nós sensores (fontes) é realizada por um planejador centralizado, de forma que cada robô sente-se atraído por apenas um subconjunto de nós sensores. Assim, não existe a possibilidade de dois robôs rastrearem a mesma fonte eventos e permanecerem juntos durante a execução de suas tarefas.

[Batalin and Sukhatme, 2004] endereçam os seguintes problemas i) deposição de

nós sensores; ii) exploração do ambiente por um robô móvel. O robô utiliza informação dos nós presentes na rede para realizar a exploração do ambiente e a deposição de outros nós sensores. Os nós recomendam direções (norte, sul, leste, oeste) para as quais o robô pode navegar com base em variáveis internas. Quando o robô entra no raio de alcance do nó sensor, esse indica, primeiramente, as direções não visitadas. Caso todas direções já tenham sido visitadas, então o nó irá indicar a direção com menos ocorrências de visitas. Assim, é possível, ao mesmo tempo, navegar por todo o ambiente e efetuar a deposição de nós em locais que o robô estiver explorando pela primeira vez. O ambiente em que o robô atua é modelado como um grafo no qual a transição para um nó é realizada através de uma função de probalidade $\mathcal{F}_P(s'|s,a)$ (probabilidade de chegar ao vértice s' estando no vértice s e executando a ação a). A escolha de uma abordagem probabilística deu-se devido ao desconhecimento do mapa do ambiente. Se o mapa fosse conhecido a priori, o problema da navegação entre dois pontos poderia ser resolvido de forma ótima realizando-se uma busca em profundidade no grafo (*Depth First Search*, [Cormem, 2001]). Ainda que não tenha sido proposto com a função de extração de dados na rede, o algoritmo se mostra interessante para realizar tal ação, em cenários nos quais nenhum mapa é fornecido e o robô não possua dispositivos de localização (GPS).

[Meguerdichian et al., 2001] tratam o *Problema da Cobertura* realizado por agentes móveis, no pior e no melhor caso, em redes de sensores. O problema da cobertura pode ser enunciado como: *dado um ambiente $\mathcal{W}$ e um conjunto $\mathcal{S}$ de nós sensores com posições conhecidas, ache o caminho P_b entre dois pontos i e f dados, de forma que, para cada ponto p presente em P_b, a distância de p ao nó sensor mais próximo é maximizada (pior caso) ou minimizada (melhor caso).* A solução para o pior caso foi obtida gerando-se um *Diagrama de Voronoi* [Aurenhammer, 1991] no qual cada linha do diagrama está eqüidistante de dois nós sensores. O Diagrama de Voronoi pode ser facilmente representado por um grafo G cujas arestas são as linhas que separam cada uma das células do diagrama e cujos vértices são as intersecções dessas linhas. Como cada linha está eqüidistante de dois nós vizinhos (maximização da distância entre os nós), então o caminho correspondente ao pior caso de cobertura será traçado sobre as linhas do diagrama. A solução para o melhor caso consiste em gerar uma *Triangulação de Delaunay* [Mulmuley, 1994] entre os nós sensores. Como esse método produz triangulações que têm arestas de tamanho mínimo, entre todas as possíveis triangulações, então o caminho correspondente ao melhor caso (minimização da distância entre os nós) será traçado sobre o Diagrama de Delaunay. Identificar áreas de menor cobertura

pode ser útil na tarefa de sugerir novos pontos de deposição de nós sensores, ao passo que identificar áreas de maior cobertura pode facilitar tarefas de monitoramento do ambiente, tais como extração de dados. Entretanto, para esse último caso, a solução encontrada não minimiza o caminho entre dois pontos específicos. No Método Híbrido aqui proposto, não existe a preocupação de navegar por áreas com melhor cobertura, pois o objetivo é coletar dados de todos os nós sensores do ambiente.

[Veltri et al., 2003] também estudam o problema de cobertura máxima em redes de nós sensores reduzindo-o ao *Problema do Caminho Mais Longo* (*Longest Path Problem*), que é sabidamente um problema $\mathcal{NP}$*-completo*. Cada aresta presente no grafo G é rotulada com o valor da cobertura total recebida ao longo de todos os seus pontos. Se, segundo os autores, as arestas forem rotuladas com o comprimento do caminho entre dois vértices, o problema pode ser resolvido por meio de heurísticas utilizadas no Problema do Caminho Mais Longo e sugerem várias delas, além de elaborarem uma heurística própria. De maneira similar ao trabalho anterior, a solução encontrada não minimiza o caminho entre dois pontos específicos.

[Mehta et al., 2003] estudam também o pior e o melhor caso de cobertura em uma rede de nós sensores sem fio. Para solução do problema da cobertura é criada uma *Árvore Geradora Máxima* na qual os nós presentes representam os vértices do Diagrama de Voronoi e as arestas representam as linhas que conectam os vértices do Diagrama de Voronoi. Os autores provam que, para um par de vértices i e f no Diagrama de Voronoi, o único caminho existente entre os dois pontos na Árvore Geradora Máxima representa o pior caso de cobertura (ou similarmente, o único caminho existente entre os dois pontos em uma árvore geradora mínima representa o melhor caso de cobertura). Os autores estendem o trabalho de [Meguerdichian et al., 2001] para achar os caminhos que representam os melhores e piores casos de cobertura que sejam ao mesmo tempo os caminhos mais curtos. Para tanto, considerando-se o pior caso de cobertura, o algoritmo propõe-se a achar o melhor caminho para uma dada distância mínima fornecida entre os nós. Entretanto, a prova para esse caso é omitida. No Método Híbrido, uma das heurísticas para geração de rotas baseia-se na construção de uma Árvore Geradora Mínima a partir de um grafo G. Porém, diferentemente do trabalho descrito em [Mehta et al., 2003], a rota gerada inclui todos os nós das rede.

[Wu et al., 2004] introduzem o termo *redes distribuídas de nós sensores baseadas em agentes móveis* (*mobile agent-based distributed sensor networks*) em uma tarefa de detecção e rastreamento de alvos móveis. O problema é traçar a rota de um robô móvel na rede de nós sensores para realizar a fusão das informações coletadas pelos nós, à

medida que esses detectam um alvo de interesse dentro do seu alcance. Para tanto, é definida uma função diretamente proporcional à energia do sinal (*signal strength*) recebida pelos nós quando o alvo está sendo detectado e inversamente proporcional as perdas associadas ao sinal transmitido e recebido pelos nós ao longo do caminho (*path loss*) e também ao consumo de energia pela rede (*energy consumption*). O trabalho prova que se trata de um problema $\mathcal{N}P$*-completo* e cuja solução é obtida por meio de heurísticas baseadas em *Algoritmos Genéticos* [Holland, 1975]. Embora nenhuma função de custos tenha sido desenvolvida para o trabalho aqui proposto, o desenvolvimento dos métodos de extração de dados visa também minimizar variáveis como trajetória, tempo e quantidade de dados perdidos na rede. Resultados obtidos por simulação foram analisados.

2.2 Problema do Roteamento de Veículos

Comumente, na resolução do VRP, são empregadas heurísticas que, embora não garantam a solução ótima, conseguem obter resultados satisfatórios. As heurísticas para o VRP, segundo [Laporte and Semet, 1999], podem ser classificadas em duas categorias: *heurísticas clássicas* e *meta-heurísticas*. As heurísticas clássicas realizam uma exploração limitada no espaço de busca, mas com resultados satisfatórios. Já as meta-heurísticas, que consistem basicamente em melhorias nas heurísticas clássicas, realizam uma exploração mais refinada no espaço de busca, utilizando memória, regras de busca entre os vizinhos e recombinação de soluções. Embora as soluções sejam superiores, o custo computacional é também mais elevado.

Heurísticas clássicas são tipicamente separadas em três categorias: i) *Heurísticas Construtivas*, que contêm em sua formulação critérios de minimização de custo; ii) *Heurísticas de Duas Fases*, que separam a solução do problema em duas partes, *agrupamento* e *criação de rotas* em cada agrupamento; iii) *Heurísticas de Refinamento* que consistem em melhorar soluções previamente construídas. Essa última categoria está além do escopo desse texto.

[Clarke and Wright, 1964] propuseram um algoritmo que introduz a noção de *economia*. *Economia* é um custo calculado para cada par de vértices (s_i, s_j) da seguinte forma: $e_{ij} = w_{i0} + w_{0j} - w_{ij}$ para $i \neq j$, onde w_{ij} representa o custo de ligação entre os nós s_i e s_j. A idéia é juntar duas rotas $(s_0 \prec s_i \prec ... \prec s_0)$ e $(s_0 \prec ... \prec s_j \prec s_0)$ que sejam fisicamente *concatenáveis*, onde $\prec$ é o símbolo de precedência e s_0 é o nó inicial . É um método que apresenta boas respostas e é utilizado quando o número de veículos

a serem empregados é desconhecido, já que nós que não são fisicamente concatenáveis pertencerão a rotas diferentes. Um problema que surge com esse método, de acordo com [Laporte and Semet, 1999], é que em alguns casos é possível que se formem rotas circulares. Para corrigir essa deficiência, Gaskell e Yellow, conforme é reportado em [Laporte and Semet, 1999], incluiram na equação da *economia* um parâmetro Λ: $e_{ij} = w_{i0} + w_{0j} - \Lambda w_{ij}$. Quanto maior Λ, mais importante se torna a distância entre dois pares de nós (s_i, s_j).

Outra heurística construtiva interessante baseia-se na idéia de *inserção seqüencial* e foi proposta por Mole e Jameson e pode ser vista em [Laporte and Semet, 1999]. Para cada vértice s_k que ainda não pertença a uma rota, calcula-se o custo mínimo $min\{\mu(a, k, b)\}$ de sua inserção para todos os vértices adjacentes (s_a, s_b) pertencentes a rota $(s_i \prec ... \prec s_j)$. O custo μ é baseado na idéia de *economia* proposto por Gaskell e Yellow, ou seja $\mu(s_i, s_k, s_j) = w_{ik} + w_{kj} - \Lambda w_{ij}$. As rotas construídas por essa heurística são otimizadas com métodos como *2-opt* e *3-opt* [Aarts and Lenstra, 1997].

Christofides, Mingozzi e Toth (ver [Laporte and Semet, 1999]) propuseram uma heurística que utiliza a mesma idéia de inserção seqüencial. Uma rota com valor $k = 1$ é inicializada com um vérice s_k qualquer. Após, acha-se o vértice s_i com o menor valor de custo dado por $\sigma_i = w_{0i} + \Lambda w_{ik}$ que seja fisicamente inserível na rota k. A rota é então otimizada utilizando o método *3-opt*, por exemplo. O processo é repetido até que não existam mais nós que sejam inseríveis naquela rota. Se ainda restarem vértices sem rotas, a variável k é incrementada e o algoritmo é executado até que não existam mais nós sem rotas. [Laporte and Semet, 1999] reportam alguns resultados de comparações feitas entre o algoritmo de Christofides et al. e o algoritmo de Mole e Jameson. A heurística de Christofides et al., apresentou melhores resultados em tempo computacional menor.

A primeira heurística de duas fases a ser abordada nesse texto é chamada de *Sweep Algorithm*, proposta em [Gillet and Miller, 1974]. Aplicada em instâncias planas do VRP, é uma heurística simples. Consiste em representar os vértices em coordenadas polares (θ_i, ρ_i) com base no nó central. O algoritmo atribui para cada veículo k, começando pelos menores ângulos e raios, a quantidade de nós suficentes para que não seja ultrapassada a capacidade Q_k do veículo. Os nós restantes são atribuídos para os próximos veículos, sucessivamente. Para cada rota gerada utiliza-se uma heurística para resolução do TSP.

[Fisher and Jaikumar, 1981] apresentam uma heurística na qual *nós-semente* s_k são atribuídos para cada *agrupamento*. Para cada nó é calculado um custo de atribuição

do nó s_i ao agrupamento k, dado por $h_{ik} = min\{w_{0i} + w_{ik} + w_{k0}, w_{0k} + w_{ki} + w_{i0}\} - (w_{0k} + w_{k0})$. O problema reduz-se a solucionar um *GAP* (*Generalized Assignment Problem*) com custos h_{ik}, demandas d_i para os nós e capacidade Q_k dos veículos. Um dos problemas mais importantes com essa heurística é a necessidade de especificar *a priori* o número de veículos (formação de *agrupamentos*).

A meta-heurística baseada em colônias de formigas (*Ant Colony Systems*) apresentada em [Bullnheimer et al., 1997] tem sua origem no trabalho seminal de [Dorigo et al., 1996]. Originalmente foi utilizada para resolver instâncias do Problema do Caixeiro Viajante (TSP). A idéia consiste em empregar um conjunto de 'formigas' na escolha das cidades que farão parte da rota do caixeiro viajante. A seleção das cidades pelas formigas é feita usando-se uma função de transição probabilística que engloba fatores como tamanho da aresta que liga cidades e quantidade de '*feromônio*' entre as cidades. A importância desses fatores na função probabilística é regulado pelos parâmetros α e β. O feromônio ϱ é depositado ao longo das iterações do algoritmo, sempre que uma formiga escolhe um determinado caminho. A princípio, essa escolha (também baseada na função probabilística) é aleatória já que qualquer caminho é igualmente desconhecido. Mas à medida que o tempo passa, caminhos mais curtos são mais visitados e o acúmulo de feromônio torna-se maior, o que implica em maior probabilidade de escolha. Para o VRP, cada formiga constrói um caminho até que este não ultrapasse a capacidade Q previamente definida para os veículos. No final das iterações os caminhos que foram mais percorridos conterão maior quantidade de feromônio e poderão ser atribuídos um veículo em particular. Esses nós serão marcados e não poderão ser visitados novamente em próximas iterações.

Recozimento Simulado (*Simulated Annealing*) [Aarts and Korst, 1989] é uma técnica de otimização combinatória com inspiração em física estatística, baseada no *Algoritmo de Metrópolis*. O algoritmo pode ser descrito da seguinte maneira: dado o estado i de um sólido com energia E_i, um estado j com energia E_j é gerado devido à ocorrência de uma perturbação qualquer no sistema. Se a diferença de energia $(E_j - E_i)$ for maior ou igual a 0, o estado j torna-se o estado corrente. Se a diferença de energia for menor do que 0, o estado j é aceito com uma probabilidade igual a $e^{(E_i - E_j)/T}$, onde T é um parâmetro chamado 'temperatura'. O Método do Recozimento Simulado, ao contrário de *algoritmos de busca local* consegue escapar de *mínimos locais* já que estados diferentes podem ser aceitos com certa probabilidade. Um comparativo entre essa técnica e outras meta-heurísticas pode ser encontrado em [Dorigo et al., 1996].

Busca Tabu (*Tabu Search*) é uma técnica que frequentemente apresenta melhores

resultados que as demais meta-heurísticas aqui apresentadas. Um exemplo é mostrado em [Gendreau, 2002]. O espaço de busca no problema do VRP pode ser constituído de todos os pontos que representam rotas válidas para o problema. A cada iteração do algoritmo *Busca Tabu*, soluções vizinhas $N(S)$ podem ser geradas a partir da solução atual S. No VRP, essas soluções vizinhas podem ser representadas, por exemplo, pela mudança de um nó i de uma rota para outra. A maneira através da qual o método realiza a busca no espaço de estados baseia-se em duas idéias principais: *Lista Tabu* e *Critérios de Aspiração*. Uma lista tabu é usada principalmente para evitar ciclos durante a busca, o que poderia indicar a ocorrência de mínimos locais. Por exemplo no VRP, se um nó i foi recém movido de um caminho P_1 para um caminho P_2, o movimento deste mesmo nó do caminho P_2 para o caminho P_1 pode ser considerado um *movimento tabu*. Assim, enquanto esse movimento estiver presente na lista tabu, o algoritmo não optará por ele. Os critérios de aspiração são usados durante o processo de decisão quando a busca no espaço de estados está sendo realizada. Muitas vezes, mesmo quando não há risco de ciclos, caminhos contidos na lista tabu são evitados, ainda que levem a diminuição do custo total. Um dos critérios mais utilizados é justamente permitir movimentos que estejam presentes na lista tabu, se o ganho com esse movimento for maior do que com a solução atual. Um dos problemas inerentes a esse método diz respeito ao tempo de computação exigido para investigar o espaço de busca e para verificar se a solução S' não faz parte da lista tabu. Uma das maneiras empregadas para contorná-lo é limitando o espaço de busca a ser investigado. A cada iteração apenas uma amostra é analisada. Infelizmente, ao optar-se por essa estratégia, boas soluções podem ser ignoradas.

Algoritmos Genéticos [Holland, 1975] são técnicas com inspiração biológica, utilizadas em muitos problemas de otimização combinatória. Em [Bräysy and Gendreau, 2001] são apresentadas as principais idéias concernentes a esse método e uma revisão dos artigos mais conhecidos sobre o assunto para a resolução do VRP. O espaço de busca é constituído de *indivíduos*, codificados em uma representação geralmente binária. Entretanto, de acordo com [Bräysy and Gendreau, 2001], em problemas como o VRP e TSP, é pouco freqüente a adoção dessa estratégia. Geralmente os indivíduos são codificados em uma representação inteira, visando facilitar a identificação das cidades por seus respectivos números. A cada iteração do algoritmo genético os indivíduos são avaliados através de uma *função de fitness* (função que mede a adequação do indivíduo ao ambiente), e operadores de *crossover* (cruzamento entre os indivíduos) e mutação (troca aleatória de uma parte da informação contida nos

indivíduos, visando a variabilidade genética) são aplicados. A cada geração (iteração) é selecionada uma parcela dos indíviduos que farão parte da próxima geração. Por exemplo, podem-se selecionar 50% dos melhores indivíduos para a próxima geração. Como exemplo, [Thangiah, 1995] apresenta o sistema GIDEON, para resolução do problema do roteamento de veículos com restrição de tempo. É um algoritmo de duas fases (*cluster-first, route-second*): primeiramente a separação em *agrupamentos* é realizada por um algoritmo genético que atribui nós a rotas de acordo com suas coordenadas polares (semelhantemente à Heurística de Varredura). Posteriormente, as rotas locais a cada setor são criadas com um método semelhante à *inserção seqüencial* (também visto alguns parágrafos acima). Finalizando o processo, é utilizado um método para a otimização local das rotas criadas. Em [Bräysy and Gendreau, 2001] é também apresentado um comparativo entre os diversos métodos baseados em algoritmos genéticos apresentados no artigo e outras heurísticas incluindo *Ant Colony System* e *Tabu Search*.

A utilização de qualquer uma das técnicas discutidas acima requer um estudo cuidadoso do algoritmo e da aplicação que fará uso deles. Como já foi dito, as meta-heurísticas podem ser vistas como melhoramentos das heurísticas clássicas, mas todas tem em comum o fato de requererem maior tempo de execução do que os métodos convencionais. O espectro de utilização dessas técnicas pode ficar restrito, portanto, a aplicações que não exijam respostas imediatas, em tempo real. Nesses casos, ainda que o custo geral das rotas criadas seja superior ao calculado pelas meta-heurísticas, a velocidade da execução dos algoritmos é um fator mais relevante.

Como o objetivo do presente trabalho é extrair os dados da rede de forma eficiente, as soluções baseadas nas heurísticas apresentadas anteriormente podem ter um impacto positivo nos resultados. Dessa forma, a navegação do robô pelo ambiente pode considerar todos os nós sensores como pontos contidos em sua trajetória. Já que a rota é otimizada, pode-se esperar que muitas variáveis de interesse para o sistema também sejam minimizadas. Porém, uma seleção rigorosa dos métodos está fora de escopo, visto que o trabalho não é direcionado às áreas de otimização combinatória e afins. O Método Híbrido, que inclui soluções para o VRP em sua metodologia, foi testado com duas heurísticas conhecidas e apresentadas anteriormente: Heurística de Varredura e Heurística de Clarke & Wright.

Capítulo 3

Método Reativo

Nesse método a quantidade de dados contido no sensor é representado por meio de uma *função de potencial*. Essas funções geram campos de potencial que guiam o robô em direção aos nós com maior 'urgência' de atendimento. Consequentemente, espera-se que as perdas de dados da rede sejam minimizadas. A próxima seção desse capítulo descreverá brevemente conceitos fundamentais necessários à compreensão do método de extração de dados apresentado.

3.1 Conceitos Fundamentais

Antes de apresentar a metodologia empregada, é necessário descrever as redes de nós sensores utilizadas na implementação desse método. O conceitos fundamentais associados a redes de nós sensores também serão utilizados para o Método Híbrido mostrado no próximo capítulo. A segunda parte dessa seção descreverá brevemente as arquiteturas reativas e noções básicas sobre campos de potencial.

3.1.1 Características da rede de nós sensores sem fio

Cada nó sensor s_i da rede é composto de alguns transdutores (temperatura, umidade, sensor de luz, etc.), um dispostivo de comunicação omnidirecional com alcance definido por um raio r_i, e um processador com capacidade limitada de armazenamento e processamento. Os principais componentes de um nó sensor podem ser vistos na Figura 3.1, extraída de [Vieira et al., 2003]. O nó sensor s_i é representado no espaço de configuração $\mathcal{C}$ por $q_i = (x_i, y_i)$. A quantidade de dados, d_i, contida em um nó sensor

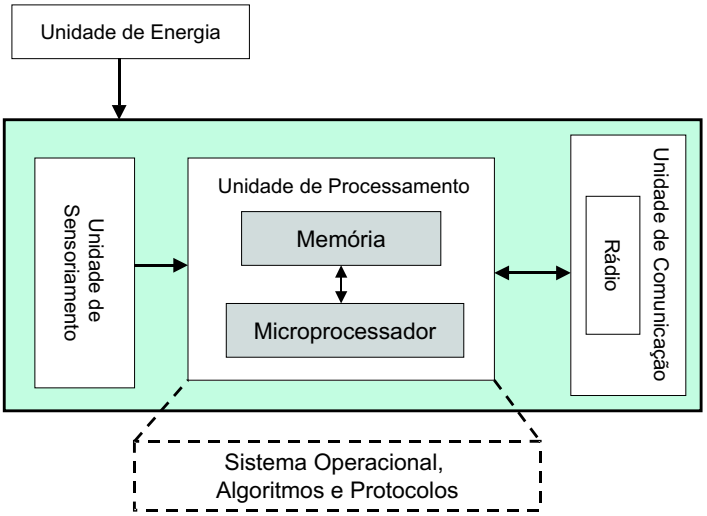

Figura 3.1: Componentes de um nó sensor. Unidade de energia, Unidade de Sensoriamento, Unidade de Processamento (memória, processador) e Unidade de Comunicação (rádio).

cresce de forma determinística segundo a equação:

$$\dot{d}_i = \begin{cases} (g_i - h_i) & \text{para } \|q_R - q_i\| \leq r_i \\ g_i & \text{de outra maneira} \end{cases} , \tag{3.1}$$

onde g_i é taxa de aquisição de dados do nó sensor s_i, h_i é a taxa de transferência de dados do nó sensor para o robô, q_R e q_i representam as posições do robô R e do nó sensor s_i, respectivamente, em um ambiente planar. Se $d_i \leq 0$ então $d_i = 0$, pois a quantidade de dados no nó sensor s_i necessariamente deve ser maior ou igual a zero.

Conforme pode ser visto na Figura 3.2, quando o robô móvel entra no raio alcance estabelecido pelo sistema de comunicação do nó sensor s_i, um *link* pode ser estabelecido entre ambos e, dessa forma, a informação armazenada pode ser transmitida para o robô. Nesse trabalho, não é feito qualquer tipo de roteamento de dados pela rede, logo a informação que está contida nos sensores só será transmitida para o robô . Nesse ínterim, a capacidade restante de armazenamento, c_i, em um nó sensor s_i é dada pela equação

$$c_i(t) = C_i - d_i \, . \tag{3.2}$$

Perceba que a quantidade d_i deve ser mantida inferior à capacidade máxima de armazenamento C_i do nó sensor s_i, pois caso contrário isso resultaria em perda de informação.

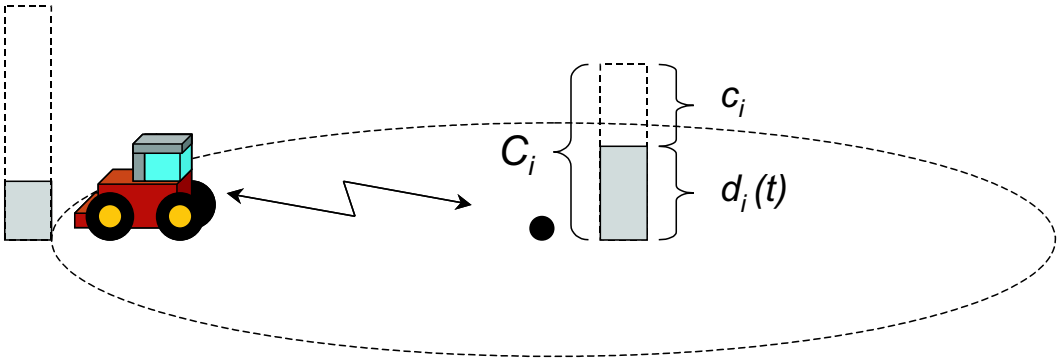

Figura 3.2: Transferência de dados entre um nó sensor e um robô. O nó sensor (pequeno ponto escuro) coleta dados do ambiente com uma taxa dada por g_i e transfere para o robô com uma taxa dada por h_i. Observe que a quantidade d_i não pode ultrapassar a capacidade máxima de armazenamento do sensor s_i dada por C_i.

É importante comentar também que

$$h_i >> g_i \,,$$

devido às características físicas dos nós sensores e dos dispositivos de comunicação.

