

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA
DE AUTOMAÇÃO E SISTEMAS**

Sidney Roberto Dias de Carvalho

**CONTROLE DE TOPOLOGIA EM REDES DE ROBÔS
MÓVEIS COOPERATIVOS UTILIZANDO CONSENSO**

Florianópolis (SC)
2015

Sidney Roberto Dias de Carvalho

**CONTROLE DE TOPOLOGIA EM REDES DE ROBÔS
MÓVEIS COOPERATIVOS UTILIZANDO CONSENSO**

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para obtenção do grau de “Mestre em Engenharia de Automação e Sistemas”.

Orientador: Prof. Dr. Ubirajara Franco Moreno, PGEAS – UFSC.

Florianópolis (SC)
2015

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Carvalho, Sidney Roberto Dias de
Controle de topologia em redes de robôs móveis
cooperativos utilizando consenso / Sidney Roberto Dias de
Carvalho ; orientador, Ubirajara Franco Moreno -
Florianópolis, SC, 2015.
124 p.

Dissertação (mestrado) - Universidade Federal de Santa
Catarina, Centro Tecnológico. Programa de Pós-Graduação em
Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Sistemas multi-
robôs. 3. Controle de topologia. 4. Consenso. 5. Controle
Preditivo. I. Moreno, Ubirajara Franco. II. Universidade
Federal de Santa Catarina. Programa de Pós-Graduação em
Engenharia de Automação e Sistemas. III. Título.

*Dedico este trabalho aos meus pais
que sempre deram o máximo de si
para que eu pudesse atingir meus
objetivos de forma plena.*

Agradecimentos

Agradeço primeiramente a Deus, por permitir minha saúde e sanidade necessárias para o desenvolvimento deste trabalho. Agradeço, também, aos meus amigos e colegas de curso, que sempre me auxiliaram, ao meu orientador o professor Ubirajara Franco Moreno, pela paciência e auxílio durante o desenvolvimento deste projeto, à minha família, da qual sempre obtive apoio nos momentos em que houve necessidade e por fim, a todos que de alguma forma contribuíram para o desenvolvimento deste trabalho.

*“O segredo da criatividade é saber
como esconder as fontes”.*

– Albert Einstein

Resumo

Em grupos de robôs móveis cooperativos, os chamados sistemas multi-robôs, a comunicação é um fator de extrema importância para a correta alocação e realização das tarefas. Essa comunicação é determinada diretamente pela disposição geográfica dos robôs uns em relação aos outros, a chamada topologia de comunicação. O controle da topologia de comunicação em um grupo de robôs permite que certas características da rede de comunicação sejam enfatizadas ou anuladas de acordo com a movimentação dos robôs que a compõem. Neste trabalho são apresentadas duas abordagens para controle de topologia em redes de robôs móveis, em função de quais propriedades dessas redes se deseja exaltar: o controle de topologia para a minimização da comunicação, que possibilita a redução do consumo de energia e da interferência causada pelos processos de comunicação; e o controle de topologia para a manutenção da conectividade, que garante condições para a não desconexão da rede, mesmo que esta esteja sob a influência de instabilidades. Através de um controle de conectividade baseado em consenso, a ação dos algoritmos de controle da topologia é aplicada aos robôs de maneira descentralizada, garantindo que as propriedades desejadas ocorram. São realizados simulações e testes com robôs reais, comprovando a eficiência dos algoritmos propostos em garantir as propriedades topológicas a eles associadas.

Palavras-chave: Sistemas multi-robôs, controle de topologia, consenso, otimização, controle preditivo.

Abstract

In cooperative robot systems, also known as multi-robot systems, the communication is an extremely important factor for the correct allocation and execution of the robot tasks. This communication is directly determined by the geographic position of the robots in relation each other, which is called communication topology. The topology control can be used to change aspects of the communication topology, allowing that some network characteristics are canceled or exalted, according with the robot's movement in the network. This work presents two approaches for topology control in mobile robot networks that ensure certain properties: the topology control for minimization of the communication, reducing the consumption of energy and the interference caused by radio communication; and the topology control for the connectivity maintenance, ensuring conditions for do not disconnection, even under unstable environments. Through of a connectivity control based on consensus, the action of topology control algorithms is applied to the robots in a decentralized way, ensuring the existence of the desired properties. Finally, are made simulations and tests with real robots, proving the efficiency of the proposed algorithms to ensure the functions assigned to them.