Redes de nós sensores são conhecidas como *redes centradas em dados* porque, diferentemente de outras redes, a identificação de cada dispositivo não é importante. O agente externo que irá processar a informação coletada pelos nós sensores pode estar interessado em saber, por exemplo, qual a temperatura presente em determinada região do ambiente e não a leitura específica do nó s_i. Para o caso descrito neste trabalho, conforme pode ser observado pela equação 3.1, cada nó sensor s_i está monitorando o ambiente constantemente e estarão sempre em estado *ativo*. Dessa forma os sensores nunca irão desligar para prolongar o tempo de vida da rede. Também não é considerada a situação na qual os nós irão "morrer" em decorrência de algum evento destrutivo ou a colapso de energia. Assim, seguindo tais restrições, nenhum *modelo de dissipação de energia* [Mini et al., 2004], [Zhao et al., 2002] foi empregado visando avaliar a situação da rede. Dessa forma, não foram feitas comparações entre os resultados obtidos pelos métodos propostos (no qual os sensores estão sempre no mesmo estado) e os resultados obtidos com os sensores operando em múltiplos estados. A energia da rede seria uma variável interessante de ser analisada, pois tornaria os métodos propostos neste trabalho mais amplos e aplicáveis a situações com maior grau de realismo. Porém, conforme dito no capítulo anterior, alguns problemas estão sendo tratados apenas indiretamente e não serão analisados. Finalmente, as redes aqui consideradas são conhecidas como

Redes Estáticas porque os nós que a compõe não se movimentam pelo ambiente.

Da mesma forma, a situação na qual a ocorrência de um evento na rede é prontamente reportada para a unidade central não é tratada neste texto. Em alguns protocolos [Intanagonwiwat et al., 2000], é comum que nós sensores comuniquem a ocorrência de um evento que foi previamente assinalado como alvo da tarefa de monitoramento da rede. Por exemplo, em uma rede com a tarefa de rastrear algum objeto móvel, os nós próximos ao objeto terão estimativas melhores da posição na qual ele se encontra e poderão comunicar tal estimativa a unidade central com maior grau de confiança. Entretanto, embora não considerado, as metodologias aqui empregadas podem ser facilmente expandidas para englobar tal aspecto.

Uma última restrição foi imposta aos métodos de extração de dados propostos para este trabalho: a posição de cada nó sensor s_i é conhecida, ou tem-se uma estimativa da mesma, na pior das hipóteses. Os robôs, com essas considerações, têm dispositivos de GPS (*Global Positioning System*) [Dudek and Jenkin, 2000] de forma a se localizarem no ambiente e seguirem em direção a cada um dos nós presentes no plano.

3.1.2 Modelo percepção-ação

A autonomia empregada pelos robôs móveis para cumprir suas tarefas tem importância fundamental nos métodos que serão apresentados neste texto, de maneira que é interessante a definição de [Wooldridge and Jennings, 1995] sobre agentes inteligentes:

> "Um agente é uma entidade computacional situada em dado ambiente que é capaz de cumprir autonomamente as ações para as quais foi designado".

Neste texto, utiliza-se "agente" para designar uma entidade robótica, um agente físico atuando em um certo ambiente. A definição acima endereça ainda muitos aspectos de forma indireta como planejamento de ações, categorização de ambientes (dinâmico, estático), aprendizado, etc. Entretando, nenhum deles esta explicitamente incluso, pois os meios pelos quais a entidade poderá agir de forma autônoma poderão variar dependendo do tipo de agente, da arquitetura empregada, da constituição física (ou virtual) do agente, etc. De fato, o ponto relevante na definição anterior é *autonomia*. Um robô precisa reagir aos estímulos do ambiente por meio de um conjunto de ações bem definidas. Isso pode ser formalizado da seguinte maneira:

$$\mathcal{F}_V : E \rightarrow P\,, \tag{3.3}$$

onde a função $\mathcal{F}_V$ mapeia estados E do ambiente em percepções P.

Os estados E do ambiente serão mapeados em percepções P por meio de sensores presentes no robô, tais como sonares, câmeras, *laser*, etc. Essas percepções podem ser representadas, por exemplo, em uma base de dados por meio de lógica de primeira ordem [Russell and Norvig, 2002], mas esse assunto está fora do escopo deste texto. O mapeamento percepção-ação do robô pode ser visto, similarmente, como uma função da seguinte forma:

$$\mathcal{F}_A : P \rightarrow A\,, \tag{3.4}$$

onde a função $\mathcal{F}_A$ mapeia um conjunto de percepções P em um conjunto de ações A.

Esse mapeamento percepção-ação, aspecto central a essa seção, pode ser feito de várias formas, por meio de comportamentos pré-definidos, planejamento ou de modelos híbridos que consideram as duas abordagens. Em [Arkin, 1998] pode ser encontrado um compêndio das diversas técnicas empregadas. Para o método proposto neste capítulo o interesse recai especialmente sobre a arquitetura comportamental proposta em [Arkin, 1989], embora ela não seja estritamente relacionada à metodologia desenvolvida. Já o método de extração de dados do Capítulo 4 utilizou em uma de suas camadas o modelo comportamental proposto em [Brooks, 1986], conhecido como *Arquitetura Subsumption*.

3.1.3 Campos de potencial

O modelo comportamental de [Brooks, 1986] implementa a função $\mathcal{F}_A$ de forma discreta mapeando as percepções P em ações A a serem tomadas pelo robô. Assim, por exemplo, dependendo da tarefa que o robô deve cumprir, existem módulos como `siga-adiante`, `evite-obstáculo`, etc. que terão prioridades definidas e serão acionados quando ocorrer algum evento no ambiente (como por exemplo, a detecção, por meio de sensores, de algum obstáculo no caminho que o robô deve percorrer).

Entretanto, a função $\mathcal{F}_A$ pode mapear percepções P em ações A de forma contínua. Essa abordagem pode ser exemplificada por *métodos baseados em campo de potencial* [Khatib, 1985], [Krogh and Thorpe, 1986]. O campo de potencial é gerado por meio de uma *função de potencial*. O emprego desses métodos permite que o robô realize a navegação pelo ambiente de forma suave.

A *lei da gravidade* é um exemplo de função de potencial que pode ser usada na geração de um campo de potencial repulsivo (embora a sua definição esteja associada

à *atração* entre corpos). Essa função pode ser descrita da seguinte maneira:

$$\text{Força} \propto \frac{1}{Distância^2}\,. \tag{3.5}$$

Assim, a Equação 3.5 diz que, quanto mais próximo do obstáculo, maior será a força de repulsão sentida pelo robô. Essa força decresce com o quadrado da distância.

Enquanto na *Arquitetura Subsumption* é possível discretizar os comportamentos necessários ao cumprimento de uma tarefa (por exempo, `siga-para-o-alvo` e `evite-obstáculo`) em um esquema de prioridades, a função de potencial engloba esses conceitos na geração de um campo através do qual o robô irá se mover. Dessa forma o robô consegue navegar pelo campo de potencial formado, reagindo aos estímulos do ambiente. Na Figura 3.3(a) pode-se ver um campo de potencial gerado para um ambiente no qual tem-se um obstáculo e um alvo. É possível perceber que o campo é repulsivo próximo ao obstáculo e converge para o alvo que está localizado na parte inferior da figura. A força de atração ou repulsão sentida pelo robô é dada pelo *gradiente* da função potencial. Na Figura 3.3(a)-(b), o gradiente é representado pelas setas direcionais contidas em todo o ambiente.

A geração de um campo de potencial pode ser custosa em termos computacionais, dependendo do tamanho do ambiente e da quantidade de obstáculos presentes. Por essa razão, geralmente, o robô computa o valor do campo na posição específica onde se encontra e reage instantaneamente aos estímulos do ambiente. Nenhum planejamento com base em todo o campo de potencial é produzido, todo o comportamento do robô é reduzido a um mapeamento percepção-ação contínuo. Isso pode ser considerado uma vantagem significativa, já que em ambientes dinâmicos é possível se adaptar às mudanças instantaneamente, de acordo com as percepções que o robô tiver a partir de seus sensores. Se fosse necessário computar um novo plano, a cada mudança no ambiente (e elas podem ser muitas, dependendo do ambiente), seria necessário recomputar todo o campo de potencial.

Entretanto, a utilização de métodos baseados em campos de potencial pode recair no problema de *Mínimos Locais* [Koren and Borestein, 1991]. Na Figura 3.3(b) é ilustrado um ambiente no qual existem três alvos com igual força de atração. Um robô que estivesse localizado no centro geométrico do triângulo formado pelos três pontos poderia sentir-se atraído simultaneamente por vetores cuja resultante fosse nula (ou próxima disso), causando assim uma situação de mínimo local.

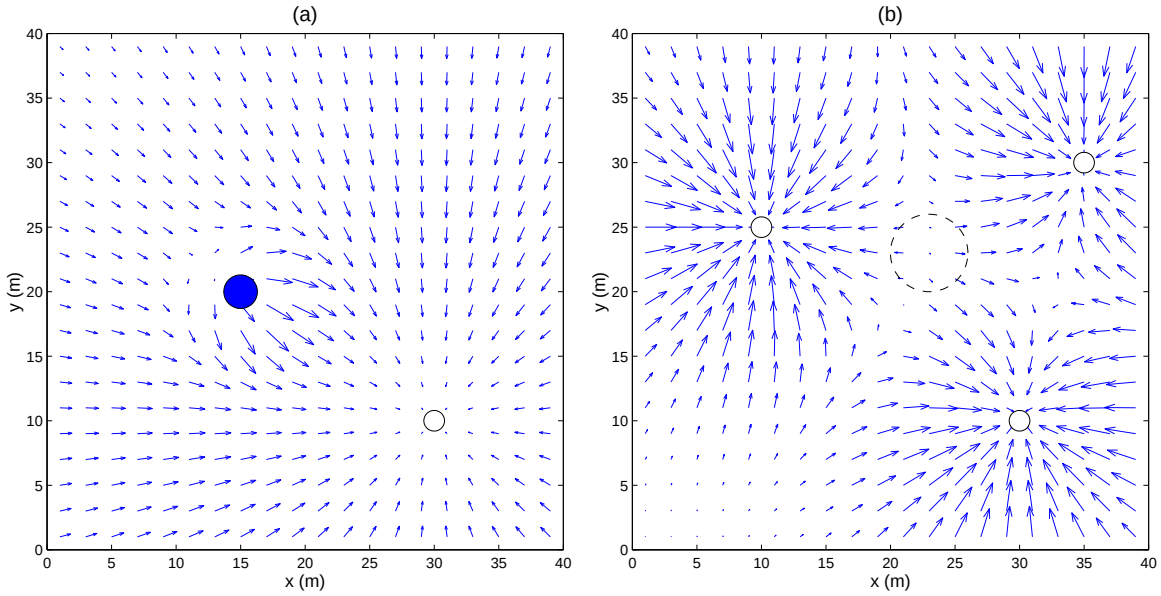

Figura 3.3: Campos de potencial em dois ambientes. (a) O obstáculo (círculo escuro) na posição $(15, 20)$ cria um campo repulsivo ao seu redor e o alvo (círculo claro) na posição $(30, 10)$ cria um campo de atração que converge para seu centro. (b) Os três pontos de atração da figura, nas posições $(10, 25)$, $(30, 10)$ e $(35, 30)$, resultam em um valor para o campo de potencial praticamente nulo no círculo pontilhado localizado na posição $(23, 23)$, indicando assim, a ocorrência de um mínimo local.

3.2 Metodologia

A tarefa que o robô deverá realizar é extrair os dados armazenados nos sensores de forma a minimizar as perdas de dados da rede. Na implementação do Método Reativo utilizaram-se funções de campo de potencial para representar a quantidade de informação coletada em cada nó sensor da rede. A idéia básica é fazer com que o robô mova-se em direção ao nó sensor que estiver com maior "urgência", isto é, com a maior quantidade de dados armazenados, pois o *gradiente* da função do nó sensor atrairá o robô com maior intensidade. Conforme pode ser visto pela Figura 3.4 o Método Reativo pode ser esquematizado em por meio de dois comportamentos:

1. `Desviar-de-obstáculo`. Implementa a detecção e o desvio de obstáculos a serem realizados pelo robô. A detecção é feita utilizando-se dispositivos de sonar.

2. `Mover-para-o-alvo`. Guia o robô em direção ao próximo nó presente em seu plano. O robô se localiza no ambiente por meio de um dispositivo de GPS.

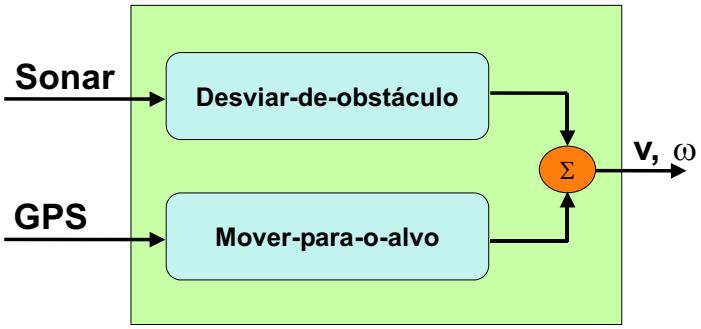

Figura 3.4: Método Reativo. Os dois módulos presentes são implementados com uma função de navegação baseada em campos de potencial. v é a velocidade linear e ω é a velocidade enviadas angular para o controle de baixo nível do robô.

Os módulos presentes são implementados utilizando uma função de navegação por campos de potencial. Dessa forma, cada módulo do sistema fornece um vetor, que é positivo para o módulo `Mover-para-o-alvo` e negativo para o módulo `Desviar-de-obstáculo`. Então, o controle aplicado ao robô, tem como entrada a soma de ambos vetores. Logo, o Método Reativo pode ser relacionado à arquitetura comportamental proposta em [Arkin, 1989].

A metodologia apresentada nesse capítulo será validada utilizando-se apenas um robô em um ambiente específico. O módulo `Desviar-de-obstáculos` não foi implementado e será considerado em trabalhos futuros. Resultados com múltiplos robôs serão apresentados no Capítulo 5 e o método por meio do qual a rede de nós poderá ser dividida entre os robôs baseia-se na metodologia a ser apresentada no Capítulo 4. Na próxima seção será mostrada a função de campo de potencial escolhida para implementar o Método Reativo.

3.2.1 Funções de base radial

O ambiente no qual o robô irá navegar é similar ao mostrado na Figura 3.3(b) no qual três alvos produzem campos de potencial atrativos. Neste trabalho, cada nó sensor s_i pode ser visto como um alvo a ser alcançado pelo robô, de forma que o campo de potencial *total* gerado no ambiente será a superposição de todos os campos de potencial gerados pelos nós sensores separadamente. De fato, para cada nó sensor no ambiente, foi modelada uma função de potencial com força de atração diretamente proporcional à quantidade de informação armazenda no sensor. A força de atração do campo de potencial gerado pelo nó sensor s_i decairá muito rapidamente quando o robô alcançar a posição na qual o nó sensor se encontra e começar a transferir os dados para si, pois

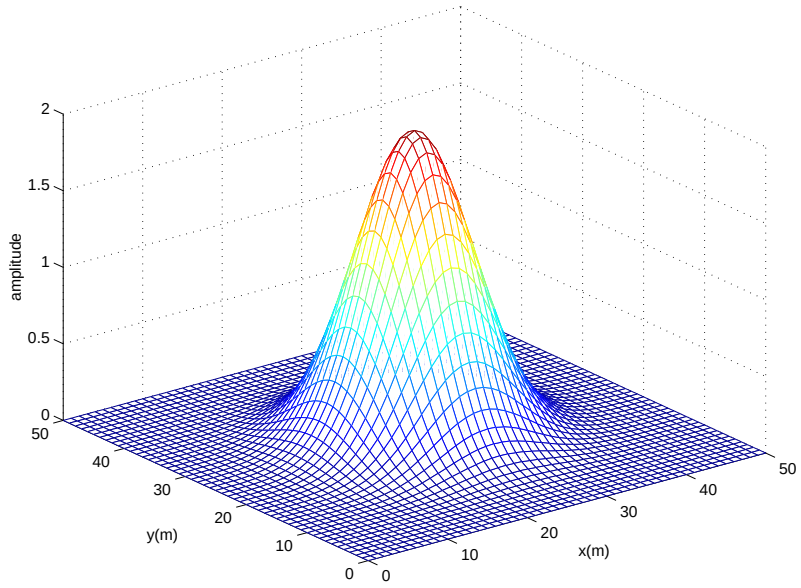

Figura 3.5: Uma Função de Base Radial no espaço 2D.

a taxa de transferência de dados h_i entre o robô e o nó sensor é muito maior que a taxa de aquisição de pacotes g_i do nó sensor.

As funções de potencial empregadas para esse método são conhecidas como *Funções de Base Radial* (RBF) [Powell, 1985]. A escolha dessas funções se deu por permitirem aumentar o *raio de alcance da função de potencial* sempre que for necessário aumentar a área de infuência do campo de potencial de um determinado nó sensor. A seguinte função de base radial é utilizada:

$$\phi_i(q_R) = \alpha_i e^{-\frac{1}{2\rho_i^2}\|q_R - q_i\|^2} \tag{3.6}$$

onde α_i é a amplitude máxima da função, ρ_i é o raio de alcance e q_i é a localização do centro da função (coordenadas do nó sensor s_i). Uma exemplo de RBF com $\alpha_i = 2$, $\rho_i = 5$ e $q_i = (25, 25)$, é mostrado na figura 3.5. A metodologia adotada fica fácil de ser compreendida se for possível visualizar o nó sensor s_i sendo colocado exatamente na posição $q_i = (25, 25)$ no plano xy da Figura 3.5. Como a base ρ_i da RBF é relacionada com a quantidade de informação contida no nó sensor s_i tem-se que:

$$\rho_i = \xi[C_i - c_i(t)]\,, \tag{3.7}$$

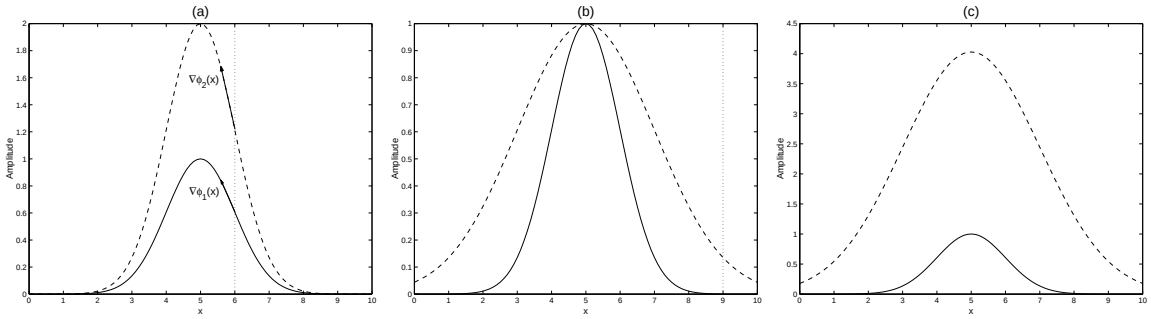

Figura 3.6: Aumentando a amplitude e o raio de uma RBF. (a) Para uma posição específica ($x = 6$), aumentando-se a amplitude da RBF corresponde a um aumento do gradiente nessa posição. (b) Aumentar o raio da RBF corresponde a aumentar a região de influência dessa função. Nessa figura, a influência (gradiente) da RBF com raio igual a 1 (linha contínua) em $x = 9$ é próximo à zero, mas torna-se consideravelmente maior se o raio é duplicado (linha tracejada). (c) O efeito de incrementar a amplitude e o raio simultaneamente na Equação 3.8: O gradiente aumenta para todo x dentro da região de influência da função. Nessa figura o valor de κ é 2.01.

e a amplitude:

$$\alpha_i = \rho_i^{\kappa}\,, \tag{3.8}$$

onde ξ é a constante de crescimento da equação linear que relaciona a quantidade de informação $C_i - c_i(t)$ armazenada no nó sensor s_i, com o raio de alcance ρ_i da função. κ é uma constante maior que 2 que relaciona a amplitude da função com seu raio. A real importância de κ será explicada ao final dessa seção.

A força de atração com a qual o robô será compelido a se mover em determinada direção é dada pelo gradiente da RBF em um determinado ponto. O gradiente em um ponto específico q_R é dado por:

$$\nabla\phi_i(q_R) = -\alpha_i \frac{\|q_R - q_i\|}{\rho_i^2}\, e^{-\frac{1}{2\rho_i^2}\|q_R - q_i\|^2} \tag{3.9}$$

A idéia de se usar RBF pode ser melhor entendida observando-se a Figura 3.6. Quando a quantidade de informação armazenada em um determinado nó sensor aumenta, é necessário que (i) a força de atração em um ponto e (ii) a área de influência aumentem. Esses dois objetivos podem ser atingidos alterando-se os valores do raio de alcance e da amplitude da função de acordo com as equações 3.7 e 3.8. Aumentando-se o raio da RBF (que se relaciona diretamente à quantidade de informação no nó sensor) é possível aumentar a área de influência da função, incluindo, assim regiões que antes não eram influenciadas pelo campo de potencial de um determinado nó sensor s_i. A

amplitude precisa ser aumentada de forma a garantir que o gradiente da função irá sempre aumentar quando a base da função aumentar. Como o valor de κ é maior que 2 tal requisito sempre será cumprido.

3.3 Controle do robô

Conforme foi desenvolvido nas seções anteriores, o robô irá seguir o campo de potencial gerado no ambiente. Para o Método Reativo proposto nesse capítulo, o ambiente no qual o robô irá atuar não contém obstáculos, que poderiam gerar gradientes em direções opostas e assim "prender" o robô em um mínimo local. Entretanto, existem múltiplos alvos (nós sensores), em uma maneira similar à mostrada na Figura 3.3(b), o que também pode propiciar a ocorrência do problema. Assim, o controle mostrado no Algoritmo 1 [Pereira et al., 2004a] foi implementado de forma a evitar que o robô fique "preso" em mínimos locais. É importante ressaltar que foram usados robôs holonômicos, de forma a simplificar o controle.

Para que o passo 1 do Algoritmo 1 possa ser realizado o robô precisa ter uma estimativa inicial da quantidade de dados em cada um dos nós sensores da rede. Essa estimativa pode ser facilmente obtida, já que a quantidade de informação em um nó sensor cresce de acordo com uma equação linear no tempo (conforme visto neste capítulo) e o robô pode verificar se a estimativa calculada para um determinado nó sensor s_i está correta sempre que se comunicar com o mesmo, atualizando sua equação de predição caso haja divergência. O passo 2 seleciona o maior de todos os gradientes, o que abilita o robô a seguir na direção desse vetor para coletar os dados do nó sensor que estiver com maior quantidade de informação acumulada. Uma alternativa interessante (e uma possível melhoria) pode ser obtida por meio do passo 3. Sabendo-se que o robô não precisa chegar na posição q_i do nó sensor s_i (basta estar dentro do raio de comunicação r_i), calcula-se o vetor resultante de todos os gradientes que estiverem no mesmo semi-plano de $\nabla\phi_{max}(q_R)$. Dessa forma o robô pode seguir trajetórias que interceptam o raio de comunicação de vários sensores, extraindo dados da rede de forma mais eficiente. Além disso, o vetor resultante nunca será nulo visto que todos os gradientes que estiverem no semi-plano oposto ao semi-plano de $\nabla\phi_{max}$ serão desconsiderados. Logo, não haverá ocorrência de mínimos locais.

O Algoritmo 1 pode ser melhor explicado observando-se a Figura 3.7. Suponha que o robô esteja sobre a influência de quatro nós sensores. O maior gradiente $\nabla\phi_{max}$ é devido ao nó sensor s_2 ($\nabla\phi_{max} = \nabla\phi_2(q_R)$). Já que $\nabla\phi_1(q_R)$ e $\nabla\phi_4(q_R)$ estão no

Algoritmo 1 Controle aplicado em um robô holonômico.

1. calcular os gradientes das funções de campo de potencial de cada nó sensor s_i no ambiente.

2. selecionar, dentre todos, o maior gradiente. Isso pode ser representado por meio da seguinte Equação:
$$\nabla\phi_{max}(q_R) = \max(\nabla\phi_i(q_R), 1 \leq i \leq n)\,, \tag{3.10}$$
onde max retorna o maior vetor dentre todos $\nabla\phi_i(q_R)$ e n é o número de sensores no ambiente.