Keywords: Multi-robot systems, topology control, consensus, optimization, model predictive control.

Lista de Figuras

1.1	Aspectos do controle de topologia para minimização e maximização da conectividade.	3
2.1	Grafo representando a topologia de comunicação entre robôs de um grupo.	9
2.2	Estrutura básica de um controlador preditivo. Adaptado de [1].	12
2.3	Comportamento de um controlador preditivo. Adaptado de [1].	13
2.4	Topologia de comunicação proposta.	18
2.5	Configuração inicial da topologia: os círculos representam áreas de comunicação e as setas o fluxo da informação.	19
2.6	Variação da posição no plano após 200 iterações	19
2.7	Variação da posição no tempo após 200 iterações. . . .	20
3.1	Topologia de comunicação definida através do raio de comunicação de um robô e seus vizinhos.	22
3.2	<i>MST</i> gerada para escopo de informação global (linhas pretas) e local a <i>1-hop</i> (linhas pretas e linha tracejada), as linhas em cinza indicam as arestas iniciais do grafo.	29
3.3	Alocação de posição entre dois robôs que são vizinhos na <i>MST</i>	32
3.4	Alocação de posição entre dois robôs não vizinhos na <i>MST</i>	32
3.5	Situações para adição de novos <i>links</i>	35
3.6	Relação entre o controle de topologia para a minimização da comunicação e o controle de movimento de um robô.	37
3.7	Topologia de comunicação no estado inicial: círculos em torno dos robôs indica a área de comunicação de cada um.	39

3.8	Topologia de comunicação após algumas iterações do controle de topologia.	40
3.9	Topologia de comunicação em seu estágio final, após a execução do controle de topologia.	41
3.10	Diferenças de atuação do algoritmo (3.8a) no caso de conhecimento global e no caso de conhecimento local.	44
3.11	Nós críticos detectados sob escopo de informação local.	46
3.12	O robô 3 detecta sua criticidade e cria uma ligação virtual entre os robôs 1 e 4 (linha tracejada).	48
3.13	Substituição e remoção de <i>link</i> virtual quando uma ligação melhor é detectada.	49
3.14	Remoção de <i>link</i> virtual associado a um nó anteriormente crítico.	50
3.15	Remoção de <i>link</i> virtual após um de seus integrantes perder comunicação com o nó crítico associado.	51
3.16	Remoção de <i>link</i> virtual após o nó crítico a ele associado perder a comunicação com seus membros.	51
3.17	Relação entre o controle de topologia para a manutenção da conectividade e o controle de movimento de um robô.	54
3.18	Configuração inicial da topologia: os círculos representam a área de comunicação de cada robô.	56
3.19	Topologia resultante para diferentes perspectivas após a execução do algoritmo de resolução do TSP: linhas tracejadas indicam ligações virtuais.	57
3.20	Evolução do controle de topologia em quatro momentos distintos: linhas tracejadas com setas indicam ligações virtuais direcionadas.	59
3.21	Evolução do controle de topologia utilizando <i>links</i> virtuais.	62
4.1	Grafos utilizados como referência.	69
4.2	Configuração inicial da topologia: círculos representam a área de comunicação de cada robô e as setas indicam troca de informação direcionada.	72
4.3	Topologia resultante após 100 iterações do controle de topologia.	73
4.4	Custo energético da rede durante 100 iterações.	74
4.5	Cobertura média das arestas da rede durante 100 iterações.	75
4.6	Variação do número de árvores geradoras da rede durante 100 iterações.	75

4.7	Topologia de comunicação em dois instantes diferentes.	77
4.8	Variação do custo energético da rede.	78
4.9	Variação da cobertura média da rede.	78
4.10	Configuração inicial da topologia: círculos em torno dos robôs indicam a área de cobertura de cada um.	80
4.11	Estado da rede após 200 iterações.	81
4.12	Variação da conectividade algébrica durante 250 iterações.	82
4.13	Variação da resistência efetiva durante 250 iterações.	82
4.14	Variação do número de árvores geradoras durante 250 iterações.	83
4.15	Variação do número de <i>links</i> durante 250 iterações.	84
4.16	Variação da conexão algébrica durante a remoção de cada um dos robôs.	86
4.17	Topologia resultante quando o robô 3 perde a comunicação com todos os outros.	86
4.18	Topologia resultante quando o robô 1 perde a comunicação com todos os outros.	87
4.19	A bi-conectividade é restaurada na rede.	88
4.20	Topologia resultante quando um novo robô é detectado.	89
4.21	Configuração inicial da topologia com 100 robôs dispostos de maneira aleatória.	89
4.22	Topologia resultante após 500 iterações do controle de topologia.	90
4.23	Variação da conectividade algébrica durante a simulação.	91
4.24	Variação da resistência efetiva durante a simulação.	91
4.25	Variação do número de <i>links</i> durante a simulação.	92
4.26	Robôs e ambiente de testes: à esquerda P3DX e à direita P3AT.	93
4.27	Topologia resultante após 100 iterações do controle de topologia.	95
4.28	Topologia resultante após 100 iterações do controle de topologia.	96
4.29	Topologia resultante após os robôs 1 e 3 perderem a comunicação com todos os demais.	96
4.30	Estrutura bi-conexa restaurada após ação do controle de topologia.	97
A.1	Fluxo de execução do simulador.	112

B.1	Determinação do erro angular e linear em relação à referência.	114
C.1	Grafo de comunicação entre a <i>ros_interface</i> e os robôs.	116

Lista de Tabelas

2.1	Configuração inicial de posição e velocidade.	17
3.1	Configuração inicial dos robôs.	38
3.2	Configuração final dos robôs.	39
3.3	Configuração inicial de posição, raio de comunicação e raio de cobertura.	55
4.1	Medidas espectrométricas para grafos conhecidos.	70
4.2	Configuração inicial de posição, raio de comunicação e raio de cobertura (valores em metros).	71
4.3	Configuração final de posição, raio de comunicação e raio de cobertura (valores em metros).	73
4.4	Configuração inicial de posição, raio de comunicação e raio de cobertura para o teste 1.	80

Lista de Abreviaturas e Siglas

MPC	Model Predictive Control	11
RHC	Receding Horizon Control	12
LMST	Localized Minimum Spanning Tree	27
MST	Minimum Spanning Tree	27
TSP	Travelling Salesman Problem	42
ROS	Robot Operating System	65
AEC	Average Edge Coverage	69
VLINK	Virtual <i>Link</i>	79
SYROCO	Symposium on Robot Control	101
SBAI	Simpósio Brasileiro de Automação Inteligente	101

Lista de Símbolos

\mathcal{G}	Grafo arbitrário	8
\mathcal{V}	Conjunto de vértices do grafo	8
\mathcal{E}	Conjunto de arestas do grafo	8
n	Número de vértices do grafo	8
A	Matriz de adjacência	9
a_{ij}	Elemento da matriz de adjacência	9
i, j	Índice da matriz de adjacência/robô	9
\mathbb{R}	Conjunto dos números reais	9
x	Estado da informação/posição	9
k	Instante de tempo discreto	9
v	Velocidade/Sinal de controle	9
L	Laplaciano da matriz de adjacência	10
X	Vetor de estados da informação	10
V	Vetor de sinais de controle	10
Δ	Matriz de valência de um grafo	10
p	Tamanho do horizonte de predição	11
J	Funcional para o algoritmo de consenso	14
Δk	Passo temporal	14
Δv	Variação da velocidade	14
γ	Fator de penalização para variação da velocidade	14
$\ \cdot\ $	Norma arbitrária	14
$\ \cdot\ _2$	Norma euclidiana	14
\hat{X}	Vetor de estados da informação previstos	16
$X_{i,0}$	Vetor de estados da informação iniciais para o robô i	16
T, T'	Matrizes constantes	16
ΔV	Vetor de variações de velocidade	16
$V_{i,0}$	Vetor de velocidades iniciais para o robô i	16
H_i	Componente matricial da função objetivo	16
f_i	Componente vetorial da função objetivo	16
v^x	Velocidade na coordenada x	17
v^y	Velocidade na coordenada y	17