3. somar todos os gradientes que se encontram no mesmo semi-plano do maior gradiente. O vetor resultante dessa adição representa a velocidade com a qual o robô holonômico deve se mover para chegar em um ponto específico. O vetor é calculado da seguinte forma:
$$u = \nabla\phi_{max} + \sum_{\nabla\phi_i \in S} \nabla\phi_i(q_R)\,, \tag{3.11}$$
onde S é o conjunto de todos os vetores no mesmo semi-plano de $\nabla\phi_{max}$, sendo definido como:
$$S = \{x | x \cdot \nabla\phi_{max} > 0, x \neq \nabla\phi_{max}\}\,. \tag{3.12}$$
Assim, o robô recebe como entrada:
$$v = \psi\frac{u}{\|u\|}\,, \tag{3.13}$$
onde ψ é uma constante positiva que determina a velocidade do robô.

mesmo semi-plano de $\nabla\phi_2(q_R)$ (a região sombreada na Figura 3.7(a)), eles são usados para obter o vetor resultante. O gradiente $\nabla\phi_3$ é ignorado já que está no semi-plano oposto ao semi-plano de $\nabla\phi_{max}$. O vetor resultante v tem um componente orientado na direção de $\phi_2(q_R)$. Uma vez que $\nabla\phi_4(q_R)$ é maior que $\nabla\phi_1(q_R)$, o robô irá mover-se em uma trajetória que se aproxima mais de s_4 do que do sensor s_1. Dessa forma, a trajetória realizada pelo robô irá interceptar o raio de comunicação de s_4. A medida que o robô se aproxima de s_2 (Figura 3.7(b)), os gradientes relativos aos outros sensores são ignorados, forçando o robô a mover-se na direção indicada por $\nabla\phi_2(q_R)$ ($\nabla\phi_{max} = \nabla\phi_2(q_R) = u$). É importante observar que o robô irá permanecer dentro do raio de comunicação do nó sensor s_i enquanto o gradiente relativo à RBF desse sensor se mantiver maior que os gradientes relativos aos outros nós sensores. À medida que o robô transfere os dados para si, esse gradiente torna-se menor. Assim, em determinado instante, o robô será atraído para outro nó sensor que possuir um gradiente de maior intensidade (ou seja, maior quantidade de dados armazenados). Logo, não necessaria-

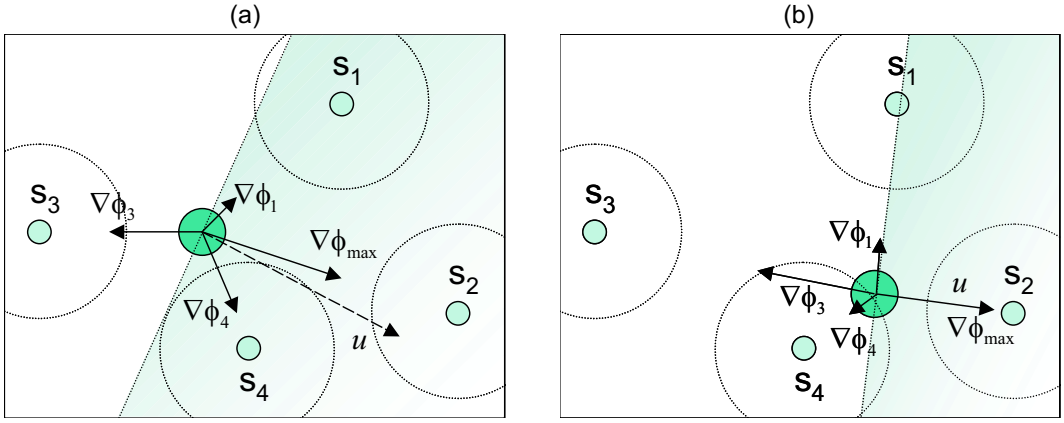

Figura 3.7: Um robô (círculo grande escuro) é influenciado por quatro nós sensores. O sensores são representados por círculos pequenos. O raio de comunicação dos nós sensores é representado por círculos pontilhados. O vetor final u é composto por vetores no mesmo semi-plano do maior gradiente, $\nabla\phi_{max} = \nabla\phi_2$. (a) Embora o robô se mova em direção a s_2, o vetor obriga o robô a interceptar o raio de comunicação do sensor s_4. (b) Mínimo Local é evitado se os vetores que não pertencem ao mesmo semi-plano de $\nabla\phi_{max}$ ($\nabla\phi_1$, $\nabla\phi_3$ e $\nabla\phi_4$) forem ignorados.

mente, o robô extrairá a quantidade total de dados do sensor s_i antes de se mover em direção à outro nó sensor.

Entretanto, dois problemas podem ser observados com esse método: i) se as funções de potencial empregadas tiverem restrições no tamanho da base, alguns nós sensores nunca serão atendidos e irão "morrer por inanição" (*starvation*), [Tanenbaum, 1995] e, complementarmente, ii) se a razão com a qual os nós sensores coletam dados do ambiente for muito alta o robô pode sentir-se atraído para um grupo de sensores e permanecer nessa região indefinidamente (*live-lock*).

A complexidade do Algoritmo 1 é de $O(n)$, como pode ser facilmente constatado nos passos 1, 2 e 3. O algoritmo é executado constantemente à medida que o robô navega pela área onde estão espalhados os nós sensores. Dessa maneira, em K iterações do algoritmo, o custo computacional é dado por Kn.

3.4 Exemplo de funcionamento

As figuras 3.8 e 3.9 mostram passos de uma simulação onde um robô holonômico está coletando dados de um rede composta por 5 nós sensores em um ambiente de 20 m × 16 m. O raio de comunicação r_i dos nós sensores foi configurado para 3m. A

razão de crescimento de dados g_i nos nós sensores é 25 vezes menor que a razão de transferência de dados h_i entre o nó sensor e o robô. Essa razão entre g_i e h_i foi utilizada em todos os experimentos realizados neste texto, e não foi baseada em parâmetros reais.

A Figura 3.8 mostra a posição e a trajetória percorrida pelo robô no plano, enquanto a Figura 3.9 mostra as funções de potencial correspondentes a cada um dos nós sensores do ambiente. Na Figura 3.8(c), o robô poderia ter optado por seguir uma trajetória em linha reta partindo do nó sensor s_3 em direção ao nó s_4. Entretanto, conforme pode-se notar nas Figuras 3.8(d)-(e), o robô realiza uma trajetória curva por meio da qual intercepta o raio de comunicação do nó sensor s_5 que, então, transfere seus dados para o robô. Isso occorre porque a resultante das soma dos vetores considera todos os gradiente contidos no mesmo semi-plano de $\nabla\phi_{max}$. Complementarmente, a Figura 3.9(d)-(e) mostra que o maior de todos os gradientes é realmente devido à função de campo de potencial do nó sensor s_4, no instante observado.

Nas Figuras 3.8(e)-(f) pode-se observar que o robô entra no raio de comunicação do sensor s_4, coleta seus dados e imediatamente sente-se atraído pelo nó s_2. Novamente, o robô intercepta o raio de comunicação do nó sensor s_5 ao mover-se em direção ao nó s_2. Resultados de simulações realizadas, sobre a quantidade de dados coletados pelo robô, serão apresentados no capítulo 5.

Outros exemplos com 2, 3, 10 e 60 nós foram realizados com sucesso e podem ser vistos em `http://www.verlab.dcc.ufmg.br/sensornet`.

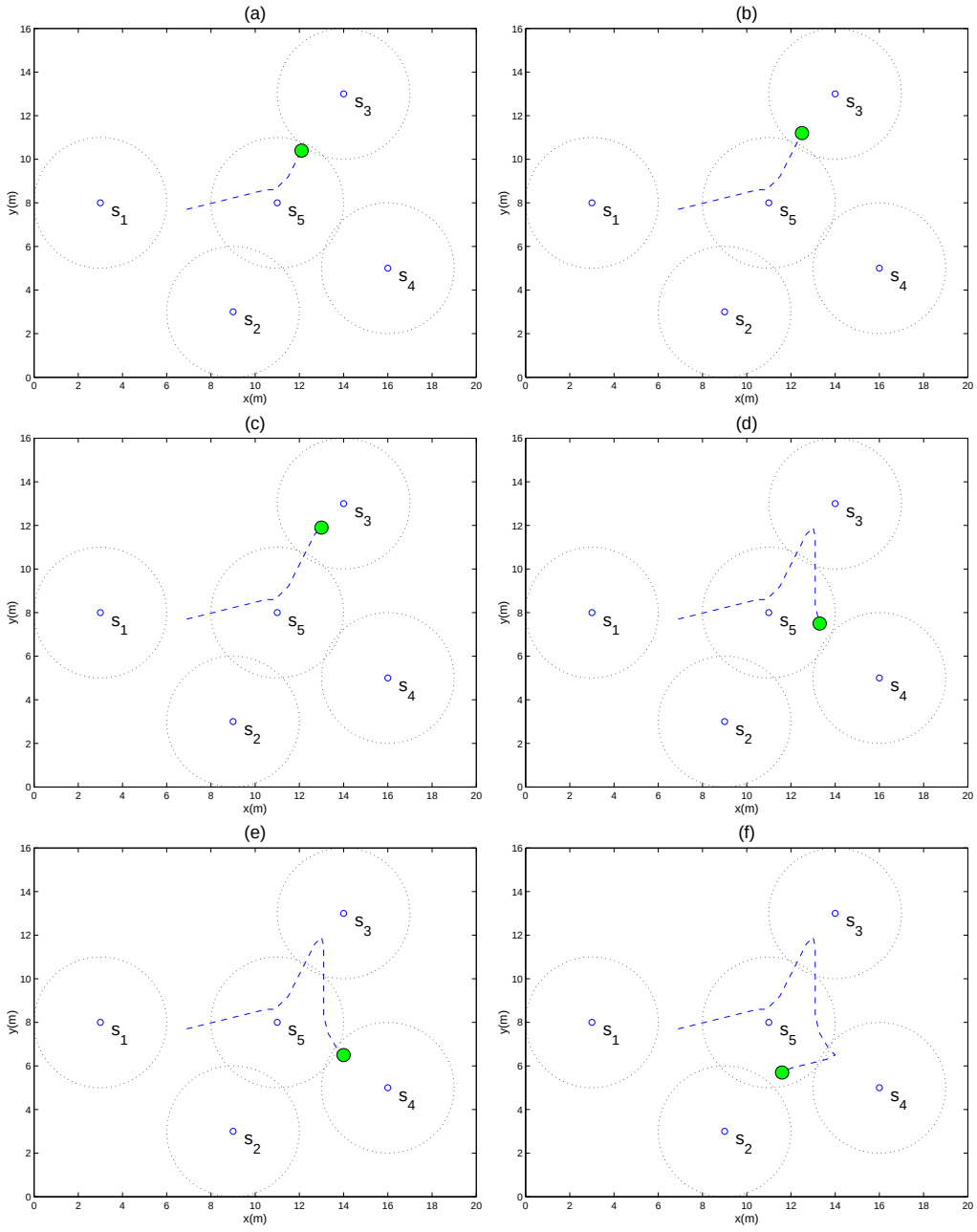

Figura 3.8: Extração de dados de uma rede de 5 nós sensores por um robô holonômico. (a)-(c) o robô move-se em direção ao nó s_3 com maior quantidade de informação. (d) Após coletar a informação armazenada em s_3 o robô move-se em direção ao nó s_4, mas executa uma "trajetória curva" através da qual entra no raio de ação do nó sensor s_5. (e)-(f) O robô parte em direção ao nó s_2, atraído pelo campo de potencial gerado pelo nó sensor.

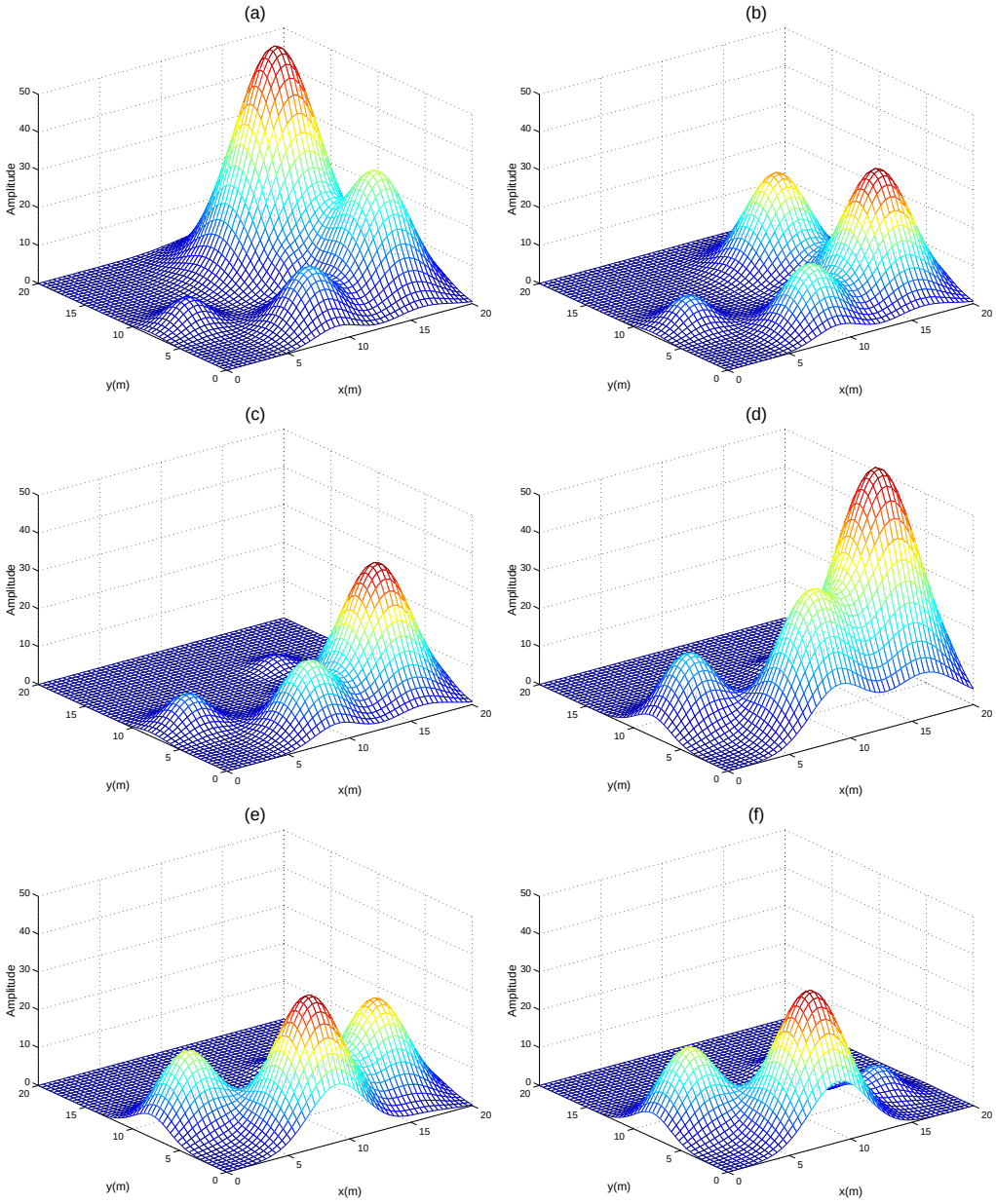

Figura 3.9: Funções de potencial relativas ao experimento mostrado na Figura 3.8. Pode-ser perceber que quanto mais informação armazenada no nó sensor maior é o raio e amplitude de sua função de potencial. (a)-(c) Quando o robô entra no raio de comunicação do nó sensor s_3 o raio e a amplitude da função diminuem, indicando que a transferência de informação está sendo realizada.

Capítulo 4

Método Híbrido

O segundo método de extração de dados que será apresentado neste texto teve sua metodologia inspirada na área de otimização combinatória. A idéia é planejar a rota de cada um dos m robôs que irão atuar no ambiente, com base nas capacidades dos robôs, na capacidade de cada um dos nós sensores e nas distâncias relativas entre os sensores do ambiente. O plano gerado para cada um dos robôs visa representar uma solução que minimize variáveis importantes tais como trajetória percorrida, tempo de execução e indiretamente perda de dados na rede. Tornar o custo mínimo (ou próximo disso) requer o uso de heurísticas, já que se trata de um problema $\mathcal{NP}$-*difícil* [Cormem, 2001]. A próxima seção apresenta em detalhes a definição formal do problema.

4.1 O Problema do Roteamento de Veículos

A modelagem foi feita baseando-se no *Problema do Roteamento de Veículos (Vehicle Routing Problem - VRP)* [Dantzig and Ramser, 1959]. A motivação inicial desse problema não tem relação alguma com o cenário empregado neste trabalho, mas os principais elementos (variáveis) que estão presentes na sua definição, podem ser facilmente mapeados para o problema aqui abordado.

No VRP tem-se um conjunto de *clientes* e um conjunto de *veículos* com a tarefa de atender esses clientes. Pode-se imaginar uma situação na qual múltiplos caminhões devem fazer entrega de materiais para clientes localizados em pontos específicos em uma cidade. Sabendo que i) a capacidade desses caminhões não pode ser excedida e ii) todos os clientes devem ser visitados uma única vez, o problema é achar uma rota para cada caminhão de forma que a soma dos custos associados a cada rota seja minimizado.

O VRP é um problema largamente estudado e mais detalhes podem ser encontrados nos trabalhos de [Dantzig and Ramser, 1959], [Bodin, 1983], [Hjorring, 1995], [Aronson, 1996], [Laporte and Semet, 1999], [Díaz, 2003]. O VRP simples admite apenas uma unidade central (*depot*) de onde os veículos partem e para onde devem voltar após terminarem suas tarefas. Existem variações desse problema que admitem a formulação com mais de uma unidade central. Também existem problemas nos quais algumas restrições são flexibilizadas, por exemplo, permitindo que cada veículo visite clientes mais de uma vez, ou problemas nos quais restrições são adicionadas, como no caso em que os veículos precisam chegar aos clientes respeitando um período de tempo pré-estabelecido (*time-windows*). Essas variações não foram consideradas neste trabalho.

O VRP pode ser facilmente relacionado com o problema tratado nesse texto. O conjunto de veículos é mapeado diretamente para o conjunto $\mathcal{R}$ de robôs. Da mesma forma, o conjunto de clientes é mapeado para o conjunto $\mathcal{S}$ de sensores. Lembrando-se, ainda, que a máxima capacidade de armazenadamento do nó sensor s_i, conforme definido na Seção 3.1, é dada por C_i, pode-se, formalmente definir o VRP da seguinte maneira [Aronson, 1996]:

Considere a instância $I = (G, W, Q, C)$ onde:

1. $G = (\mathcal{S}', \mathcal{A})$, onde G é um grafo, $\mathcal{S}' = \{s_0\} \cup \mathcal{S}$, $\mathcal{A}$ é o conjunto de arestas ligando os elementos de $\mathcal{S}'$, sendo que $|\mathcal{S}'| = (n+1)$ e s_0 é a unidade central;

2. $W = \begin{bmatrix} w_{00} & \cdots & w_{0n} \\ & \vdots & \\ w_{n0} & \cdots & w_{nn} \end{bmatrix}$ é uma matriz de custos $(n+1) \times (n+1)$ onde cada elemento w_{ij} representa o custo associado à aresta que liga o nó i ao nó j.

3. $Q = [Q_1, \cdots Q_m]$ é um vetor de m elementos representando as capacidades associadas aos robôs.

4. $C = [C_1, \cdots C_n]$ é um vetor de n elementos representando as capacidades máximas dos nós sensores.

A solução para o problema consiste em achar uma matriz X, de dimensão $(n+1) \times$

$(n+1) \times m$, onde cada elemento x_{ijk} tem o seguinte significado:

$$x_{ijk} = \begin{cases} 1, & \text{se o robô } R_k \text{ visita o nó } s_j \\ & \text{imediatamente após ter visitado} \\ & \text{nó } s_i \\ 0, & \text{de outra maneira} \end{cases},$$

de forma que o somatório

$$\sum_{i,j=0}^{n} w_{ij} \sum_{k=1}^{m} x_{ijk}\,,$$

seja minimizado. As seguintes restrições devem ser consideradas, para a solução do problema:

$$\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ijk} = \begin{cases} 1, & 1 \leq i \leq n \\ m, & i = 0 \end{cases}, \tag{4.1}$$

$$\sum_{i,j=1}^{n} C_i x_{ijk} \ \leq Q_k,\ 1 \leq k \leq m\,. \tag{4.2}$$

A restrição representada pela Equação 4.1 especifica que todos os nós sensores podem ser visitados por apenas 1 robô. A restrição representada pela Equação 4.2 especifica que a soma das capacidades C_i dos nós sensores visitados por um robô não pode ultrapassar a capacidade Q_k desse robô.

Conforme mencionado anteriormente, as soluções para esse problema, baseiam-se em heurísticas. Duas delas foram analisadas e testadas com o objetivo de validar a metodologia que será apresentada a seguir.

4.2 Metodologia

Para solucionar o problema foi construído um planejador centralizado que engloba em sua implementação a modelagem do problema apresentado na seção anterior. O plano final gerado leva em conta as capacidades Q dos robôs, as demandas C dos nós sensores e as distâncias relativas entre eles (matriz W). O plano consiste no conjunto de nós atribuídos a cada um dos robôs e a ordem na qual eles devem ser visitados. Como a geração do plano leva em consideração as capacidades dos robôs, pode ser necessário que eles retornem muitas vezes à unidade central, para transmitir as informações coletadas antes de prosseguir com a execução do plano.

Conforme pode ser visto na Figura 4.1, foram implementadas duas camadas: i) *Ca-*

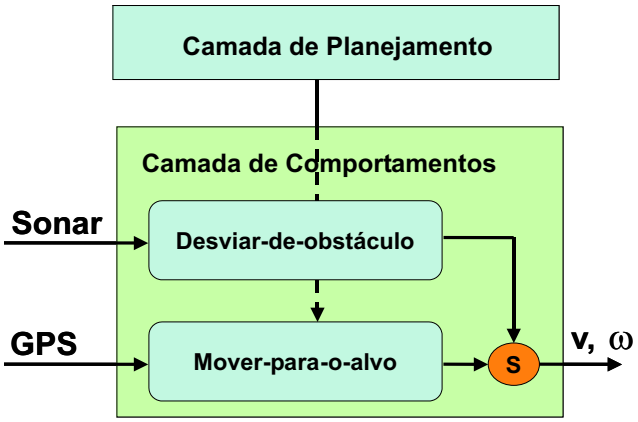

Figura 4.1: As duas camadas presentes no Método Híbrido. A Camada de Planejamento gera o plano que será enviado para os robôs. A Camada de Comportamentos implementa as ações básicas a serem executadas diante do plano e das leituras sensoriais. v e ω são as velocidades linear e angular enviadas para o controle de baixo nível do robô.

mada de Planejamento, que é responsável por gerar o plano que será utilizado por todos os robôs (essa camada está presente unicamente na unidade central); ii) *Camada de Comportamentos* que recebe como entrada o plano da camada superior e as leituras de sonar e GPS e devolve as velocidades linear v e angular w para os robôs (essa camada está presente apenas nos robôs). O esquema representado na Figura 4.1 é independente do tipo de robô empregado, seja ele holonômico ou não-holonômico, e de fato, ambas configurações foram testadas com sucesso.

As duas camadas serão mostradas em maiores detalhes nas seções 4.2.1 e 4.2.2.

4.2.1 Camada de Planejamento

A palavra *planejamento* pode assumir diferentes significados em diferentes contextos. Em inteligência artificial (IA), para um dado conjunto de *estados iniciais* de um sistema qualquer, o objetivo é encontrar um caminho que conduza até um *estado final* a partir de um conjunto de *ações* pré-definidas. Como exemplo, pode-se citar a linguagem STRIPS [Fikes and Nilsson, 1971] criada para controlar o robô *Shakey* do Laboratório de Inteligência Artificial de Stanford (*Stanford Research Institute* [Stanford Research Institute, 2004]). Embora a aplicação final esteja na área de robótica, a criação do plano utiliza os mesmos ingredientes e conceitos relativos a outros problemas de inteligência artificial, como o *mundo dos blocos* [Winston, 1970]. A idéia é discretizar o conjunto de ações a serem executadas pelo agente, como por exem-

plo, um robô. Assim, para o caso de um robô que tivesse que navegar e pegar blocos espalhados em um ambiente, poderiam existir ações descritas na sintaxe STRIPS como `(ir-para-posicao ?p)`, `(pegar-bloco ?b)`, `(largar-bloco ?b)`, onde `?p` e `?b` são variáveis instanciadas no momento da execução do plano.

O *planejamento de trajetórias* [LaValle, 2004] ocupa-se com a geração de um caminho para que o robô consiga navegar em direção a um alvo. Tais planos devem modelar o espaço de configurações $\mathcal{C}$ do robô (incluindo obstáculos) e as possíveis transformações geométricas necessárias para o cumprimento da tarefa. Nesse nível, aspectos como dinâmica e controle não são endereçados.

A Camada de Planejamento proposta neste trabalho não guarda similaridade com o formalismo STRIPS (*STRIPS-like*). Da mesma forma, é fracamente relacionado com os algoritmos estudados em planejamento de trajetórias, embora exista a preocupação em gerar trajetórias otimizadas. De fato, não é aplicada nenhuma modelagem geométrica de obstáculos e de outros elementos contidos no espaço de configuração $\mathcal{C}$ do robô. Assim, a Camada de Planejamento pode ser vista como um *tomador de decisão centralizado* por meio do qual é possível atribuir nós sensores à robôs e estabelecer a ordem na qual esses nós devem ser visitados em cada grupo.

A Camada de Planejamento restringe-se a aplicar uma função $\mathcal{F}_D(\mathcal{R}, \mathcal{S}, W, Q, C)$ onde $\mathcal{R}$ e $\mathcal{S}$ foram definidos na seção 1.2 e W, Q, C foram definidos na seção 4.1. A saída da função consiste em dois conjuntos:

$$\mathcal{G} = \{(R_k, s_i) \mid R_k \in \mathcal{R} \wedge s_i \in \mathcal{S}'\} \text{ e} \tag{4.3}$$

$$\mathcal{P} = \{(s_i \prec s_j) \mid s_i \in \mathcal{S}' \wedge s_j \in \mathcal{S}'\}, \tag{4.4}$$

onde $i \neq j$, $0 \leq i, j \leq n$ e $1 \leq k \leq m$. O conjunto $\mathcal{G}$ especifica os agrupamentos formados na rede, ou seja, quais nós sensores foram atribuídos para cada robô, enquanto o conjunto $\mathcal{P}$ especifica a ordem ($\prec$ indica precedência e $\wedge$ é o operador lógico *and*) na qual os nós devem ser visitados em cada um dos agrupamentos.

As rotas a serem realizadas pelos robôs nos agrupamento contidos em $\mathcal{G}$ são obtidas por meio da operação de *concatenação de rotas* que pode ser esquematizada como:

$$(s_i \prec s_j) + (s_j \prec ... \prec s_z) = (s_i \prec s_j \prec ... \prec s_z), \tag{4.5}$$

onde $(s_i \prec s_j)$ é um elemento de $\mathcal{P}$ e $(s_j \prec ... \prec s_z)$ é uma rota formada a partir de

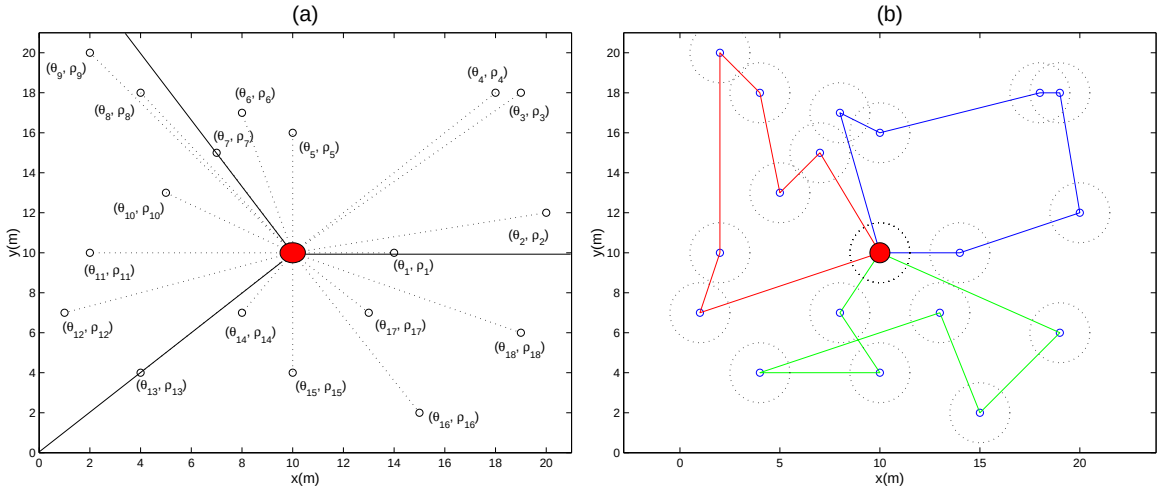

Figura 4.2: Funcionamento da Heurística de Varredura executado para 3 robôs (não mostrados na figura), com $Q_k = 6$ em um ambiente com 18 nós sensores, com $C_i = 1$. (a) Os nós são classificados de acordo com o ângulo θ_i e a distância ρ_i em relação à unidade central (círculo escuro). São gerados 3 agrupamentos que irão ser atribuídos para os três robôs. (b) As rotas geradas pela heurística baseada na Árvore Geradora Mínima para cada um dos agrupamentos.

concatenações anteriores sobre os elementos de $\mathcal{P}$.

Para que a Camada de Planejamento consiga gerar o plano, é necessário ainda, como pré-condição, que a localização exata, (ou relativa à unidade central) de todos os nós seja conhecida *a priori*, conforme estabelecido no Capítulo 3.

As duas heurísticas empregadas no processo de validação dessa metodologia foram: i) Heurística de Varredura e ii) Heurística de *Clarke & Wright*. Ambas serão explicadas em detalhes a seguir.

Heurística de Varredura. Essa foi a primeira heurística (*Sweep Algorithm*, [Gillet and Miller, 1974]) escolhida para testes [Soares et al., 2004], devido à simplicidade conceitual e porque em sua resolução ficam evidentes os dois principais subobjetivos do planejador: agrupamento de nós (conjunto $\mathcal{G}$) e criação de rotas internas (obtidas a partir do conjunto $\mathcal{P}$) nos agrupamentos. As rotas são encontradas a partir da solução do Problema do Caixeiro-viajante (TSP), comentado na Seção 1.2. No TSP um agente deve visitar um conjunto de cidades, sendo o ponto de partida igual ao de chegada e tendo como restrição visitar cada cidade uma única vez. O objetivo é realizar o trajeto da forma menos custosa possível. Para este trabalho, no caso limite no qual tenha-se apenas um robô R_k e n sensores (e considerando-se que a capacidade Q_k do

robô R_k é grande o bastante para que todos os sensores possam ser atendidos) o VRP reduz-se ao TSP. Essa heurística está descrita no Algoritmo 2.

Algoritmo 2 Heurística de Varredura (*Sweep Algorithm*)

1. represente os vértices (nós sensores) em coordenadas polares com base na unidade central:
$$s_i = (\theta_i, \rho_i)\,,$$
onde $s_i \in \mathcal{S}$, $(\theta_i,\ \rho_i)$ são o ângulo e o raio de varredura e $1 \leq i \leq n$.

2. Atribua para cada robô $R_k \in \mathcal{R}$, começando pelos menores ângulos e raios dos nós, a quantidade suficiente de nós para que não seja ultrapassada a capacidade Q_k do robô. Quando a capacidade Q_k é alcançada, o algoritmo passa para o próximo robô e assim sucessivamente.

3. Para cada agrupamento gerado utilize uma heurística para resolução do TSP.

Como exemplo, considere um ambiente com 18 nós e 3 robôs. A Figura 4.2(a) mostra a etapa de agrupamento realizada pela Heurística de Varredura, enquanto que a Figura 4.2(b) mostra as rotas criadas em cada um dos agrupamentos pela heurística de resolução do TSP.

Neste texto, utilizou-se para resolução do TSP, a heurística baseada na *Árvore Geradora Mínima* [Prim, 1957], que determina a partir de um grafo, a árvore de menor custo que contém todos os nós. Para o caso específico aqui apresentado, o grafo seria representado pelo conjunto $\mathcal{S}$ de nós sensores. Então, a resolução desse problema utilizando essa abordagem, consiste em dois passos: i) criar a Árvore Geradora Mínima a partir do conjunto $\mathcal{S}$ e ii) criar rotas mínimas (ou próximas do mínimo) a partir da árvore gerada.

A Figura 4.3 ilustra a criação de uma Árvore Geradora Mínima a partir de quatro nós em um grafo, utilizando esse algoritmo. Os pesos de ligação entre os nós estão indicados na figura. Na Figura 4.3(a), primeiramente é escolhido um nó inicial qualquer (nó `a`), que é adicionado a árvore. A seguir, uma linha de corte (*cut line*) é traçada sobre o grafo. Essa linha de corte identifica todas as arestas que fazem fronteira com a árvore geradora mínima (arestas `ab` e `ad`). Dentre todas as arestas que são cruzadas pela linha de corte, aquela que possui peso mínimo (aresta `ad`) é selecionada (no texto original essa aresta é chamada de *light edge*). Na Figura 4.3(b) o nó `d` presente na extremidade da aresta selecionada é adicionado à árvore. O processo repete-se até que todos os nós estejam presentes na árvore.

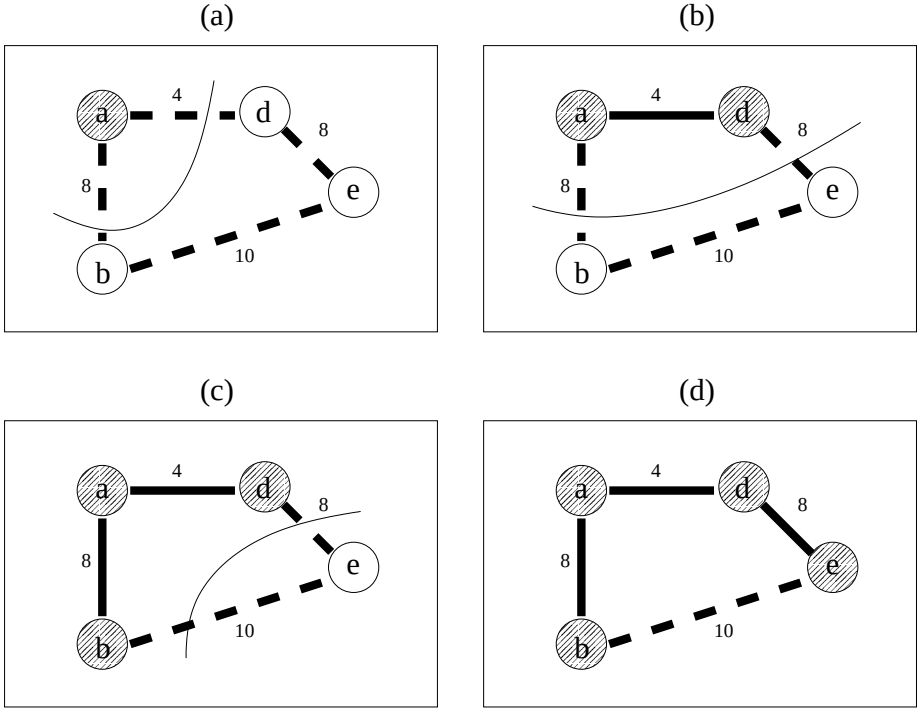

Figura 4.3: Criação da Árvore Geradora Mínima. (a) O nó `a` é adicionado à árvore e uma linha de corte cruza todas as arestas que fazem fronteira com a árvore (arestas que ligam os nós `d` e `b`. (b) o nó `d` é adicionado à árvore e outra linha de corte cruza as arestas que ligam os nós `b` e `e`. (c) O nó `b` é adicionado e a linha de corte cruza as duas arestas que ligam a árvore ao nó `e`. (d) Finalmente o nó `e` é adicionado e a árvore é gerada.

O segundo passo, então, consiste em criar as rotas a partir da árvore gerada. Esse processo está ilustrado na Figura 4.4. Os nós da Figura 4.3 foram expandidos para tornar o exemplo (que pode ser visto em [Cormem, 2001]) mais interessante. Após a criação da árvore geradora mínima, vista na Figura 4.4(b), percorre-se a árvore prefixadamente. O caminhamento prefixado, começando-se pelo nó `a` é `a`, `b`, `c`, `h`, `d`, `e`, `f`, `g`. A ordem estabelecida entre os nós estabelece a rota mostrada na Figura 4.4(c). A ordem na qual os nós sensores se encontram será representado por elementos ($s_i \prec s_j$) contidos em $\mathcal{P}$. O caminho ótimo é ilustrado na Figura 4.4(d) e difere, visivelmente, da solução encontrada.

Os passos 1 e 2 do Algoritmo 2 são executados em $O(n)$. O passo 3, usando-se a heurística baseada na Árvore Geradora Mínima, é realizado em $O(|\mathcal{A}| \log n)$ operações, onde $\mathcal{A}$ é o conjunto de arestas que ligam os nós sensores (conjunto $\mathcal{S}'$). O Algoritmo aqui implementado é muito similar ao apresentado em [Cormem, 2001], onde é feita

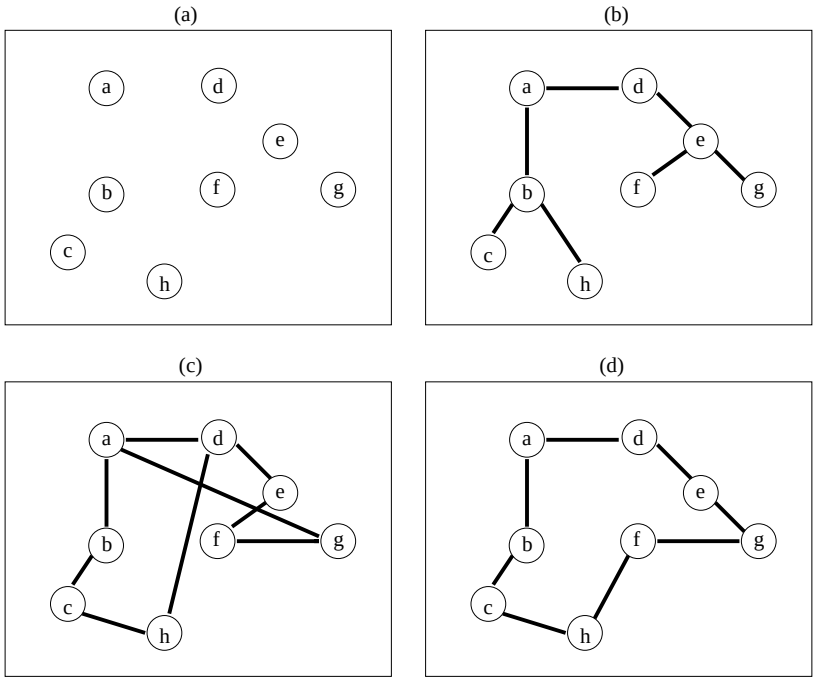

Figura 4.4: Funcionamento da heurística baseada na Árvore Geradora Mínima. (a) Conjunto de nós sensores rotulados alfabeticamente. (b) Árvore Geradora Mínima criada a partir dos nós. (c) O caminho gerado é obtido percorrendo a árvore de forma pré-fixada: `a`, `b`, `c`, `h`, `d`, `e`, `f`, `g`. (d) Pode-se perceber que o caminho ótimo é diferente do gerado pelo algoritmo.

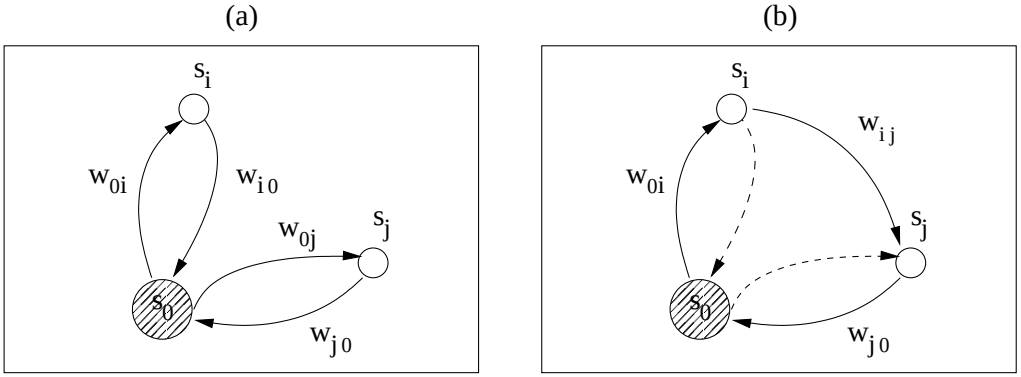

Figura 4.5: Concatenação de rotas pela Heurística de *Clarke* & *Wright*. (a) Inicialmente os nós i e j, com custos $w_{0i} + w_{i0}$ e $w_{0j} + w_{j0}$ respectivamente, são atendidos por dois veículos distintos. (b) As duas rotas podem ser concatenadas em uma única com custo $w_{0i} + w_{ij} + w_{j0}$. A rota gerada pode ser realizada por apenas um veículo.

uma cuidadosa análise de complexidade.

Algoritmo 3 Heurística de *Clarke & Wright*

1. Selecione o robô R_k, para $R_k \in \mathcal{R}$ e $k = 1$.
2. Gere n rotas $(s_0 \prec s_i \prec s_0)$, para $s_i \in \mathcal{S}$ e $i = 1, ..., n$.
3. Calcule o valor de *economia* $e_{ij} = w_{i0} + w_{0j} - w_{ij}$ para $i, j = 1, ... , n$ e $i \neq j$. Ordene o valores de *economia* em ordem decrescente.
4. Atribua uma rota $(s_0 \prec s_i \prec ... \prec s_j \prec s_0)$ ao robô R_k. Se todas as rotas já pertencem a algum robô, termine.
5. Selecione o primeiro e_{zi} ou e_{jl} da lista de *economia's*. Com essa operação, a rota atual do robô R_k será concatenada com a rota $(s_0 \prec s_z \prec s_0)$ ou $(s_0 \prec s_l \prec s_0)$.
6. Se a capacidade Q_k do robô R_k for ultrapassada, faça $k \leftarrow k + 1$ e vá para o 4, senão vá para 5.

Heurística de Clarke & Wright. A criação do conjunto $\mathcal{P}$ na Heurística de Varredura só ocorre após a criação do conjunto $\mathcal{G}$. Por outro lado, na Heurística de *Clarke & Wright* os conjuntos $\mathcal{G}$ e $\mathcal{P}$ são formados simultaneamente. Em outras palavras, a rotas em cada agrupamento são construídas à medida que os nós sensores são atribuídos aos robôs. Essa atribuição é realizada com base no conceito de *economia*, que pode ser definido como um custo gerado quando dois nós distintos são concatenados na mesma rota. Por exemplo, na Figura 4.5(a), duas rotas, $(s_0 \prec s_i \prec s_0)$ e $(s_0 \prec s_j \prec s_0)$, são apresentadas e os custos associados a cada uma delas podem ser descritos como $w_{0i}+w_{i0}$ e $w_{0j} + w_{j0}$, respectivamente (lembre-se que de acordo com a definição apresentada na seção 4.1, os custos de ligação entre os nós estão contidos na matriz W). Em ambas as rotas, o primeiro nó e o último nó são representados pela unidade central s_0. A Figura 4.5(b) mostra a concatenação das duas rotas, resultando na rota $(s_0 \prec s_i \prec s_j \prec s_0)$ com um custo $w_{0i} + w_{ij} + w_{j0}$. Então uma *economia* para os nós i e j pode ser obtido da seguinte forma:

$$e_{ij} = w_{0i} + w_{i0} + w_{0j} + w_{j0} - (w_{0i} + w_{ij} + w_{j0}) = w_{i0} + w_{0j} - w_{ij} \, .$$

A Heurística de Clarke & Wright é descrita de acordo com o Algoritmo 3. No passo 1, um robô é selecionado, e no passo 2, são geradas n rotas para cada sensor s_i da rede, começando e terminando na unidade central, s_0, de forma similar a Figura 4.5. No passo 3, o valores de *economia* e_{ij} relativos a cada par de nós sensores da rede são gerados e colocados em ordem decrescente. Essa estratégia se deve ao fato de que

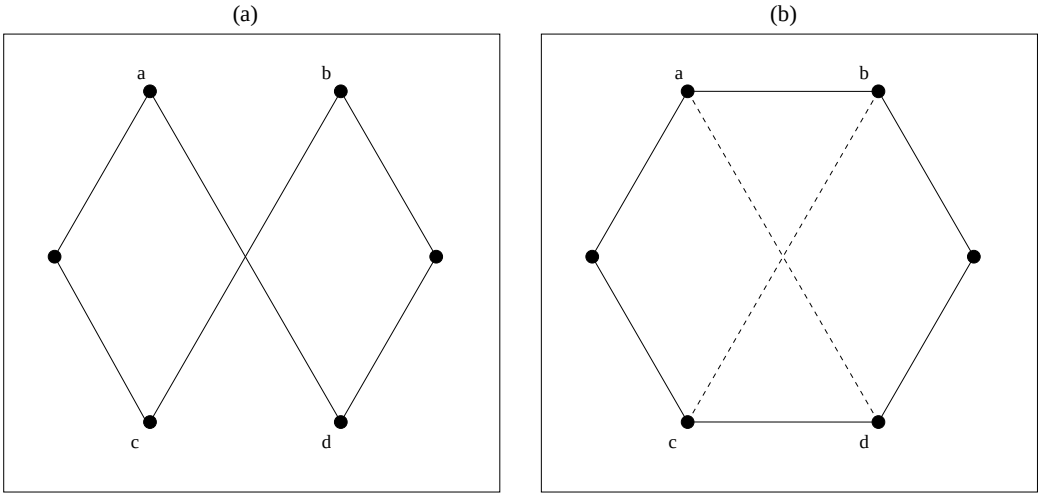

Figura 4.6: Algoritmo de otimização de rotas *2-opt*. O algoritmo funciona removendo pares de segmentos e tentando rearranjá-los em todas as combinações possíveis. (a) Considere, por exemplo a rota mostrada. (b) Dois segmentos, `ad` e `bc` são removidos e outros dois segmentos, `ab` e `cd` são gerados.

quanto maior o valor de *economia* entre dois nós, mais próximos eles estarão um do outro e, portanto, a junção de ambos em uma única rota se torna mais apropriada. No passo 4, uma rota qualquer ($s_0 \prec s_i \prec ... \prec s_j \prec s_0$) é atribuída ao robô R_k. Porém, inicialmente, todas a rotas terão a forma ($s_0 \prec s_i \prec s_0$) e só irão alcançar a representação ($s_0 \prec s_i \prec ... \prec s_j \prec s_0$) à medida que as iterações do algoritmo forem executadas. Perceba que a unidade central s_0 mantém-se como o primeiro e último nó dessa rota. Os diversos nós intermediários, são limitados à esquerda pelo nó s_i e à direita pelo nó s_j. No passo 5, com base nos valores de *economia* e_{zi} e e_{jl} uma rota será selecionada. Se a rota ($s_0 \prec s_z \prec s_0$) for selecionada, o resultado da concatenação com a rota ($s_0 \prec s_i \prec ... \prec s_j \prec s_0$) do robô R_k será ($s_0 \prec s_z \prec s_i \prec ... \prec s_j \prec s_0$). Em contrapartida, se a rota ($s_0 \prec s_l \prec s_0$) for selecionada, o resultado da concatenação com a rota ($s_0 \prec s_i \prec ... \prec s_j \prec s_0$) do robô R_k será ($s_0 \prec s_i \prec ... \prec s_j \prec s_l \prec s_0$). Finalmente, observe que, de acordo com o passo 6, a capacidade do robô pode ser um critério para que as concatenações de rotas sejam finalizadas. Cada nó sensor da rede, terá uma demanda específica C_i e cada robô móvel poderá atender um número limitado de nós sensores com base nessas demandas. Logo, quando a soma das demandas C_i de um determinado grupo de nós sensores ultrapassar a capacidade Q_k do robô, a rota em questão poderá ser "fechada" e outro robô deverá ser usado para outro agrupamento de nós.

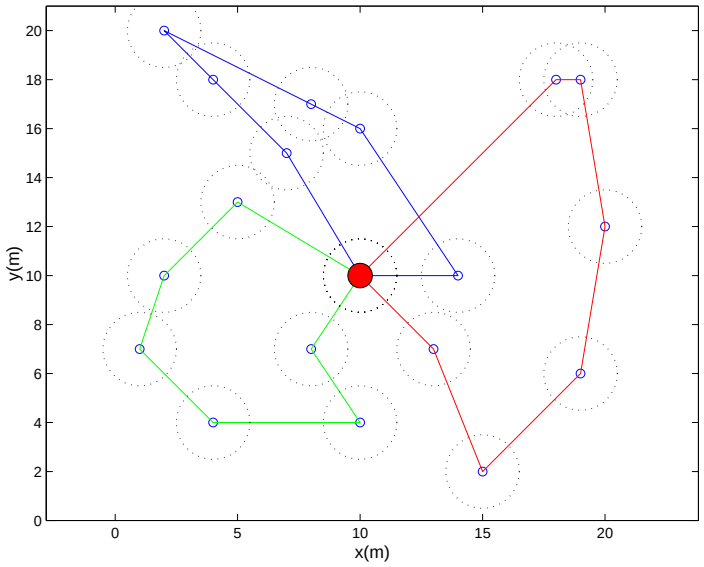

Figura 4.7: Funcionamento da Heurística de *Clarke & Wright* executado para 3 robôs (não mostrados na figura), com $Q_k = 6$ em um ambiente com 18 nós sensores, com $C_i = 1$. Após as rotas terem sido geradas elas são otimizadas com algum algoritmo de otimização de rotas como, por exemplo, *3-opt*.

Após as rotas terem sido geradas, é aplicado algum algoritmo de otimização de rotas, como o *2-opt* ou *3-opt* [Aarts and Lenstra, 1997]. Esse algoritmo é brevemente descrito na Figura 4.6. O algoritmo *2-opt* trabalha removendo *pares de arestas não-adjacentes* da rota em questão e recolocado-as em outras posições. A mudança realizada é mantida se isso representar uma melhora no custo total da rota, senão é desconsiderada. É importante dizer que isso é feito para todos os pares de arestas não-adjacentes da rota. Na Figura 4.6, as arestas `ad` e bc são removidas. Perceba ainda, por exemplo, que o vértice `a`, após a remoção só poderá ser conectado ao vértice `b`, pois a criação de uma aresta `ac` resultaria em uma sub-rota, o que não é permitido. O método *3-opt* trabalha de forma similar, mas as permutações são feitas com 3 arestas.

Os passos 1 e 2 do Algoritmo 3 têm complexidade $O(n)$. No passo 3, os valores de *economia* são gerados à um custo $O(n^2)$, e podem ser ordenados (usando-se algum algoritmo de ordenação) em $O(n^2 \log n)$. Os próximos passos consistem em analisar a lista de *economias* gerada e selecionar vértices com base nesses valores. Tal operação consome, no máximo, $O(n^2)$, visto que o tamanho da lista de economias é n^2. Assim, a complexidade do Algoritmo 3 é $O(n^2 \log n)$. Os Algoritmos 2 e 3 são executados apenas uma vez, diferentemente do Algoritmo 1 que é executado constantemente.

A Figura 4.7 mostra o resultado da aplicação da Heurística de Clarke & Wright para

o mesmo exemplo da Figura 4.2. O objetivo aqui não é fazer uma análise comparativa dos tamanhos das rotas geradas pelos algoritmos, embora isso seja de fato relevante. As duas heurísticas foram mostradas apenas como exemplos que podem ser empregados na Camada de Planejamento, mas outras heurísticas podem ser usadas. Mesmo assim, convém salientar que experimentalmente, os resultados obtidos pela Heurística de Clarke & Wright são melhores que os obtidos pela Heurística de Varredura (como será mostrado no Capítulo 5). Esse comportamento pode ser devido ao não emprego de algum algoritmo de otimização (*3-opt*) das rotas geradas pela Heurística de Varredura (após a geração de todas elas), ao passo que a Heurística de Clarke & Wright utiliza o método *3-opt* na fase final.

4.2.2 Camada de Comportamentos

O plano gerado pela Camada de Planejamento especifica os nós que devem ser visitados pelos robôs móveis em um dado ambiente. A Camada de Comportamentos foi desenvolvida para processar sinais dos sensores e enviar comandos para os atuadores segundo as informações recebidas do nível de planejamento.

Os módulos presentes na camada de comportamentos são:

1. `Desviar-de-obstáculo`. Implementa a detecção e o desvio de obstáculos a serem realizados pelo robô. A detecção é feita utilizando-se dispositivos de sonar.

2. `Mover-para-o-alvo`. Guia o robô em direção ao próximo nó presente em seu plano. O robô se localiza no ambiente por meio de um dispositivo de GPS.