v_{min}	Limite mínimo de velocidade	17
v_{max}	Limite máximo de velocidade	17
r^{com}	Raio de comunicação	17
r^{cov}	Raio de cobertura	17
\mathcal{N}	Conjunto de vizinhos diretos de um robô	24
\mathcal{N}^1	Conjunto de vizinhos a <i>1-hop</i> de um robô	25
\mathcal{N}^h	Conjunto de vizinhos a <i>h-hop</i> de um robô	25
$A^{(j)}$	Linha <i>j</i> da matriz de adjacência	25
w_{ij}	Distância euclidiana entre <i>i</i> e <i>j</i>	27
α_{ij}	Variável de decisão binária/entrada da matriz de adjacência lógica	27
\mathcal{S}	Subconjunto de vértices	27
\tilde{d}_{ij}	Distância vetorial estimada entre <i>i</i> e <i>j</i>	27
\mathcal{A}	Matriz de adjacência lógica	29
κ_v	Vértice-conectividade	42
κ_e	Aresta-conectividade	42
\mathcal{W}	Matriz de distâncias	43
u	Variável arbitrária de valor singular	43
\mathcal{O}	Ordem de complexidade	43
λ_2	Segundo menor auto valor da matriz Laplaciana	46
\mathcal{L}	Matriz Laplaciana obtida no contexto lógico	46
$M^{(i)}$	Submatriz gerada pela eliminação da linha e coluna <i>i</i> de uma matriz de adjacência.	47
δ	Valência mínima de um grafo	67
ξ	Número de árvores geradoras	67
R	Resistência efetiva	68
Γ	Custo energético de um grafo	68
ϕ	Fator de atenuação do sinal	68
\bar{C}	Cobertura média das arestas de um grafo	69
C_{ij}	Cobertura da aresta (<i>i, j</i>)	69

Sumário

1	Introdução	1
1.1	Motivação	4
1.2	Objetivos	5
1.2.1	Objetivos Específicos	5
1.2.2	Organização	6
2	Cooperação e Consenso	7
2.1	Introdução ao Problema de Consenso	8
2.2	Otimização e Controle Preditivo	10
2.3	Resolvendo o Problema de <i>Rendezvous</i> Utilizando Consenso e MPC	13
2.3.1	Exemplo	17
3	Controle de Topologia	21
3.1	Modelo de Comunicação	21
3.1.1	Escopo de Informação Global	23
3.1.2	Escopo de Informação Local	24
3.2	Minimização da Topologia	25
3.2.1	Minimização da Topologia Utilizando Programação Linear	26
3.2.2	Minimização do Consumo Energético e da Interferência	29
3.2.3	Minimização de <i>Links</i> em Redes Dinâmicas	30
3.2.4	Controle de Adição de <i>Links</i>	34
3.2.5	Exemplo	37
3.3	Manutenção da Conectividade	40
3.3.1	Correção de Nós Críticos Utilizando o Problema do Caixeiro Viajante	42
3.3.2	Correção de Nós Críticos Utilizando Ligações Virtuais	45
3.3.3	Controle de Conectividade Utilizando Consenso	52