Diferentemente do Método Reativo, os dois módulos da Camada de Comportamentos do Método Híbrido não foram implementados com uma função de navegação por campos de potencial. Aqui, ambos podem ser identificados como módulos da *Arquitetura Subsumption*. O módulo `Desviar-de-obstáculo` tem prioridade de execução sobre o módulo `Mover-para-o-alvo`. Um dos problemas de se utilizar esse tipo de arquitetura está no fato de que é possível que o robô fique "preso" em mínimos locais. Entretanto, nesse trabalho, tal problema foi solucionado utilizando-se a seguinte estratégia: o robô *marca* os locais por onde já passou de forma a não permanecer na mesma região por um longo período de tempo. Isso é feito por meio de uma *tabela cache* de três entradas que armazena as coordenadas x e y do robô. A tabela é atualizada sempre que o robô detecta um obstáculo. Quando o robô passa por um ponto cujas coordenas estejam presentes em alguma entrada na tabela (com uma diferença

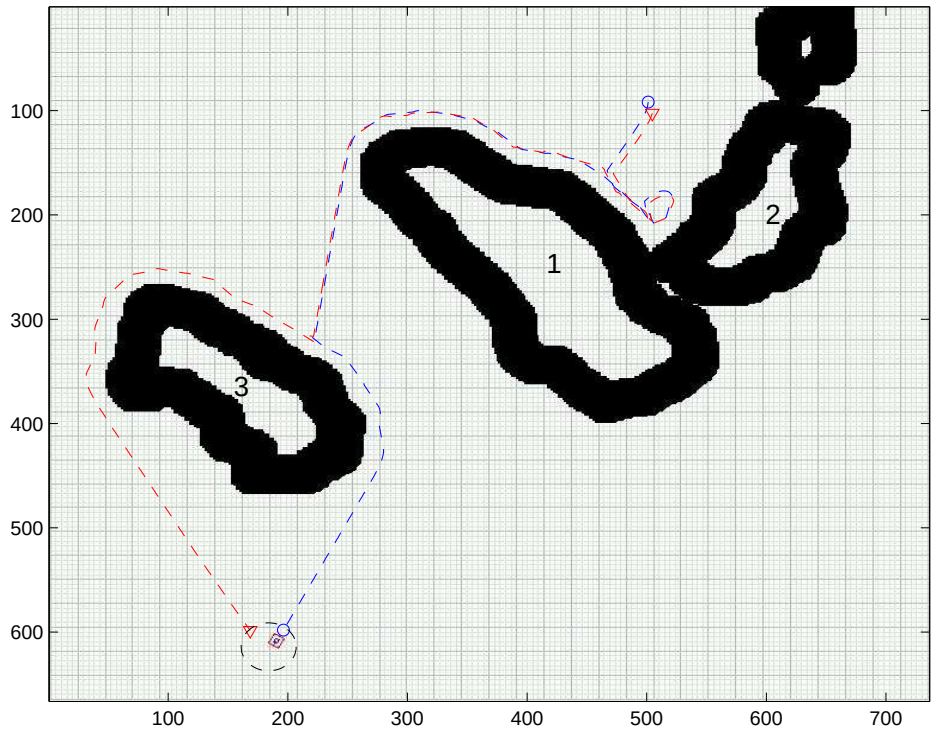

Figura 4.8: Duas trajetórias diferentes realizadas por um robô não-holonômico. O alvo está indicado por um círculo tracejado em um ambiente ocupado por 3 obstáculos grandes. O robô tem a capacidade de identificar as regiões do ambiente por onde já passou e assim consegue explorar novas áreas. Essa situação é mostrada quando o robô segue em direção oposta ao obstáculo 2, após realizar algumas trajetórias "circulares" (na região em forma de "V" entre os obstáculos 1 e 2). As duas trajetórias mostradas na figura divergem quando o robô detecta o obstáculo 3.

máxima que não ultrapasse um valor δ), o número de ocorrências para aquela entrada é incrementado. Quando o número de ocorrências em qualquer umas das três entradas atinge um determinado limite, o robô sabe que deve se afastar do local. Se Se esse limite for igual a 2, por exemplo, significa que o robô irá se afastar da região sempre que passar duas vezes pelo local. As entradas na tabela também são substituídas com base no número de ocorrências: se esse número for pequeno para determinada entrada, a mesma pode ser substituída, pois o robô deve estar afastado da região. O algoritmo de desvio de obstáculo está descrito no Anexo D.

A Figura 4.8 mostra o funcionamento da Camada de Comportamentos em um ambiente com três obstáculos para um robô não-holonômico. O objetivo do robô é alcançar a região marcada com um círculo tracejado, desviando dos obstáculos presentes no mapa. Duas trajetórias diferentes são mostradas. Pode-se perceber que o robô

parte do ponto inicial, detecta a presença do obstáculo 1 e então desvia deste, indo em direção ao obstáculo 2. Quando as leituras dos sonares não indicarem a presença de obstáculos, o robô pode ajustar sua direção para o alvo (círculo tracejado). Porém, de acordo com o que pode ser visto na figura, novamente é detectada a presença do obstáculo 1, o que mais uma vez ocasiona o desvio em direção ao obstáculo 2. Por esse motivo, o robô realiza algumas trajetórias "circulares" entre os obstáculos 1 e 2. Tal situação poderia se prolongar indefinidamente, porém, conforme foi dito no parágrafo anterior, a Camada de Comportamentos provê o robô com a capacidade de identificar as regiões pelas quais passou recentemente e dessa forma possibilita que outras áreas sejam visitadas. Assim sendo, o robô consegue contornar o obstáculo 1 e seguir em direção ao alvo. Ao detectar o obstáculo 3, as duas trajetórias mostradas divergem indicando caminhos diferentes seguidos pelos robô.

Capítulo 5

Simulações

Nesse capítulo serão mostrados resultados referentes aos métodos de extração de dados propostos nos Capítulos 3 e 4. A próximas seções apresentarão as especificações técnicas em cima das quais se basearam as simulações, bem como o ambiente escolhido para testes. A seção de resultados é dividida em duas partes: i) simulações com apenas 1 robô e ii) simulações com múltiplos robôs.

5.1 Especificações técnicas

Os resultados a serem apresentados nesse capítulo foram obtidos a partir de ambientes simulados no *software Player/Stage* [Gerkey et al., 2001], [Gerkey et al., 2003]. O Anexo A descreve brevemente tal ferramenta. Porém, é importante que as variáveis a serem usadas nas simulações sejam baseadas nas especificações técnicas dos robôs e nós sensores.

Em todos o testes realizados foram usados robôs holonômicos (a dinâmica desses robôs é dada por $v = \dot{q}$). A razão para tal escolha é devido à maior simplicidade na especificação do controle aplicado ao robô. Robôs holônomicos conseguem se mover *instantaneamente* em qualquer direção, reagindo aos comandos que lhes são passados. Além disso, protótipos de baixo custo podem ser desenvolvidos com relativa rapidez conforme descrito em [Pereira et al., 2004b]. Entretanto, os métodos de extração de dados aqui propostos podem ser adaptados para serem usados com robôs não-holonômicos, tais como robôs *pioneers* [ActivMedia Robotics, 2004]. A restrição de utilizar robôs holônomicos, aqui imposta, deve-se exclusivamente à necessidade de separar o problema do planejamento de rotas do planejamento do controle.

Os nós sensores utilizados nas simulações basearam-se em dispositivos MICA1

[Crossbow Technology, Inc., 2004b]. O MICA1 faz parte da família MICA Motes de nós sensores, desenvolvidos inicialmente pela *Universidade da Califórnia*, Berkeley. [Silva et al., 2004] apresentam diversos modelos de nós sensores, e suas especificações técnicas. As principais características do MICA1 podem ser vistas a seguir:

- Plataforma de *hardware* MPR300/310 (MPR - Mote Processor Radio);
- Plataforma de *software* MIB300 (programação e interface de rede) ;
- Processador ATMEGA103L;
- Sistema Operacional Tiny OS, que utiliza poucos recursos de *hardware*, ocupando apenas 178 bytes de memória RAM e menos de 2KB de memória flash.
- 512KB de memória *flash* para armazenar dados;
- taxa de aquisição de dados do ambiente de 40 kbits/seg.

Os nós sensores MICA Motes são comercializados pela empresa *Crossbow Technology, Inc* [Crossbow Technology, Inc., 2004a]. O MICA1 é o dispositivo mais antigo dessa geração e não está sendo mais comercializados. Os nós MICA2 e o MICAz são de tecnologia mais recente sendo que esse último, além de possuir largura de banda maior (250Kbps no caso do MICAz) utiliza o padrão 802.15.4 que visa apresentar baixo custo, baixa potência e confiabilidade.

5.2 O ambiente de testes

O ambiente no qual foram feitas as simulações é mostrado na Figura 5.1(a). O ambiente tem 70m $\times$ 70m com 60 nós sensores espalhados aleatoriamente. Como é possível perceber, não existem obstáculos no ambiente. A metodologia proposta no Capítulo 3 não considera a presença de obstáculos, de forma que optou-se pela ausência dos mesmos. Entretanto, conforme pode ser visto no Capítulo 4, o Método Híbrido foi testado em ambientes com obstáculos, validando a *Camada de Comportamentos* ali proposta.

A Figura 5.1(b) mostra o mesmo ambiente da Figura 5.1(a) no qual pode ser visto uma rota traçada para 1 único robô. Essa rota foi criada utilizando a Heurística de Clarke & Wright e tem tamanho de $380,6$m. A escolha dessa heurística para os testes realizados se deve, conforme foi dito no Capítulo 4, ao fato de esta ter apresentado

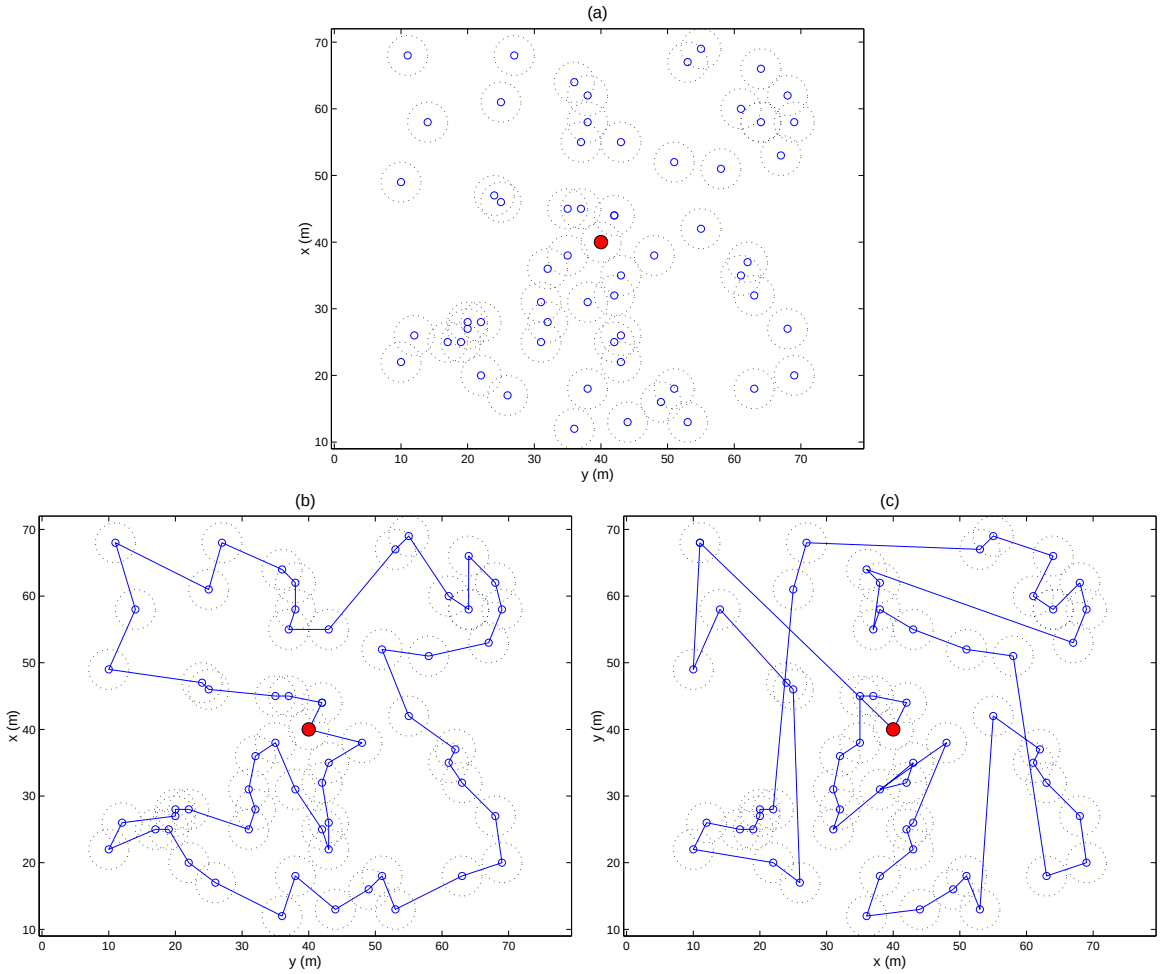

Figura 5.1: (a) Ambiente com 60 nós sensores utilizado nas simulações apresentadas nesse capítulo. (b) Rota traçada com a Heurística de Clarke & Wright. (c) Rota Traçada com a Heurística de Varredura.

melhores resultados do que a Heurística de Varredura, cuja rota não foi melhorada posteriormente por nenhum algoritmo de otimização. A rota gerada pela Heurística de Varredura, tem tamanho $538,57$, e pode ser vista na Figura 5.1(c). A Heurística de Varredura foi usada, porém, no ambiente de 18 nós sensores mostrados no Capítulo 4. Para a geração das rotas com a Heurística de Clarke & Wright, usou-se o *software* VRP Solver, descrito brevemente no Anexo B. Pode-se perceber, ainda, que para 1 robô a solução para o problema do VRP reduz-se à solução para o TSP. Os resultados mostrados na próxima seção para o Método Híbrido, foram obtidos utilizando-se essa rota.

5.3 Resultados

Primeiramente, será apresentado um conceito teórico importante para a análise dos dados que serão mostrados nessa seção. Esse conceito será chamado de *ciclo* e pode ser definido da seguinte forma:

Definição 2 *Um ciclo η_k é o período de tempo decorrido entre a saída e o retorno de um robô R_k à unidade central, após realizar a coleta de informações dos nós sensores que lhe foram atribuídos.*

O ciclo η_k do robô R_k pode ser dependente de fatores como número de nós sensores atribuídos ao robô e presença de obstáculos no caminho. No Método Híbrido as capacidades máximas C dos nós sensores e Q dos robôs foram usadas na obtenção da rota final computada pela heurística do VRP. Uma solução, talvez mais inteligente, porém não testada, seria usar as demandas dos nós sensores no instante do planejamento. Essas demandas podem ser facilmente calculadas, uma vez que a quantidade de informação em cada nó sensor s_i cresce de acordo com uma equação linear no tempo e o robô R_k sempre pode atualizar sua estimativa, caso haja divergência, ao passar pelo nó sensor s_i. O replanejamento poderia ser feito sempre que o robô R_k retornasse à unidade central após a realização de um ciclo.

Os resultados apresentados nessa seção foram obtidos sobre as seguintes variáveis:

- Eficiência t(s): tempo para realização da tarefa de extração de dados da rede;
- trajetória τ(m): trajetória percorrida pelo robô após visitar todos os nós sensores da rede;
- quantidade γ(KB): de dados perdidos na rede: quantidade total de dados perdidos pelos nós sensores da rede;
- quantidade λ(KB): de dados coletados na rede: quantidade de dados coletados pelo robô após a realização da tarefa de extração de dados.

As simulações a serem vistas na próxima seção foram realizadas 7 vezes e o resultado apresentado é a média das amostras obtidas.

5.3.1 Resultados para 1 robô

As primeiras simulações realizadas foram feitas utilizando-se apenas 1 robô no ambiente da Figura 5.1. As análises foram feitas sobre as variáveis t, τ, γ e λ. Para tanto, variou-se o raio r de alcance dos nós sensores e a velocidade v do robô.

Análise de t e τ para o Método Híbrido. A Figura 5.2(a) mostra os valores de t variando-se o raio r e a velocidade v. Pode-se perceber pelas Figuras 5.2(b)-(c) que, à medida que o raio e a velocidade aumentam, o tempo para realização de um ciclo diminui. De acordo com o resultado, o tempo t do sistema diminui mais rapidamente com o acréscimo da velocidade do que com o acréscimo do raio. A Figura 5.2(d) mostra os valores da trajetória τ variando-se r e v. Conforme esperado, com o aumento do raio r, τ diminui bruscamente. É partircularmente importante notar na Figura 5.2(e) que a trajetória máxima obtida (com raio 1.0) é menor que a trajetória total da Figura 5.1(b). A trajetória τ se mantém menor porque o robô não precisa chegar na posição exata do nó sensor, bastando entrar no raio de comunicação deste. De acordo com a Figura 5.2(f), percebe-se que a trajetória τ mantém-se aproximadamente constante com a variação da velocidade, pois para um *ciclo* a trajetória τ é igual para toda velocidades v com a qual o robô possa navegar pelo ambiente. Mesmo assim, é possível perceber uma pequena variação em τ para cada velocidade v, pois o controle do robô se torna mais difícil à medida que a velocidade v aumenta. De fato, quando o robô entra no raio de comunicação do nó sensor s_i, a mudança de direção para o próximo nó sensor s_{i+1} é efetuada mais rapidamente para velocidades menores devido à inércia. Quanto maior for a velocidade v, mais o robô terá penetrado no raio de comunicação do sensor s_i e, portanto, maior será a trajetória τ.

Análise de γ e λ para o Método Híbrido. Os resultados da Figura 5.3 não foram obtidos em apenas 1 ciclo. Para a análise de γ e λ o robô atuou no ambiente durante um período de tempo pré-determinado. A análise da trajetória τ em um ciclo permite observar se o tamanho da trajetória percorrida pelo robô corresponde à trajetória computada pela heurística do VRP. No que diz respeito à variável t, é óbvia a escolha por um ciclo, já que para um tempo pré-definido os resultados manter-se-iam sempre constantes. A Figuras 5.3(a)-(c) e 5.3(d)-(e) mostram a quantidade de dados perdidos γ e a quantidade de dados coletados λ, variando-se o raio r e a velocidade v. Conforme esperado, pode-se constatar que, com o aumento do raio e velocidade, γ diminui e λ, ao contrário, aumenta. Com raios maiores o robô pode simultaneamente penetrar no raio de comunicação de vários sensores, coletando mais dados e diminuindo as perdas na rede. Com o aumento da velocidade, o robô consegue visitar uma maior quantidade de nós sensores em um mesmo período de tempo. Com base nesses resultados, como exemplo para a velocidade $v = 1\text{m/s}$ e $r = 3\text{m}$, a quantidade γ obtida foi 92352 KB enquanto que a quantidade λ foi 56823 KB. Então é possível considerar o ganho φ_h e

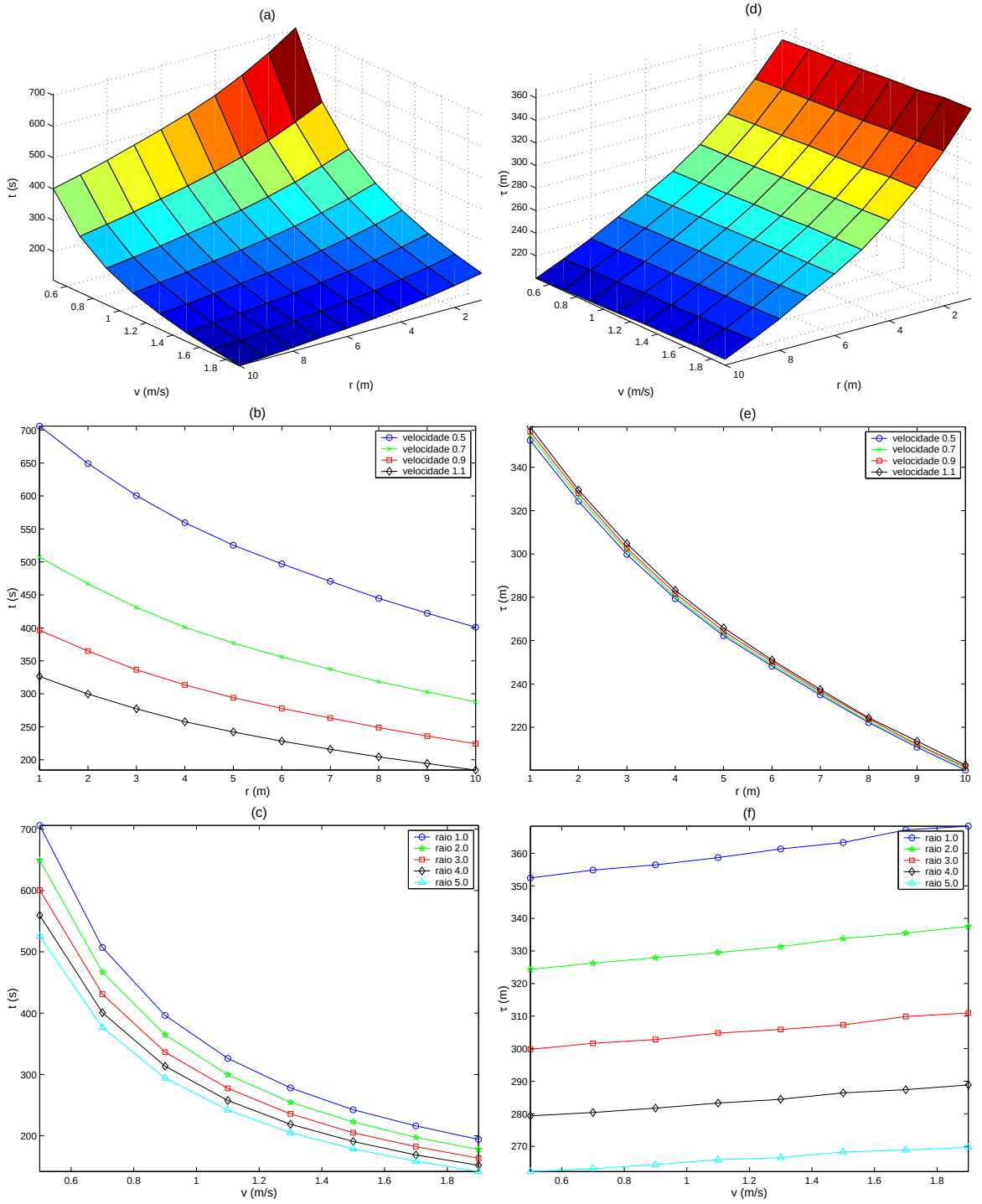

Figura 5.2: Análise de t e τ em 1 ciclo para o Método Híbrido. (a)-(c) Aumentando-se o raio r e velocidade v, t decresce. (d)-(f) Aumentando-se o raio r, τ decresce, enquanto se mantém praticamente constante com a variação de v.

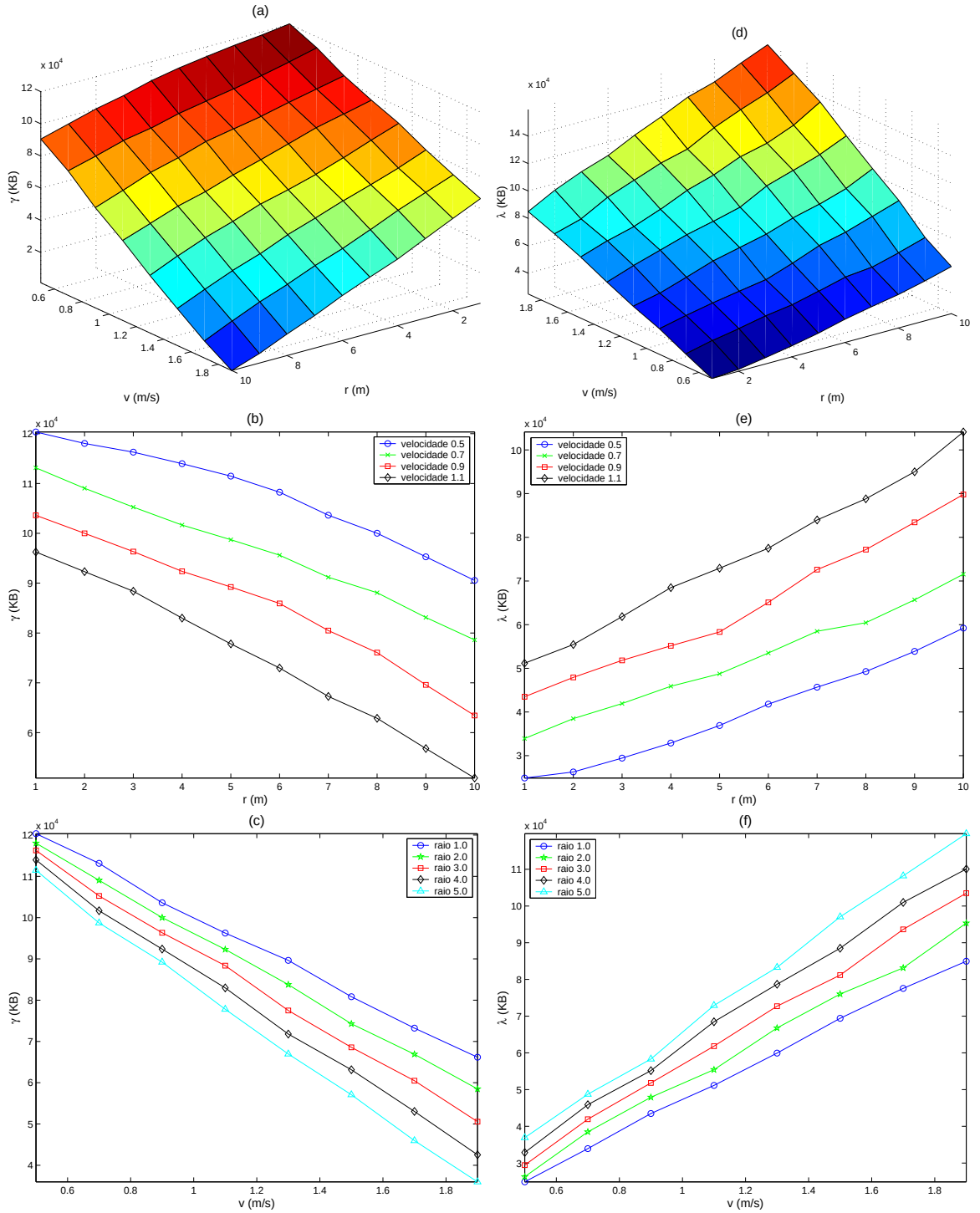

Figura 5.3: Análise de γ e λ de dados coletados em um período de tempo pré-determinado para o Método Híbrido. (a)-(c) Aumentando-se o raio r e velocidade v, γ decresce. (d)-(f) Aumentando-se o raio r e velocidade v, λ aumenta.

a perda ς_h da rede como

$$\varphi_h = \frac{\lambda}{\text{Quantidade total de dados}} = 0,32 \text{ e} \tag{5.1}$$

$$\varsigma_h = \frac{\gamma}{\text{Quantidade total de dados}} = 0,52\,, \tag{5.2}$$

onde a *Quantidade total de dados* pode ser facilmente calculada para um determinado período de simulação, conhecendo-se a equação que determina o crescimento de dados nos sensores, bem como o número de sensores no ambiente. Para um período de 600s e sabendo-se que os 60 nós sensores da rede coletam dados do ambiente a uma taxa de 40 Kb/s, a quantidade total de dados coletados é de 175680 KB. A soma de φ_h e ς_h é sempre menor ou igual a 1, pois a quantidade total de dados na rede é a soma de três grandezas: i) a quantidade λ de dados coletados, ii) a quantidade γ de dados perdidos e iii) a quantidade de dados que permanece armazenada nos nós sensores sem ser extraída pelo robô móvel.

Análise de t e τ para o Método Reativo. A Figuras 5.4(a)-(f) são equivalentes às Figuras 5.2(a)-(f). O comportamento é similar nos gráficos observados, porém, nas Figuras 5.4(a)-(b) e 5.4(d)-(e) é possível notar um crescimento do tempo t e da trajetória τ para valores de r iguais a 5 e 6. Deve-se lembrar que os resultados são referentes ao Método Reativo cujo comportamento não é determinístico e assim não pode ser *previsto* com facilidade. Mesmo com essa consideração, ainda é difícil explicar esse resultado. Paralelamente, observe o resultado de outra simulação realizada para as variáveis t e τ, na Figura 5.5. Nela é possível notar um aumento das variáveis t e τ para o raio 3, particularmente para as velocidades 0.5 e 0.6. Os resultados mostrados nas figuras 5.4 e 5.5 utilizaram os mesmos parâmetros mostrados no início desse capítulo e foram realizadas no mesmo ambiente. A explicação para essas "anomalias" está na posição do robô ao iniciar a navegação. Nas simulações referentes à Figura 5.4 o robô *sempre* começou no centro do ambiente, onde se localiza a unidade central, enquanto nas simulações referentes à Figura 5.5, a posição inicial do robô era aleatória. Perceba que o robô será influenciado pelos campos de potencial da região onde se encontra (é importante observar que eles crescem de forma determinística), e assim, é possível que as trajetórias realizadas, partindo-se da mesma posição inicial sejam semelhantes. Como o Método Reativo não se preocupa com a otimalidade das rotas, não é possível garantir que os resultados se comportem de forma similar aos resultados do Método Híbrido. Já o gráfico mostrado na Figura 5.4(c) mostra que o tempo t diminui à medida

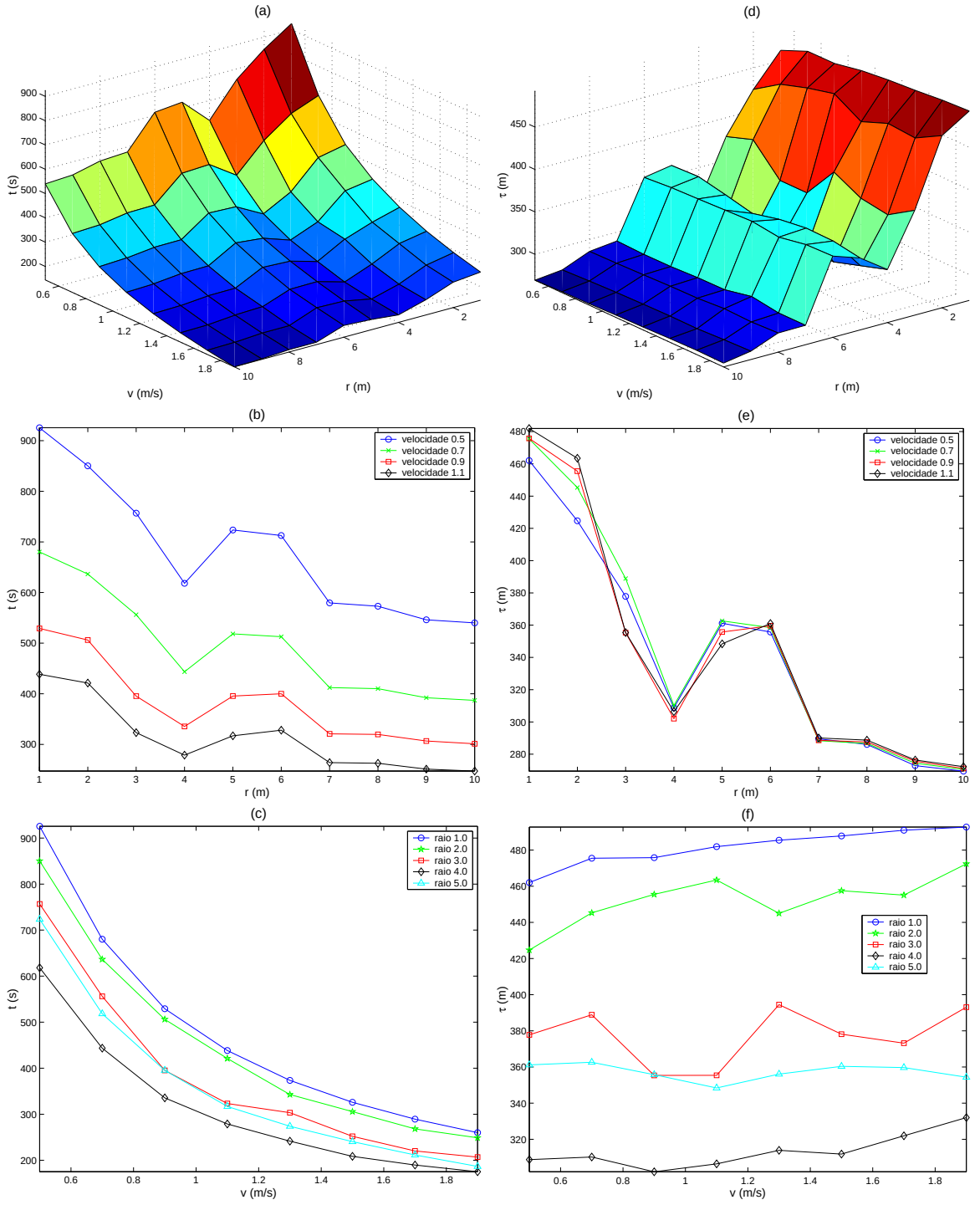

Figura 5.4: Análise de t e τ em um ciclo para o Método Reativo. (a)-(c) Aumentando-se o raio r e a velocidade v, t decresce, exceto para valores de $r = 5$ e $r = 6$. (d)-(e) Aumentado-se o raio r, τ decresce, exceto para valores de $r = 5$ e $r = 6$.

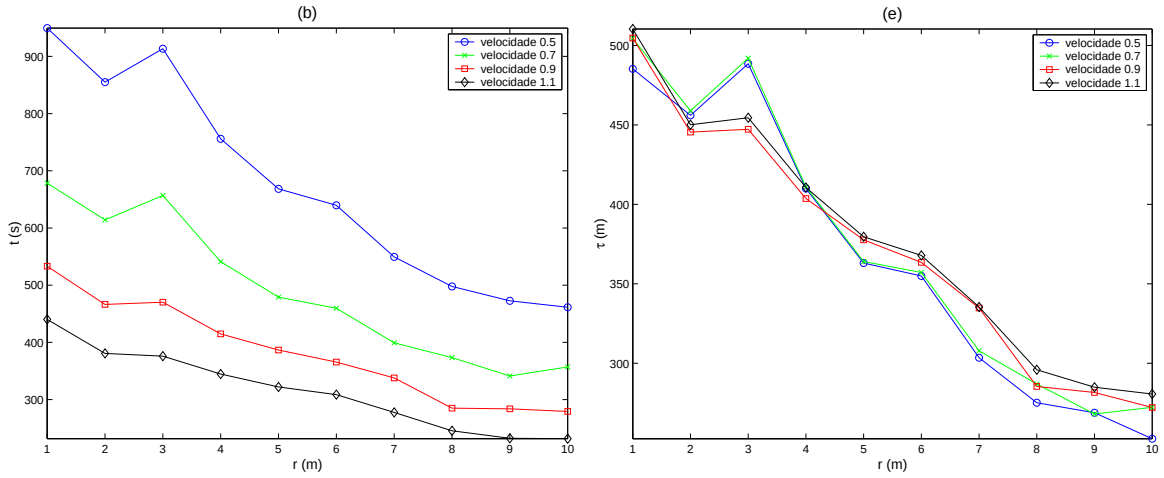

Figura 5.5: variação das variáveis t e τ em uma simulação realizada no ambiente da Figura 5.1, com posição inicial aleatória.

que velocidade v aumenta, pois o robô consegue visitar todos os nós (e assim completar o ciclo) mais rapidamente. Porém, de acordo com a Figura 5.4(f) e diferentemente do Método Híbrido, não é possível garantir que a trajetória τ se mantenha constante com a variação de v em 1 ciclo, já que o caminho a ser seguido pelo robô não é necessariamente o mesmo para todas as velocidades.

Análise de γ e λ para o Método Reativo. As Figuras 5.6(a)-(f) são equivalentes às Figuras 5.3(a)-(f). O comportamento observado, nesse caso, é bastante similar entre os Métodos Híbrido e Reativo. Para $v = 1\text{m/s}$ e $r = 3\text{m}$, a quantidade γ obtida foi 101000 KB enquanto que a quantidade λ obtida foi 49127 KB. Então pode-se considerar o ganho φ_r e a perda ς_r da rede como sendo

$$\varphi_r = \frac{\lambda}{\text{Quantidade total de dados}} = 0,28 \text{ e} \tag{5.3}$$

$$\varsigma_r = \frac{\gamma}{\text{Quantidade total de dados}} = 0,57\,, \tag{5.4}$$

Percebe-se que φ_r é inferior à φ_h o que significa que mais dados forma coletados pelo Método Híbrido. Da mesma maneira, ς_r é superior à ς_h o que significa que mais dados foram perdidos pelo Método Reativo. Logo, é possível notar uma certa superioridade do Método Híbrido para esse ambiente.

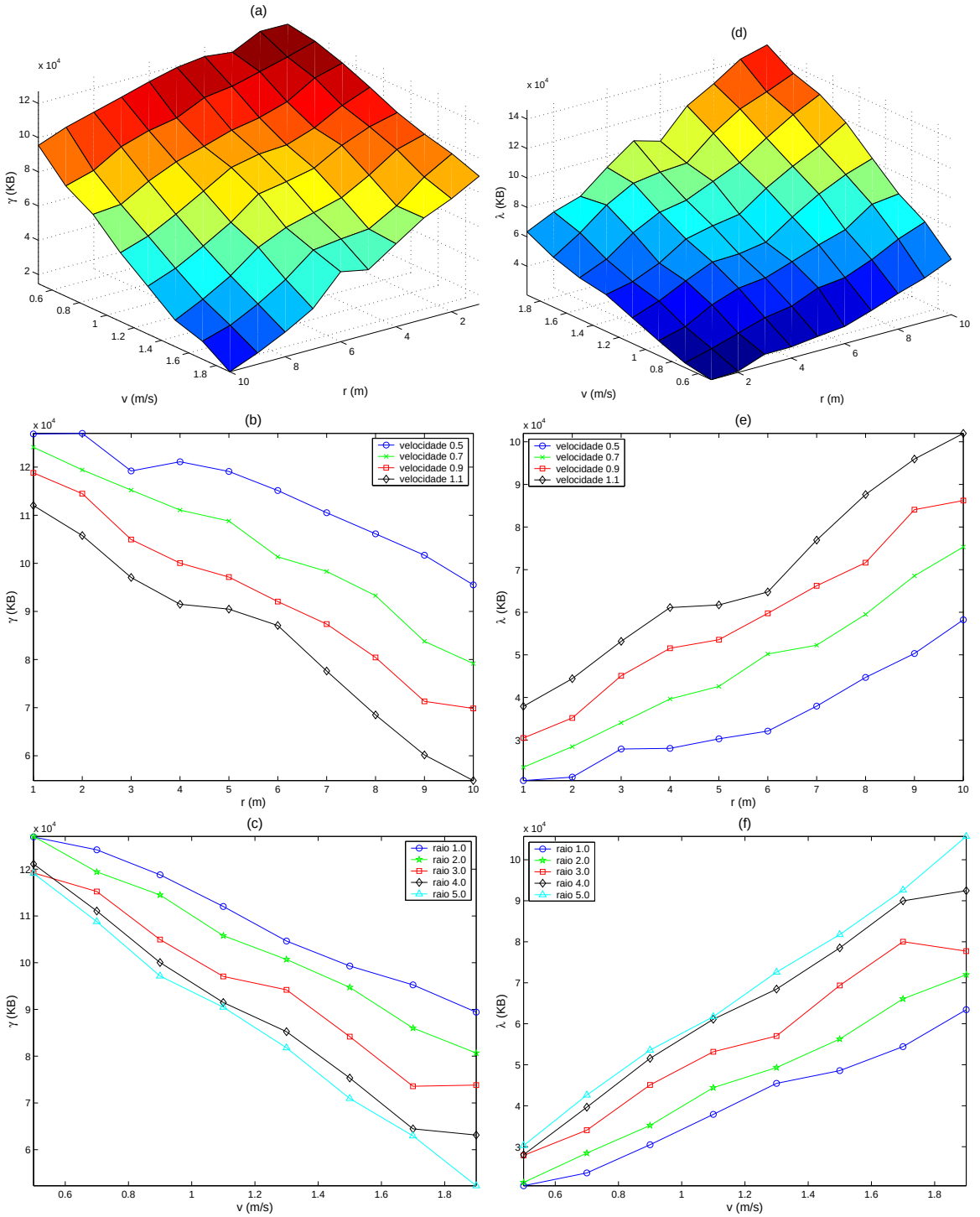

Figura 5.6: Análise de γ e λ de dados coletados em um período de tempo pré-determinado para o Método Reativo. (a)-(c) Aumentando-se o raio r e velocidade v, γ decresce. (d)-(f) Aumentando-se o raio r e velocidade v, λ aumenta.

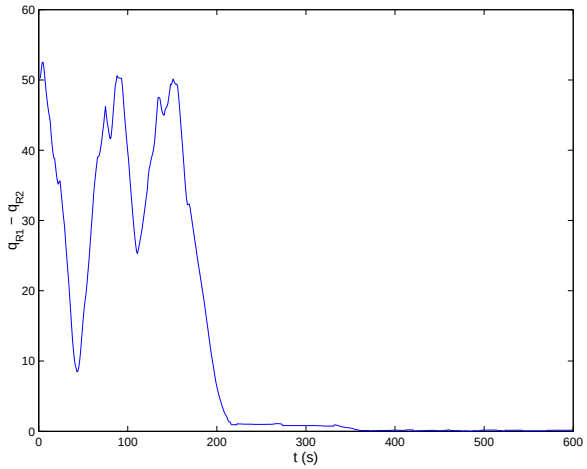

Figura 5.7: Distância entre dois robôs ao longo do tempo no Método Reativo. Perceba que aproximadamente aos 200s, os robôs se encontram na mesma posição e assim permanecem indefinidamente.

5.3.2 Resultados para mais de um robô

Essa seção mostra os resultados obtidos a partir do emprego de múltiplos robôs no ambiente. Para o Método Híbrido, a divisão dos ambiente entre os robôs está intrinsecamente associada a resolução do Problema. Porém, o Método Reativo, não preocupa-se com essa divisão.

Uma das alternativas a serem empregados no Método Reativo é espalhar os robôs pelo ambiente e deixá-los navegando livremente seguindo os campos de potencial. Nesse caso, não haveria de fato, uma divisão dos nós sensores entre os robôs, já que os robôs poderiam coletar os dados de qualquer nó sensor. Mesmo assim a *possível* melhora na eficiência utilizando-se mais de um robô já seria compensadora. Mas, o que esperar, em termos de comportamento reativo, dos robôs que irão navegar no ambiente? Se os robôs começarem a realizar suas tarefas a partir da mesma posição irão sempre permanecer juntos? E se as posições iniciais forem diferentes, manterão distância um do outro (otimizando assim a área a ser coberta)?

A Figura 5.7 mostra a distância absoluta entre as posições de dois robôs R_1 e R_2, que começam a tarefa em posições iniciais distantes entre si. Percebe-se que para $t > 200$s os robôs encontram-se em uma posição e permanecem juntos indefinidamente. A Figura 5.8(a) mostra um período da simulação da Figura 5.7, entre 10s e 60s, no qual os robôs se aproximam muito mas conseguem seguir em direções distintas, explorando novas áreas. A Figura 5.8(b) mostra um período ocorrido entre 170s e 260s no qual os dois robôs são atraídos para o mesmo nó sensor na parte inferior da figura, convergindo

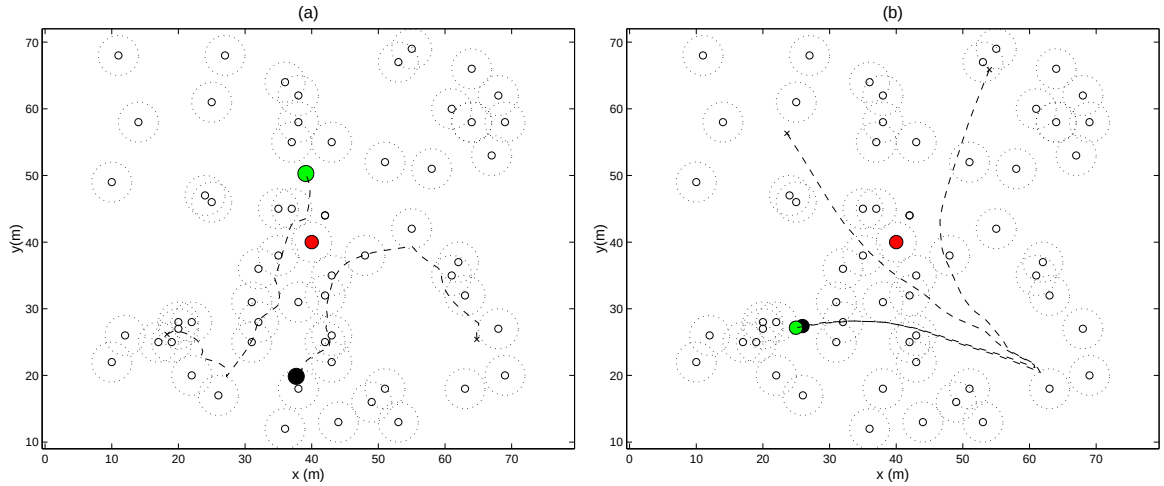

Figura 5.8: Dois robôs holômicos sob a influência do campo de potencial. (a) Período da simulação no qual os dois seguem em direções distintas, após estarem relativamente próximos. (b) Período da simulação no qual os robôs convergem em um mesmo ponto após serem atraídos pelo campo de potencial de um nó sensor. Eles permancem juntos indefiniddamente.

em um ponto e permanecendo juntos até o final da simulação. Esse problema, eventualmente, poderia ser solucionado considerando-se os robôs como obstáculos. Pretende-se tratar esse problema em trabalhos futuros.

O comportamento observado torna-se, portanto, desinteressante sob a ótica do problema que está-se abordando nessa trabalho. É preciso garantir que a divisão dos nós entre os robôs possa ocorrer, de forma a poder tirar proveito da redundância do sistema. Assim, neste trabalho os agrupamentos atribuídos a cada robô, no Método Reativo, foram os mesmos agrupamentos obtidos no Método Híbrido. Essa escolha permite também uma análise mais justa dos resultados, já que os robôs serão responsáveis pelos mesmos nós sensores, independentemente do método de extração utilizado.

Para as simulações dessa seção escolheu-se $r = 3\text{m}$, $v = 1\text{m/s}$ e a taxa de transferência de dados h entre o robô e o nó sensor é 25 vezes maior que a taxa de aquisição de pacotes g_i do sensor s_i. Além disso, as capacidades Q dos robôs variaram à medida que o número de robôs aumentava, de forma a dividir a rede de nós sensores igualmente entre eles. Por exemplo, sabendo-se que o número de nós sensores é 60, e considerando-se 1 robô, a capacidade Q do robô precisa ser necessariamente igual a 60 para que a tarefa seja realizada em 1 ciclo. Para 2 robôs a capacidade de cada um dos robôs deve ser igual a 30 e assim por diante. Resultados obtidos com as capacidades Q dos robôs constantes e m variável podem ser vistos no Anexo C.

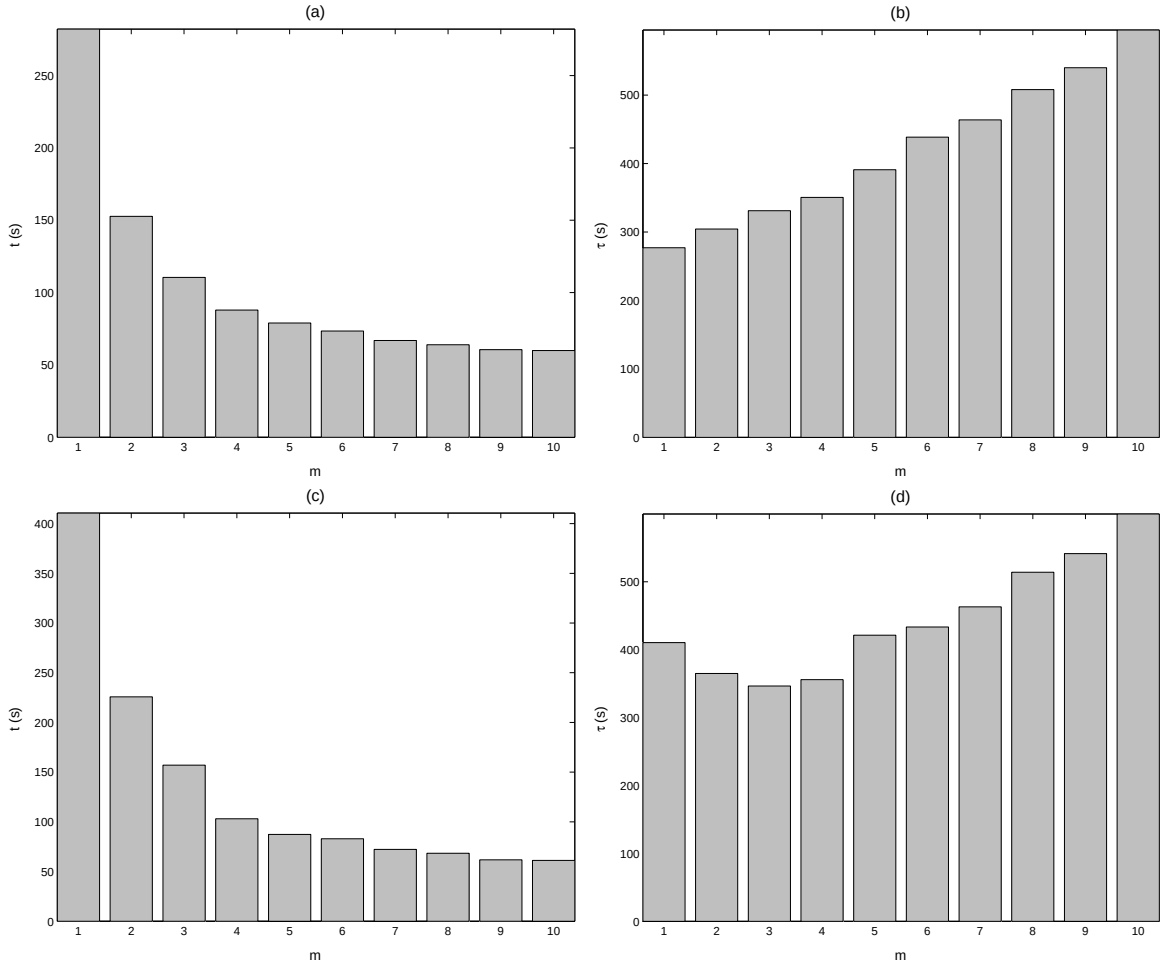

Figura 5.9: (a)-(b) Variação de t e τ em função do número de robôs para o Método Híbrido. Quanto maior m mais rápido a tarefa é realizada. Entretanto a trajetória total τ aumenta. (c)-(d) Variação de t e τ em função do número de robôs para o Método Reativo. O comportamento é similar ao encontrado no Método Híbrido. Perceba ainda que para $m > 4$, os valores de t e τ nos dois Métodos são equivalentes.

Análise de t e τ. A Figura 5.9(a)-(b) mostra os resultados obtidos para o Método Híbrido. Como é possível perceber à medida que o número de robôs aumenta o tempo para a realização das tarefas diminui rapidamente para $m \leq 3$ e diminui muito lentamente a para $m \geq 4$. Em contrapartida, a trajetória τ aumenta a medida que o número de robôs cresce, demonstrando que uma diminuição na trajetória de cada robô (local) não representa uma diminuição global da trajetória percorrida.