3.3.4	Exemplos	54
4	Resultados e Avaliação	65
4.1	Métricas para Análise de Robustez	66
4.1.1	Conectividade Algébrica	66
4.1.2	Número de Árvores Geradoras	67
4.1.3	Resistência Efetiva	67
4.1.4	Custo Energético	68
4.1.5	Cobertura Média das Arestas	68
4.1.6	Valores para Grafos Conhecidos	69
4.2	Simulações do Controle de Topologia para a Minimização do Número de <i>Links</i>	70
4.2.1	Teste 1 - Avaliação da Redução no Consumo Energético e Interferência	71
4.2.2	Teste 2 - Efetividade em Redes Densas	76
4.3	Simulações do Controle de Topologia para Manutenção da Conectividade	79
4.3.1	Teste 1 - Robustez da Topologia Resultante	79
4.3.2	Teste 2 - Convergência em Topologias Dinâmicas	85
4.3.3	Teste 3 - Desempenho em Topologias Densas	88
4.4	Testes em Plataforma Experimental	92
4.4.1	Minimização da Comunicação Utilizando a Plataforma Experimental	94
4.4.2	Manutenção da Conectividade na Plataforma Experimental	94
5	Considerações Finais	99
5.1	Produções Técnicas	100
5.2	Limitações	101
5.3	Trabalhos Futuros	101
	Referências	103
Apêndice A	Simulador para o Controle de Topologia	109
Apêndice B	Controle de Movimento de “Baixo Nível”	113
Apêndice C	Interface de Comunicação com o ROS	115
Apêndice D	Gerador de Grafos Aleatórios	119

Capítulo 1

Introdução

Os desafios humanos na busca por soluções a problemas complexos mostram uma linha de trabalho voltada à distribuição de tarefas. A divisão de um problema complexo em problemas menores, que podem ser solucionados por vários indivíduos, reduz a complexidade inicial da tarefa à colaboração entre esses indivíduos para se atingir um objetivo compartilhado, que representa a solução comum esperada.

Em robótica móvel o conceito de trabalho colaborativo ocorre pela necessidade de se realizar tarefas complexas que um único robô não consegue resolver. Sistemas multi-robôs podem executar tarefas com custo menor e com maior flexibilidade que robôs trabalhando individualmente. No entanto, manter a colaboração entre todos os indivíduos de um sistema como este exige que uma série de ações sejam efetuadas, ações estas que muitas vezes não podem ser realizadas de maneira distribuída, prejudicando a autonomia de tais robôs [2]. Além desse problema, há aqueles relacionados à comunicação e à sua manutenção durante a realização de tarefas colaborativas, que são capazes de interferir na forma como os robôs interagem uns com os outros, representando um grande obstáculo na integração entre diversos robôs.

Dentre as diversas tarefas que podem ser solucionadas por grupos de robôs colaborativos, estão o monitoramento, vigilância ou exploração de uma região, operações de busca e resgate de vítimas ou alvos em áreas desconhecidas, aplicações militares, entre outras. Todas essas tarefas exigem a troca de informação entre os robôs do grupo para que um estado consensual de algumas destas informações seja atingido entre eles, isto possibilitará a ação colaborativa sem prejudicar a autonomia de cada robô no grupo [2]. Tradicionalmente, as diversas abordagens colaborativas encontradas na literatura são classificadas entre duas estratégias principais:

- *cooperação*: ação colaborativa visando a realização de um objetivo comum para todos os integrantes do sistema multi-robô;
- *coordenação*: interação entre os robôs para definir as ações a serem realizadas por cada um, auxiliando no processo de tomada de decisão no grupo.

Entre os principais trabalhos envolvendo cooperação e coordenação em sistemas multi-robôs estão os apresentados por Murray [3], Lucas et al. [4] e Cao et al. [5].

Ambas as estratégias são colaborativas e podem ser realizadas de maneira centralizada e descentralizada. Em sistemas multi-robôs é comum a utilização de estratégias centralizadas, apesar das descentralizadas permitirem uma maior autonomia para cada robô do grupo. Isso ocorre porque, dependendo da tarefa a ser executada, há uma série de restrições que não podem ser garantidas quando se utiliza estratégias de controle descentralizadas.

Na abordagem descentralizada cada robô calcula sua ação baseado apenas na troca de informações com os vizinhos, enquanto que na abordagem centralizada, existe uma entidade de gerenciamento centralizado, que possui a informação global sobre o estado do grupo de robôs e determina, sozinha, quais serão as ações de cada robô.