A Figura 5.9(c)-(d) mostra os resultados obtidos para o Método Reativo. O mesmo comportamento é observado nos gráficos de t e τ embora, em relação a esse último, o gráfico não seja sempre crescente (veja os valores de τ para $m = 1$ e $m = 2$). Como o

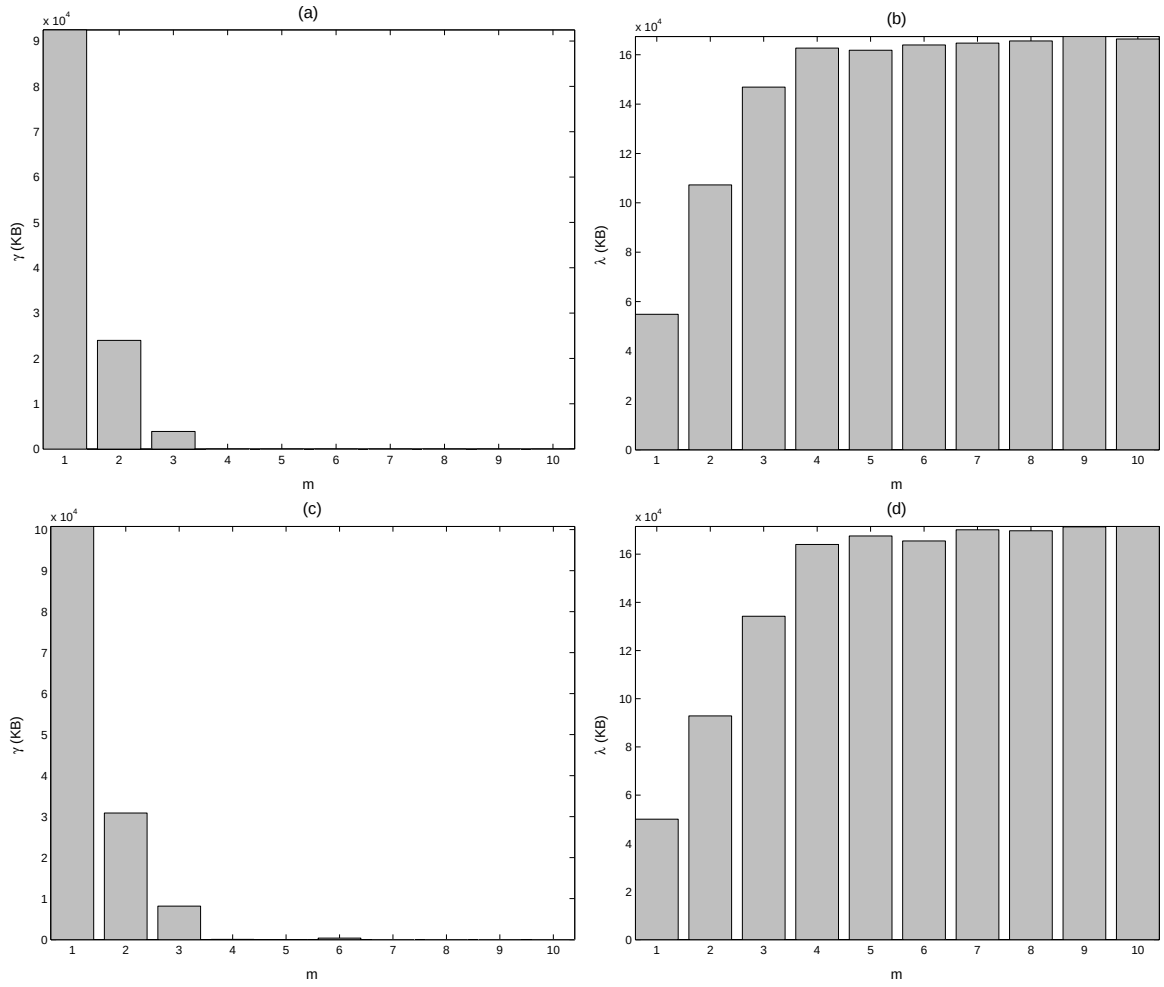

Figura 5.10: (a)-(b) Variação de γ e λ em função do número de robôs para o Método Híbrido. Para $m \geq 4$, a quantidade de dados perdidos γ é nula e a quantidade de dados coletados λ é constante. (c)-(d) O Método Reativo apresenta um comportamente similar.

robô percorre o ambiente seguindo os campos de potencial gerados pelos nós sensores e sem a preocupação de que as trajetórias sejam otimizadas, o caminho total realizado para um robô, pode ser maior que o caminho total para dois robôs. É interessante observar também que para valores pequenos de m, o tempo t e a trajetória τ são maiores no Método Reativo em comparação com Método Híbrido. Entretando, à medida que o m aumenta essa diferença diminui e torna-se equivalente nos dois métodos. Isso também pode ser facilmente entendido, já que, à medida que o número de robôs m aumenta no ambiente, os agrupamentos atribuídos a cada um deles tornam-se menores e, assim, a rota otimizada computada a partir do Método Híbrido não representa uma vantagem tão significativa em relação a rota usada pelo robô no Método Reativo.

Tabela 5.1: Ganhos φ_h e perdas ς_h para $1 \leq m \leq 10$, no Modelo Híbrido.

m	1	2	3	4	5	6	7	8	9	10
φ_h	0,31	0,61	0,84	0,93	0,92	0,93	0,94	0,94	0,95	0,95
ς_h	0,53	0,14	0.02	0,0	0,0	0,0	0,0	0,0	0,0	0,0

Tabela 5.2: Ganhos φ_r e perdas ς_r para $1 \leq m \leq 10$, no Modelo Reativo.

m	1	2	3	4	5	6	7	8	9	10
φ_h	0,28	0,53	0,76	0,93	0,95	0,94	0,97	0,97	0,98	0,98
ς_h	0,57	0,18	0,05	0,0	0,0	0,0	0,0	0,0	0,0	0,0

Análise de γ e λ. Seguindo o padrão ditado pela seção anterior, os resultados para essas duas variáveis foram obtidos em um período de tempo pré-determinado e não apenas em 1 ciclo η_k para cada robô R_k. As Figuras 5.10(a)-(b) apresentam os resultados para o Método Híbrido. É possível notar em (a) que, à medida que o número de robôs cresce, a quantidade de dados perdidos γ diminui muito rapidamente, tornando-se nula para $m \geq 4$ robôs. Complementarmente, em (b) para $m \geq 4$, a quantidade de dados coletados λ torna-se aproximadamente constante. Já que a quantidade de informação em cada nó sensor cresce de acordo com a equação linear mostrada no Capítulo 3 e nenhuma informação é perdida para $m \geq 4$, λ será sempre constante para um período de tempo pré-determinado.

As Figuras 5.10(c)-(d) apresentam os resultados para o Método Reativo. O comportamento dos gráficos obtidos é mais uma vez similar ao das Figuras 5.10(a)-(b). A diferença principal está no fato de que no Método Reativo a perda de dados da rede é maior. A Tabela 5.1 mostra o ganho φ_h e a perda ς_h obtidos para o Método Híbrido. Similarmente, a Tabela 5.2 mostra o ganho φ_r e a perda ς_r obtidos para o Método Reativo.

O Método Híbrido mostrou-se superior ao Método Reativo para $1 \leq m \leq 3$. Para $m \geq 4$ os dois métodos apresentam ganhos e perdas muito próximos, com uma pequena vantagem para o Método Reativo.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Nessa Dissertação foram propostos dois métodos para a extração de dados em uma rede de nós sensores utilizando robôs móveis: o Método Híbrido e o Método Reativo. Ambos podem ser vistos como heurísticas que tentam minimizar variáveis de interesse do sistema como o tempo t de realização das tarefas, a trajetória τ e a quantidade γ de dados perdidos na rede. Ao mesmo tempo é importante que a quantidade λ de dados coletados na rede seja maximizada, pois o objetivo primário dos métodos é realizar a extração de dados da rede.

O Método Híbrido proposto no Capítulo 4 tem sua metodologia baseada em soluções para o Problema do Roteamento de Veículos (VRP). Conforme foi dito naquele capítulo, tal abordagem visava minimizar o custo total para que todos os nós da rede fossem visitados, dividindo-os entre os robôs e gerando rotas otimizadas por alguma heurística. Para este trabalho, o custo baseava-se exclusivamente na variável τ. Porém, outras variáveis como t, e γ poderiam ser minimizadas por meio desse processo. O principal problema com essa metodologia, advém do fato de ser centralizada. Defeitos no nó central, responsável pela geração do plano, impossibilitariam definitivamente o cumprimento das tarefas pelos robôs.

O Método Reativo proposto no Capítulo 3 representa a quantidade de dados coletados pelo nó sensor por meio de funções de potencial. Assim, quanto maior for essa quantidade, maior será a força de atração sentida pelo robô (devido ao campo de potencial gerado). O objetivo principal dessa metodologia é minimizar a quantidade de dados γ perdidos na rede. Entretanto, não é possível garantir que nenhuma informação será perdida pela rede, conforme foi constatado no Capítulo 5, já que isso depende de

muitos fatores como raio de comunicação dos sensores, velocidade dos robôs, taxa de aquisição de dados do sensor, distância entre os nós sensores, número de robôs empregados, etc. Existe também a possibilidade de que alguns nós sensores da rede nunca sejam visitados. Isso pode ocorrer quando o raio ρ da função de base radial desses sensores tiver uma limitação de tamanho e consequentemente área de influência limitada. Esse nós, então, "morreriam por inanição" *(starvation)*, pois o robô nunca se sentiria atraído por esse nós. Entretanto, diferentemente do Método Híbrido, o Método Reativo tem a vantagem de ser implementado de forma descentralizada e, portanto, problemas na unidade central não comprometem o funcionamento do sistema.

Conforme visto no Capítulo 3, o Método Reativo tem complexidade $O(Kn)$, onde $n \in \mathbb{N}^+$ e representa o número de sensores da rede e K é um valor constante positivo que representa o número de iterações do algoritmo. No Capítulo 4 foi mostrado que a Heurística de Varredura tem complexidade $O(|\mathcal{A}| \log n)$ e a Heurística de Clarke & Wright tem complexidade $O(n^2 \log n)$. Como $Kn = o(|\mathcal{A}| \log n)$ (lembre-se que $|\mathcal{A}| = 2n$), então $Kn \leq 2n \log n$ para $n \geq n_0$, onde $n_0 \in \mathbb{N}^+$. Similarmente, como $Kn = o(n^2 \log n)$, então $Kn \leq n^2 \log n$ para $n \geq n_0$. Assim, o Método Híbrido, *eventualmente*, pode ser mais vantajoso computacionalmente que o Método Reativo para $n < n_0$, uma vez que é executado apenas 1 vez. Entretanto, para $n \geq n_0$, o Método Reativo será sempre menos custoso, para um valor constante de K iterações. Outra análise interessante pode ser obtida variando-se o valor de K e mantendo-se n constante. Nesse caso, a partir de determinado valor de K, o Método Reativo terá um custo computacional maior que o Método Híbrido. Tal comportamento se deve ao fato de que no Método Híbrido esse custo não variará, pois depende de n (mantido com valor constante), ao contrário do que ocorre no Método Reativo cujo custo aumentará sempre que K aumentar.

De acordo com os resultados apresentados no Capítulo 5, o Método Híbrido mostrou-se melhor que o Método Reativo, tanto para quantidade $m = 1$ de robôs como para uma quantidade $m > 1$ de robôs. Para $m = 1$, $\varphi_h \geq \varphi_r$ indica que o Método Híbrido coletou mais dados que o o Método Reativo. Pode-se perceber também que $\varsigma_h \leq \varsigma_r$ indicando que o Método Híbrido perdeu menos dados que o Método Reativo. Para $m > 1$, esse comportamento manteve-se entre os métodos, tanto para os ganhos como para as perdas. Entretanto para $m > 4$, em ambos os métodos propostos, percebe-se que a quantidade de dados perdidos γ é nula e a quantidade de dados coletados λ permanece aproximadamente constante, o que significa, em outras palavras, ganhos e perdas aproximadamente iguais. Nessa situação, pode-se observar ainda que o tempo t

e trajetória τ são praticamente iguais, tornando indiferente a escolha por um dos dois métodos. É importante notar, contudo, que o comportamento observado para $m > 4$ está relacionado com o ambiente de testes. Provavelmente, dependendo do tamanho do ambiente, das distâncias entre os nós sensores, do raio de comunicação dos nós sensores e da velocidade dos robôs, o resultado seria outro. Por exemplo, em ambiente maiores, espera-se que o número de robôs m exigido, para que não haja perdas na rede, seja maior.

O fato de o Método Híbrido ter apresentado resultados melhores nas simulações realizadas, pode ser devido às características do ambiente. É importante lembrar que os nós sensores coletam informação a uma taxa constante no tempo. Dessa forma a alternativa de usar um robô móvel que monitore a rede seguindo caminhos otimizados parece bem mais promissora do que fazê-lo seguir campos de potencial que podem ser igualmente fortes em muitas regiões do ambiente, pois a intensidade desses campos cresce a uma taxa constante para todos os nós sensores. Percorrer o ambiente o mais rápido possível, além minimizar custos relativos a trajetória e tempo, significa poder chegar a fonte dos dados com maior rapidez, mesmo que essa estratégia ignore qual sensor necessita ser visitado com maior urgência. Em contrapartida, isso não invalida o Método Reativo, apenas o torna inadequado para ambientes como os utilizados nessa Dissertação. Em ambientes nos quais a rede de nós respondesse à ocorrência de eventos, como pode ser visto em [Intanagonwiwat et al., 2000], [Wu et al., 2004], o método reativo apresenta-se como uma alternativa bem mais promissora, pois o robô conseguiria reagir prontamente a um evento específico que ocorreu em uma determinada região da rede. Os campos de potencial de outras regiões da rede não "atrapalhariam"o robô de forma significativa, já que materiam-se menos influentes até que um evento fosse detectado.

Também foi possível observar que a quantidade de dados perdidos γ manteve-se relativamente alta nos dois métodos de extração propostos, principalmente para $m = 1$. Essas perdas podem ser toleradas se a informaçao contida nos dados coletados não variar a uma taxa muito rápida. Assim, por exemplo, para o caso de dados sobre temperatura de um ambiente (por exemplo, floresta), a perda de dados é algo aceitável visto que a variação da temperatura pode ser mínima entre duas visitas sucessivas de um robô ao nó sensor. Obviamente, o período de tempo decorrido entre duas visitas sucessivas depende do número de robôs, tamanho do ambiente, velocidade do robô, etc. Caso as perdas de dados sejam intoleráveis, pode-se aplicar algum algoritmo de roteamento de dados nas redes de sensores e transmitir uma parcela dos dados para

a unidade central. Mesmo nesse caso, os métodos aqui propostos são aplicáveis, pois diminuiriam o tráfego de dados na rede e o consumo de energia dos sensores.

6.2 Trabalhos Futuros

Melhorias no Método Híbrido. O plano gerado nas simulações realizadas leva em conta as capacidades máximas C de cada nó sensor, bem como as capacidades Q dos robôs e as distâncias entre os nós. Dessa forma, o fato de utilizar-se um planejador centralizado, não representa um problema significativo, pois o plano permanece o mesmo ao longo do tempo para todos os robôs. Entretanto, conforme foi comentado no capítulo anterior, uma abordagem mais interessante (por levar em conta a dinâmica das redes de nós sensores) seria empregar as demandas $d_i(t)$ de cada nó sensor no instante do planejamento. Ou seja, sempre que o robô retornasse à unidade central, um novo plano seria gerado com base na quantidade de dados presentes nos sensores. Nessa situação, o problema de ter um planejador centralizado voltaria a ser pertinente, de maneira que duas alternativas poderiam ser empregadas visando solucioná-lo:

1. Replicar o número de unidades centrais e agrupar os nós sensores para cada uma delas de acordo com algum critério (por exemplo, proximidade): essa abordagem permiriria que existissem n planos, um para cada agrupamento. Sempre que a unidade central responsável por um dos agrupamentos falhasse, os nós sensores que lhe pertenciam seriam atribuídos a outra unidade central.

2. Permitir que cada robô gere um plano: cada um dos robôs assumiria a posição de "unidade central" e dessa forma geraria um plano que levasse em conta todos os robôs. Os planos de cada um dos robôs poderiam diferir de acordo com a percepção que cada um deles tem do ambiente e de suas próprias capacidades. Logo, para cada plano estaria associado um custo (que além de levar em conta as rotas geradas, poderia considerar energia do robô, restrições de tempo para chegar a um determinado nó sensor, etc). Por meio de algum processo de negociação, os robôs decidiriam qual desses planos seria de fato executado. Esse processo de negociação poderia utilizar algum Protocolo de Alocação de Tarefas (que teria por base o custo de cada plano) como, por exemplo, o *Contract Net* [Smith, 1980], ou uma variante deste apresentada em [Botelho, 2000].

Melhorias no Método Reativo. Para fins de análises futuras, é interessante considerar a presença de obstáculos no ambiente. Complementarmente, os robôs poderiam ser modelados também como obstáculos móveis, criando campos de potencial repulsivos. Isso faria com que outros robôs permanecessem afastados, explorando áreas diferentes e evitando o problema relatado na seção 5.3.2.

Melhorias no Ambiente. O ambiente no qual os testes foram feitos possuía uma série de restrições. Todos os nós sensores coletam informação do ambiente linearmente no tempo e a ocorrência de eventos na rede não é tratada. A modelagem de eventos tornaria o sistem útil se os métodos fossem expandidos para solucionar problemas como rastreamento de objetos móveis, manutenção da conectividade da rede ou recalibração de nós. Dessa forma, o robô poderia empregar os métodos aqui propostos de forma a chegar no sensor, ou na região da rede na qual o evento se originou.

Os nós sensores nas simulações foram implementados como dispositivos fisicamente perfeitos, sempre em estado ativo e com energia infinita, já que nenhum modelo de dissipação foi proposto. Esse é um problema que pretende-se corrigir em trabalhos futuros, uma vez que essa situação é irreal e desconsidera a energia da rede, uma variável importante, em sua modelagem. Porém, para os métodos propostos neste trabalho essa era uma felixibilidade permitida, para fins do que se pretendia avaliar.

Utilização em robôs reais. Finalmente, planeja-se implementar os métodos apresentados (e suas modificações) em robôs reais. O Laboratório de Visão Computacional e Robótica (VeRLab [VeRLab, 2004]) do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais (UFMG) possui 3 robôs Pioneer P3AT, que poderão ser usados para esse propósito.

Referências Bibliográficas

[Aarts and Korst, 1989] Aarts, E. and Korst, J. (1989). *Simulated Anneling and Boltzmann Machines, A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Jonh Wiley & Sons.

[Aarts and Lenstra, 1997] Aarts, E. and Lenstra, J. K. (1997). *Local Search in Combinatorial Optimization*. Jonh Wiley & Sons.

[ActivMedia Robotics, 2004] ActivMedia Robotics (2004). *Pioneer* 3^{TM} *&* *Pionner* 2^{TM} *H8-Series Operations Manual*.

[Arkin, 1989] Arkin, R. C. (1989). Motor schema-based robot navigation. *International Journal of Robotics Research*, 8:92–112.

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press.

[Aronson, 1996] Aronson, L. D. (1996). Algorithms for Vehicle Routing - A Survey. Technical Report 21, Technishe Universiteit Delft.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams - a survey of a fundamental data structure. *ACM Computing Surveys*, 23:345–405.

[Batalin and Sukhatme, 2004] Batalin, M. and Sukhatme, G. S. (2004). Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, 26(2):181–196.

[Bodin, 1983] Bodin, L. (1983). Rounting and scheduling of vehicle and crews, the state of the art. *Comput. and Ops. Res.*, 10(2):63–211.

[Botelho, 2000] Botelho, S. (2000). *Une Architecture Décisionelle Pour La Coopération Multi-Robots*. PhD thesis, Laboratoire d'Analyse et d'Archicteture des Sytèmes du CNRS.

[Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 1(2):14–23.

[Bräysy and Gendreau, 2001] Bräysy, O. and Gendreau, M. (2001). Genetic Algorithms for the Vehicle Routing Problem with Time Windows. Technical Report STF42 A01021, SINTEF Applied Mathematics, Department of Optimization, Oslo, Norway.

[Bullnheimer et al., 1997] Bullnheimer, B., Hartl, R. F., and Strauss, C. (1997). Applying the Ant System to the Vehicle Routing Problem. In *2nd International Conference on Metaheuristics*, pages 699–719, Sophia-Antipolis, France.

[Clarke and Wright, 1964] Clarke, C. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Ops. Res.*, 12:568–581.

[Corke et al., 2004] Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., and Sukhatme, G. (2004). Autonomous deployment and repair of a sensor network using an ummanned aerial vehicle. In *Proceedings of the International Conference on Robotics and Automation (ICRA'04)*, volume 4, pages 3602–3608, New Orleans, USA.

[Corke et al., 2003] Corke, P., Peterson, R., and Rus, D. (2003). Networked Robots: Flying Robot Navigation Using a Sensor Net. In *International Symposium of Robotics Research (ISRR'03)*.

[Cormem, 2001] Cormem, T. H. (2001). *Introduction to Algorithms*. McGraw-Hill.

[Crossbow Technology, Inc., 2004a] Crossbow Technology, Inc. (2004a). `http://www.xbow.com`.

[Crossbow Technology, Inc., 2004b] Crossbow Technology, Inc. (2004b). *MPR/MIB User's Manual*.

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Mgmt Sci*, 6(1):80–91.

[Díaz, 2003] Díaz, B. D. (2003). The VRP Web. `http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html`.

[Dhariwal et al., 2004] Dhariwal, A., Sukhatme, G. S., and Requicha, A. A. G. (2004). Bacterium-inspired robots for environmental monitoring. In *IEEE International Conference on Robotics and Automation (ICRA-04)*, volume 2, pages 1436–1443.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41.

[Dudek and Jenkin, 2000] Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*, chapter 3. Cambridge University Press.

[Elfes, 1987] Elfes, A. (1987). Sonar-based real world mapping anda navigation. *IEEE Transaction on Robotics and Automation (ICRA 1987)*, 3(3):249–265.

[Fikes and Nilsson, 1971] Fikes and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:198–208.

[Fisher and Jaikumar, 1981] Fisher, M. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124.

[Gendreau, 2002] Gendreau, M. (2002). An Introduction to Tabu Search. Disponível em: `http://www.ifi.uio.no/infheur/Bakgrunn/Intro\_to\_TS\_Gendreau.htm`.

[Gerkey et al., 2002] Gerkey, B. P., Vaughan, R. T., and Howard, A. (2002). *Player User Manual*.

[Gerkey et al., 2003] Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, Coimbra, Portugal.

[Gerkey et al., 2001] Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. S., and Mataric, M. J. (2001). Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, volume 3, pages 1226–1231, Wailea, Hawaii.

[Gillet and Miller, 1974] Gillet, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatching problem. *Ops. Res.*, 22:340–349.

[Heinzelman et al., 2000] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, pages 1–10.

[Hjorring, 1995] Hjorring, C. (1995). *The Vehicle Rounting Problem and Search Meta-Heuristic*. PhD thesis, University of Auckland.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial System*. The University of Michigan Press.

[Intanagonwiwat et al., 2000] Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, pages 56–67, Boston, Massachussetts.

[Kantor et al., 2003] Kantor, G., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereira, G., and Spletzer, J. (2003). Distributed Search and Rescue with Robot and Sensor Team. In *4th International Conference on Field and Service Robotics*.

[Khatib, 1985] Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA 1985)*, pages 500–505, St. Louis, USA.

[Koren and Borestein, 1991] Koren, Y. and Borestein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA 1991)*, volume 2, pages 1398–1404, Sacramento, USA.

[Krogh and Thorpe, 1986] Krogh, B. and Thorpe, C. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. In *IEEE International Conference on Robotics and Automation (ICRA 1986)*, volume 3, pages 1664–1669, San Francisco, USA.

[LaMarca et al., 2002] LaMarca, A., Koizumi, D., Lease, M., Sigurdsson, S., Borriello, G., Brunette, W., Sikorski, K., and Fox, D. (2002). Making Sensor Networks Practical with Robots. Technical Report IRS-TR-02-004, Intel Research.

[Laporte and Semet, 1999] Laporte, G. and Semet, F. (1999). Classical Heuristics for the Vehicle Routing Problem.

[Latombe, 1991] Latombe, J. C. (1991). *Robot Motion Planning*. Kluwer Academic Publisher.

[LaValle, 2004] LaValle, S. M. (2004). Planning algorithms. Disponível em: `http://msl.cs.uiuc.edu/planning/`.

[Li et al., 2003] Li, Q., de Rosa, M., and Rus, D. (2003). Distributed Algorithms for Guiding Navigation across a Sensor Net. In *11th International Symposium of Robotics Research*, Siena, Italy.

[Meguerdichian et al., 2001] Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M. B. (2001). Coverage problems in wireless ad-hoc sensor networks. In *In Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-01)*, pages 1380–1387.

[Mehta et al., 2003] Mehta, D. P., Lopez, M. A., and L., L. (2003). Optimal coverage paths in ad-hoc sensor networks. In *IEEE International Conference on Communications (ICC-03)*, pages 507–511.

[Mini et al., 2004] Mini, R. A. F., Loureiro, A. A. F., and Nath, B. (2004). The distinctive design characteristc of wireless sensor network: the energy map. *Computer Communications*, 27:935–945.

[Moravec, 1988] Moravec, H. P. (1988). Sensor fusion in certainty grids fo mobile robots. *AI magazine*, 9(2):61–74.

[Mulmuley, 1994] Mulmuley, K. (1994). *Computational Geometry: An introduction Through Randomized Algorithms*. Prentice-Hall.

[NASA, 2004] NASA (2004). `http://robotics.jpl.nasa.gov`.

[Pereira et al., 2004a] Pereira, G. A. S., Soares, M. B., and Campos, M. F. M. (2004a). A potencial field approach for collecting data from sensor networks using mobile robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3469–3474, Sendai, Japan.

[Pereira et al., 2004b] Pereira, G. A. S., Torres, F. B., and Campos, M. F. M. (2004b). Desenvolvimento de robôs holonômicos de baixo custo para o estudo de robótica móvel. In *XV Congresso Brasileiro de Automática (CBA'04)*, Gramado, RS, Brasil.

[Peterson and Rus, 2004] Peterson, R. and Rus, D. (2004). Interacting with sensor networks. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 180–186.

[Player/Stage, 2004] Player/Stage (2004). `http://playerstage.sourceforge.net`.

[Powell, 1985] Powell, M. J. D. (1985). Radial basis function for multivariable interpolation: A review. In *IMA Conference on Algorithms for the Approximation of Function and Data*.

[Prim, 1957] Prim, R. (1957). Shortest connection networks and some generalization. *Bell System Technical Journal*, 36:1389–1401.

[Russell and Norvig, 2002] Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*, chapter 8. Prentice Hall.

[Sensornet, 2004] Sensornet (2004). `http://www.sensornet.dcc.ufmg.br`.

[Sibley et al., 2002] Sibley, G. T., Rahimi, M. H., and Sukhatme, G. S. (2002). Robomote: A tiny mobile robot platform for large-scale sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA'02)*, volume 2, pages 1143–1148, Washington, USA.

[Silva et al., 2004] Silva, F., Braga, T. R., Ruiz, L. B., and Nogueira, J. M. S. (2004). Tecnologia de nós sensores sem fio. *Revista Controle e Instrumentação*, (92):76–86.

[Smith, 1980] Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 10(No. 12):1104–1113.

[Snyder, 2004] Snyder, L. (2004). `http://www.lehigh.edu/~lvs2/download/vrpsolver.html`.

[Soares et al., 2004] Soares, M. B., Campos, M. F. M., Pereira, G. A. S., and Mateus, R. G. (2004). Planejamento de rotas de robôs móveis em redes de sensores sem fio. In *XV Congresso Brasileiro de Automática (CBA'04)*, Gramado, RS, Brasil.

[Stanford Research Institute, 2004] Stanford Research Institute (2004). `http://www.sri.com`.

[Tanenbaum, 1995] Tanenbaum, A. S. (1995). *Modern Operating System*. Prentice Hall.

[Tang et al., 2004] Tang, Y., Birch, B., and Parker, L. E. (2004). Planning mobile sensor net deployment for navigationally-challenged sensor nodes. In *IEEE International Conference on Robotics and Automation (ICRA-04)*, volume 1, pages 172–179.

[Thangiah, 1995] Thangiah, S. (1995). *Application Handbook of Genetic Algorithms: New Frontiers*, volume II, pages 253–277. CRC Press, Boca Raton.

[USC Robotics Research Lab, 2004] USC Robotics Research Lab (2004). `http://www-robotics.usc.edu/`.

[Vaughan et al., 2002] Vaughan, R. T., Howard, A., and Gerkey, B. P. (2002). *Stage User Manual*.

[Veltri et al., 2003] Veltri, G., Huang, Q., Qu, G., and Potkonjak, M. (2003). Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *First International Conference on Embedded Networked Sensor Systems (Sensys-03)*, pages 40–50.

[VeRLab, 2004] VeRLab (2004). `http://www.dcc.ufmg.br/verlab`.

[Vieira et al., 2003] Vieira, M. A. M., da Silva Junior, D. C., Jr, C. N. C., and da Mata, J. M. (2003). Survey on wireless sensor network devices. In *IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, pages 537–544.

[Winston, 1970] Winston, P. H. (1970). *Learning structural descriptions from examples*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and pratice. *The Knowledge Enginnering Review*, 10(2):115–152.

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

[Zhao et al., 2002] Zhao, Y. J., Govindan, R., and Estrin, D. (2002). Residual energy scans for monitoring wireless sensor networks. In *IEEE Wireless Communications and Networking Conference*, volume 1, pages 356–362, Orlando, USA.

Apêndice A

Player/Stage

A.1 Player

O *software Player* [Gerkey et al., 2002], [Gerkey et al., 2001]. [Gerkey et al., 2003] foi desenvolvido no Laboratório de Robótica da Universidade da Califórnia sobre a supervisão de Brian Gerkey e Kasper Støy. Criado para ser um *servidor de dispositivos* para robôs, permite acesso a sonares, *bumpers*, *lasers*, câmeras, etc. Esse acesso é realizado por meio de *programas clientes* que podem se comunicar com o servidor enviando e recebendo mensagens.

O Esquema de Funcionamento da plataforma cliente/servidor pode ser descrito de acordo com a Figura A.1. O servidor pode executar no robô ou em um computador com dispositivos anexados a ele. O programa cliente, que pode se comunicar com o servidor via TCP, envia comandos e requisições de leitura dos dados provenientes dos dispositivos. Por exemplo, uma requisição pode ser feita ao servidor para obter os dados do *array* de sonares do robô. Por outro lado, os comandos enviados podem configurar os dispositivos. Por exemplo, o dispositivo `position` permite efetuar o controle do robô, configurando as velocidades translacionais e angular.

Os programas clientes podem ser escritos em linguagens como C, C++, java, LISP. Uma consulta a essas bibliotecas pode ser realizada em [Player/Stage, 2004]. Para essa dissertação foi utilizada a biblioteca `libplayerc` (liguagem C) para implementar os clientes. Um simples comando de leitura como:

`playerc_client_read(client)`,

utiliza o comando de leitura `playerc_client_read` para ler os dados de todos os dispositivos anexados ao robô, conectado por meio do cliente `client`. Já o comando

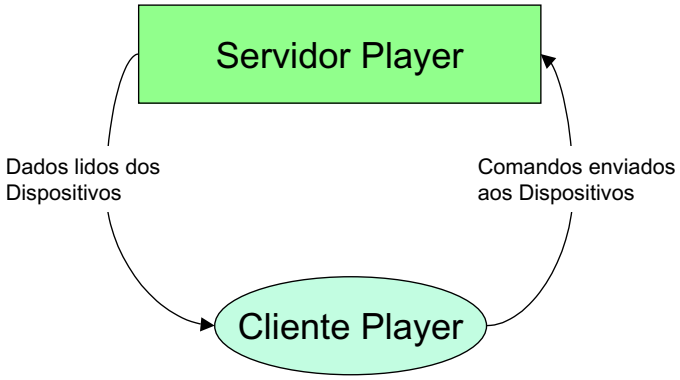

Figura A.1: Esquema de funcionamento da plataforma cliente/servidor do *software* Player. O *cliente player* envia requisições de leitura de dados e comandos ao *servidor player* que gerencia dispositivos como sonares, laser, câmeras, etc.

```
playerc_position_set_speed(position, 1, 0, 0),
```

configura o robô com a velocidade $v_x = 1$, por meio do dispositivo `position`.

A.2 Stage

Em muitas ocasiões deseja-se testar o funcionamento de algoritmos específicos antes de migrá-los para robôs reais, de forma a ter controle de todos os processos e evitar situações inesperadas ou que possam causar danos aos equipamentos. Ou então, é possível que os testes devam ser realizados com um número considerável de robôs, o que tornaria inviável experimentos reais. De forma a poder solucionar esses problemas, o ambiente de simulação *Stage*, [Vaughan et al., 2002] foi desenvolvido como parte do Projeto Player/Stage.

O ambiente no qual o robô irá atuar é descrito em um arquivo de mundo que posteriormente é lido pelo *software* Stage. Nesse arquivo, que tem extensão `.word` são especificados todos os componentes que farão parte do mundo, como objetos, obstáculos, robôs, etc. A descrição de um robô não-holonômico pode feita da seguinte forma:

```
position(
    port 6665
    pose [ 2 2 0]
    sonar ()
    laser ())
```

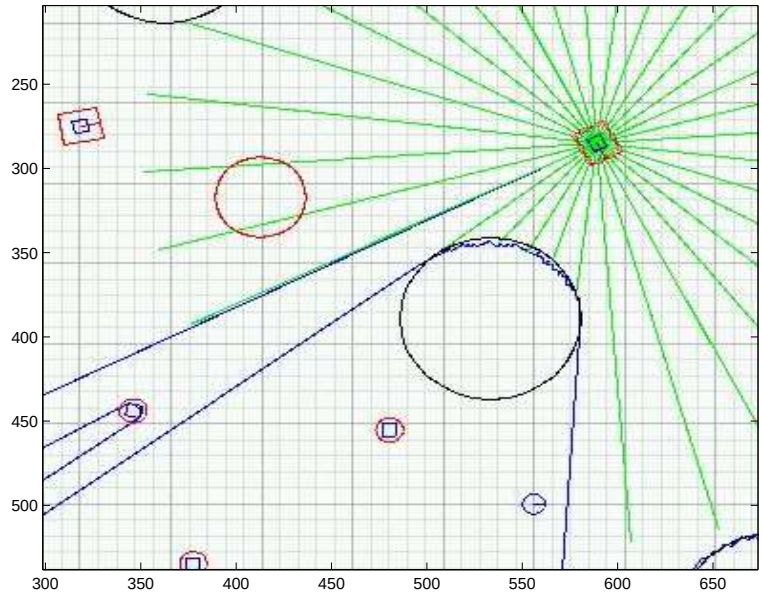

Figura A.2: Ambiente de Simulação *Stage*. No Figura é possível ver 2 robôs não-holonômicos (forma retangular), 3 robôs holonômicos (forma circular) e alguns obstáculos circulares espalhados pelo ambiente. Os raios que saem de um dos robôs representam as leituras obtidas por sonares e *laser*.

A posição inicial do robô é $(2, 2)$ com sonar e laser anexados. O programa cliente (desenvolvido com a biblioteca *libplayerc*) deverá conectar-se a porta 6665 de forma a poder ler dados e enviar comandos para o robô simulado. É interessante comentar que, o código desenvolvido para executar no ambiente simulado pode ser usado com poucas modificações em robôs reais já que a interface TCP para o *Player* é idêntica a interface TCP para o *Stage*. De fato, quando o Stage é executado, automaticamente um servidor *Player* é carregado para a memória. Dessa forma, o programa cliente irá se comunicar com o servidor Player que, por sua vez, irá ler dados e enviar comandos a dispositivos simulados pelo Stage (ao invés de dispositivos reais). Como exemplo, a Figura A.2 mostra um ambiente com um 3 robôs holonômicos, 2 robôs não-holonômicos e alguns obstáculos.

Apêndice B

VRP Solver

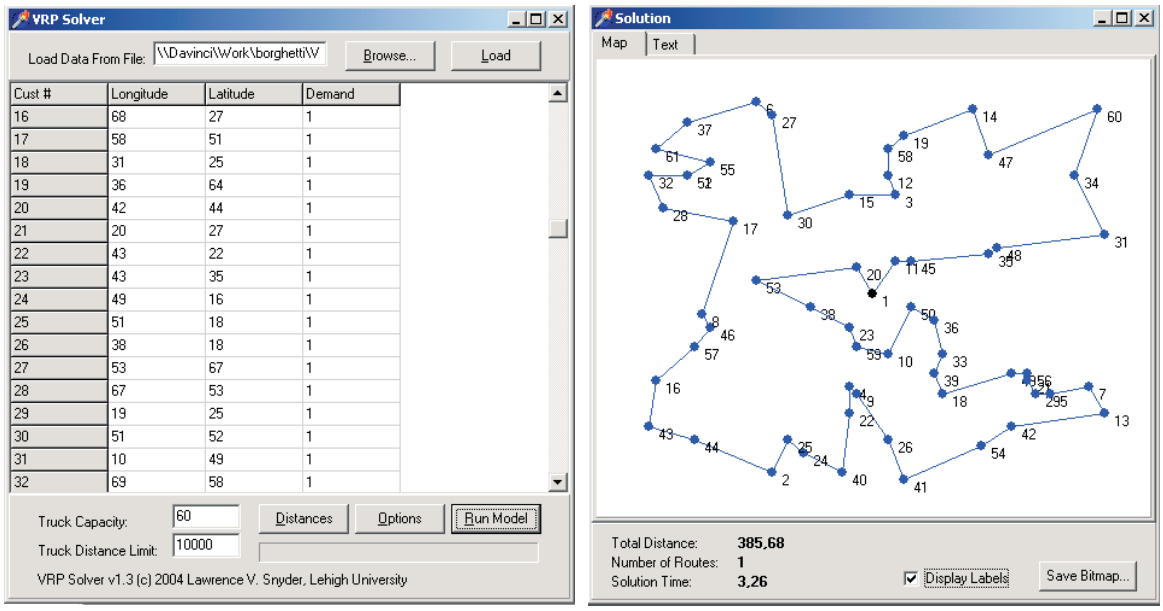

Figura B.1: VRPSolver. Na imagem à esquerda é mostrada a caixa de diálogo com o arquivo de posições dos nós sensores carregado. Na imagem à direita pode-se ver o caminho gerado aplicando a heurística de Clarke & Wright.

Desenvolvida na *Universidade de Lehigh*, por Larry Snyder [Snyder, 2004], essa ferramenta permite a geração de rotas em um ambiente 2D, a partir da posição de um conjunto de nós lidos de um arquivo de entrada. Conforme pode ser visto na imagem à esquerda da Figura B.1, a caixa de diálogo mostrada dispõe o conjunto de pontos lidos em *longitude*, *latitude* e *demanda*. Tais grandezas correspondem as coordenadas x, y e a capacidade C_i (ou $d_i(t)$) do nó sensor s_i. Na parte inferior da figura, é possível especificar a capacidade dos veículos (robôs) e a distância mínima requerida entre eles.

O botão com *Run Model* executa o algoritmo de Clarke & Wright, e a otimiza com métodos como *2-opt* e *3-opt*. A imagem à direita da Figura B.1 mostra a rota gerada a partir dos dados fornecidos pelo arquivo de posições dos nós. Na parte inferior aparecem algumas informações como o tamanho da rota e o tempo para gerá-la. É possível, ainda, obter a rota em modo de texto, selecionando a aba *text* que pode ser vista na parte superior da Figura.

Apêndice C

Análise da eficiência t do sistema para Q constante e m variável

Em um caso limite, é possível dizer que, se forem considerados agrupamentos de igual tamanho em um ambiente $\mathcal{W}$ (no que diz respeito ao número de nós), espalhamento médio (que pode ser obtido por meio das distâncias relativas de cada nó ao centro geométrico do agrupamento) aproximadamente igual entre os nós sensores e a pouca ocorrência de obstáculos que desviem o robô da sua trajetória, o tempo t necessário para realizar um ciclo é aproximadamente igual para todos os robôs. Assim:

$$\eta_k \cong \eta_z \,,$$

onde η_k representa o tempo de realização de um ciclo pelo robô R_k, $1 \leq k, z \leq m$ e $k \neq z$.

Nesse ponto, pode-se definir o tempo necessário para que todos os robô coletem uma determinada quantidade de dados como (considerando-se C_i igual para todos os nós sensores e Q_k igual para todos os robôs):

$$\left\lceil \frac{\sum_{i=2}^{n} C_i}{\sum_{k=1}^{m} Q_k} \right\rceil \eta \,, \tag{C.1}$$

onde m é o número de robôs, Q_k é a capacidade do robô R_k, η é o ciclo para todos os robôs e $1 \leq k \leq m$.

Considere um ambiente no qual tem-se $n = 30$, $Q_k = 10$, $C_i = 1$, para $1 \leq k \leq m$

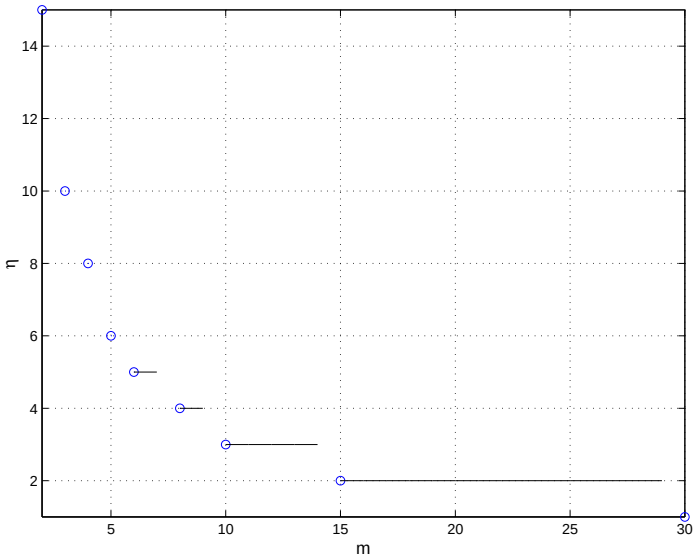

Figura C.1: Pareto que mostra os valores de η em função do número de robôs. Na prática espera-se que, em certo ponto, à medida que o número de robôs aumente o tempo de realização das tarefas também aumente devido a possibilidade dos robôs interferirem na operação uns dos outros.

e $2 \leq i \leq n$. Utilizando a eq. C.1 chegamos a:

$$\begin{aligned}
6 \text{ robôs:} \quad & \lceil 60/(6 \times 10) \rceil \, \eta = \lceil 1.0 \rceil \, \eta \; = 1\eta \\
5 \text{ robôs:} \quad & \lceil 60/(5 \times 10) \rceil \, \eta = \lceil 1.2 \rceil \, \eta \; = 2\eta \\
4 \text{ robôs:} \quad & \lceil 60/(4 \times 10) \rceil \, \eta = \lceil 1.5 \rceil \, \eta \; = 2\eta \\
3 \text{ robôs:} \quad & \lceil 60/(3 \times 10) \rceil \, \eta = \lceil 2.0 \rceil \, \eta \; = 2\eta \\
2 \text{ robôs:} \quad & \lceil 60/(2 \times 10) \rceil \, \eta = \lceil 3.0 \rceil \, \eta \; = 3\eta \\
1 \text{ robô:} \quad & \lceil 60/(1 \times 10) \rceil \, \eta = \lceil 6.0 \rceil \, \eta \; = 6\eta
\end{aligned}$$

onde nota-se que para 3, 4 ou 5 robôs o número de ciclos é o mesmo. Logo, para o caso em que Q permanece constante com a variação do número m de robôs (e igual para todos os robôs) pode-se dizer que a *eficiência t do sistema é diretamente relacionada ao número de ciclos η calculados pela equação C.1.*

Pode-se compreender o enunciado anterior, constatando-se que para 3 robôs será necessário que cada um deles recolha as informações em dois ciclos (2η), pois 3 robôs com capacidades máximas $C_i = 10$ poderão visitar 30 sensores no primeiro ciclo e

mais outros 30 sensores no segundo ciclo. Já para 4 robôs, no primeiro ciclo será possível atender 40 sensores (4 robôs com $C_i = 10$) e para o segundo ciclo restarão mais 20 sensores a serem visitados. Porém nesse caso apenas dois robôs irão atuar, enquanto que os outros 2 robôs ficarão inoperantes no segundo ciclo. A eficiência do sistema, de qualquer forma fica inalterada. Esses resultados podem ser encontrados em [Soares et al., 2004] e foram obtidos usando a Heurística de Varredura.

Em um caso limite, com a seguinte configuração: $n = 30$, $Q_k = 1$, $C_i = 1$, para $1 \leq k \leq m$ e $2 \leq i \leq n$, pode-se chegar ao gráfico teórico mostrado na Figura C.1. É importante salientar que a equação C.1 exige que todos os robôs tenham capacidades C_i de igual valor, pois de outra forma a $\eta_k \cong \eta_z$ não será verdadeira para todos os valores de k e z. Na prática, entretanto, espera-se que a eficiência de um sistema dessa natureza não tenha esse comportamento. Mesmo sem a presença de obstáculos, que tenderiam a tornar imprevisível o tempo necessário para que cada robô cumprisse seu ciclo, o aumento do números de robôs também pode ter uma influência considerável. À medida que aumenta-se o número de robôs espera-se que o tempo para realização da tarefa de extração de dados diminua. Porém, em um certo ponto o aumento do número de robôs pode implicar em um aumento de tempo, visto que eles podem interferir na realização das tarefas de outros robôs (algo semelhante a um obstáculo).

Apêndice D

Implementação de detecção e desvio de obstáculos

O Algoritmo 4 mostra a implementação da detecão e desvio de obstáculos contido na Camada de Comportamentos do Método Híbrido. O laço contido nas linhas (4-20) é executado enquanto a posição $q_R(t)$ do robô não for igual à posição final q_{Alvo}. Dentro desse laço o robô estará constantemente recebendo as leituras do dispositivo de sonar, conforme mostrado na linha 5. Se um obstáculo for detectado (linha 6), o robô R irá girar com uma dada velocidade angular ω. Essa velocidade angular é especificada pelo procedimento `Define_Velocidade_Angular`. A importância desse procedimento será explicada quando o Algorimo 5 for descrito. O laço contido nas linhas (9-12) especifica que o robô R continuará o movimento rotacional com velocidade ω enquanto a presença do obstáculo for detectada. Quando os sonares do robô R não detectarem mais a presença de obstáculos, o laço contido nas linhas (13-16) será executado. Nesse laço, é especificado que a velocidade $\|v_x\|$ será aplicada em R enquanto a distância entre posição $q_R(t)$ do robô e a posição q_{obst} for inferior à Δ. Dessa forma, o robô consegue se afastar do local onde último o obstáculo foi detectado. Quando a execução desse laço terminar o robô R voltará ao laço das linhas (4-20). Nesse momento, se nenhum outro obstáculo for detectado, a linha 18 será executada.

A execução do Algoritmo 4 não é suficiente para garantir que o robô R chegue à posição q_{Alvo}. Ao detectar um obstáculo, R rotaciona com uma dada velocidade angular ω. Se ω tiver o sentido de rotação horário, R terá tendência em explorar áreas do ambiente que estejam a sua direita. Similarmente, Se ω tiver o sentido de rotação anti-horário, R terá tendência em explorar áreas do ambiente que estejam a sua esquerda. Dependendo do controle aplicado e da forma do obstáculo, é possível

Algoritmo 4 Desvio de Obstáculos

1: $\omega \leftarrow$ `cte`.
2: $v_x \leftarrow$ `cte`.
3: $v_y \leftarrow 0$.
4: ENQUANTO $q_R(t) \neq q_{Alvo}$ FAÇA
5: Ler dados do dispositivo de sonar.
6: SE obstáculo for detectado ENTÃO
7: $q_{obst} \leftarrow q_R(t)$.
8: `Define_Velocidade_Angular()`
9: ENQUANTO obstáculo for detectado FAÇA
10: Aplicar velocidade ω no robô R.
11: Ler dados do dispositivo de sonar.
12: FIM ENQUANTO
13: ENQUANTO $\|q_R(t) - q_{obst}\| \leq \Delta$ E (obstáculo *não* for detectado) FAÇA
14: Aplicar velocidades $\|v_x\|$ e $\|v_y\|$ no robô R.
15: Ler dados do dispositivo de sonar.
16: FIM ENQUANTO
17: SENÃO
18: Mover em direção ao alvo.
19: FIM SE
20: FIM ENQUANTO

que R fique 'preso' na mesma região (ver Figura 4.8). Como ω mantém o mesmo sentido, o robô tentaria 'escapar' dessa região movendo-se para locais já visitados. Esse processo pode se repetir indefinidamente. Se no momento em que um obstáculo for detectado, o robô for capaz de identificar que passou por ali recentemente, é possível fazê-lo rotacionar com velocidade $-\omega$. Dessa forma, possivelmente, R detectará áreas livres em direções ainda não exploradas.

Com isso em mente, o Algoritmo 5, implementa uma tabela `Tab_cache` para armazenar as posições x e y pelas quais o robô passou recentemente. Essa tabela é atualizada sempre que R detecta um obstáculo, conforme pode ser verificado no Algoritmo 4. A tabela `Tab_ocorrencias` relata o número de vezes que cada entrada na tabela `Tab_cache` foi atualizada. Na linha 1 o algoritmo verifica se existe uma entrada em `Tab_cache` que seja próxima da atual posição $q_R(t)$ do robô R. Essa proximidade é dada pela distância δ. Se uma entrada for encontrada, `Tab_cache[index]` é atualizada com $q_R(t)$ e o número de ocorrências em `Tab_ocorrencias[index]` é incrementado. Na linha 4, é verificado se o número de ocorrências contido em `Tab_ocorrencia[index]` alcançou um determinado limite indicado por `THRESHOLD`. Por exemplo, se `THRESHOLD` $= 3$, R terá condições de identificar que passou 3 vezes pela mesma região. Caso o limite `THRESHOLD`

Algoritmo 5 `Define_Velocidade_Angular()`

1: SE existir um índice `index` tal que
   $\|$`Tab_cache[index]` $- q_R(t)\| \leq \delta$ ENTÃO
2: `Tab_cache[index]` $\leftarrow q_R(t)$.
3: `Tab_ocorrencia[index]` $\leftarrow$ `Tab_ocorrencia[index]` $+ 1$.
4: SE `Tab_ocorrencia[index]` $\geq$ `THRESHOLD` ENTÃO
5: `Tab_ocorrencia[1..num_entradas]` $\leftarrow 0$.
6: $\omega \leftarrow -\omega$.
7: FIM SE
8: SENÃO
9: Procurar um índice `index` tal que `Tab_ocorrencia[index]` seja mínimo.
10: `Tab_cache[index]` $\leftarrow q_R(t)$.
11: `Tab_ocorrencia[index]` $\leftarrow 0$.
12: FIM SE

tenha sido alcançado, R anula todas as entradas em `Tab_ocorrencia` e inverte o sentido de rotação da velocidade angular ω. A linha 9 é executada caso a condição presente na linha 1 seja falsa. `Tab_cache` é atualizada constantemente, substituindo posições menos frequentemente visitadas (linha 10). Ao efetuar essa substituição o número de ocorrências em `Tab_ocorrencias` para aquela entrada recebe o valor 0.