Independentemente se a colaboração é realizada de forma centralizada ou descentralizada, a troca de informação é um fator decisivo para a realização da tarefa compartilhada. Falhas na comunicação oriundas de instabilidades provocadas pela presença de obstáculos no ambiente ou por faltas nos dispositivos de comunicação podem comprometer a realização de tarefas. Neste aspecto, a topologia da rede de comunicação é determinante para que os robôs completem suas tarefas mesmo quando expostos a problemas de comunicação como esses.

Como os robôs são móveis e seus raios de comunicação são limitados, a partilha de informação em um grupo de robôs é determinada diretamente pela topologia de comunicação apresentada por esse grupo, que é oriunda da localização espacial e, portanto, da distância entre cada um dos robôs. Então, ajustar a posição desses robôs, objetivando estruturar uma melhor configuração da topologia de comunicação, é uma alternativa para solucionar os problemas específicos dessa rede de comunicação que tornam o grupo de robôs vulnerável.

De acordo com Burkhart et al. [6], o controle ótimo de topologia é formado pelo equilíbrio entre a manutenção da conectividade e a minimização do número de *links* na rede. Entretanto, ambas as abordagens são conflitantes, desta forma sempre que se optar por atuar preferencialmente com uma das abordagens a outra será anulada pro-

porcionalmente. Esta relação é descrita na Figura 1.1.

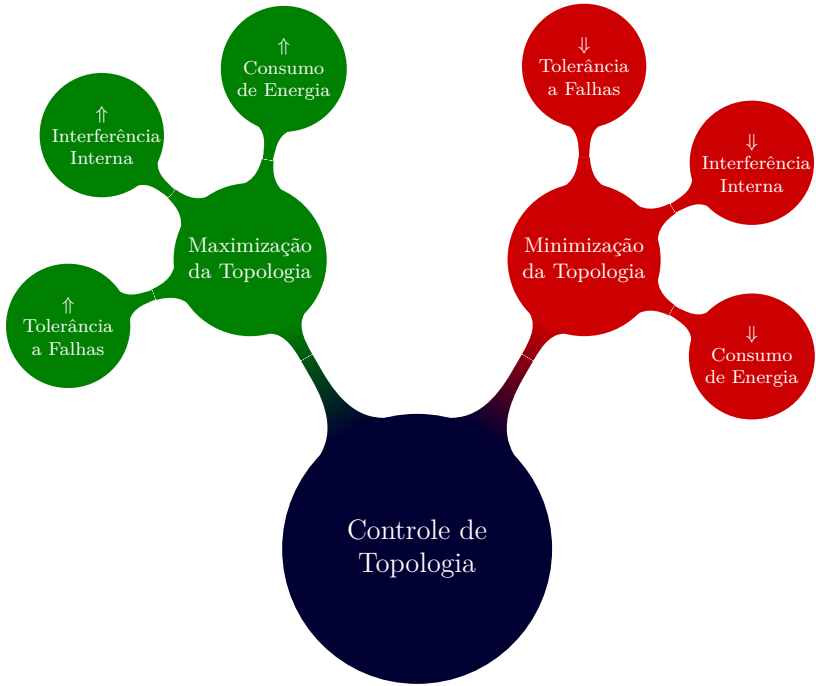


Figura 1.1: Aspectos do controle de topologia para minimização e maximização da conectividade.

No entanto, é comum na literatura, os trabalhos serem mais objetivos, optando por apenas uma abordagem, de acordo com o tipo de problema que se pretende resolver. Os trabalhos de Li et al. [7], Ahmadi e Stone [8], e Wagenpfeil et al. [9], por exemplo, tratam especificamente do controle de topologia para a manutenção da conectividade, utilizando para isso ferramentas oriundas de teoria dos grafos e redes de computadores. Já os trabalhos de Casteigts et al. [10], Hassan e Abuhaiba [11], e Halldórsson e Tokuyama [12], apresentam abordagens relacionadas à minimização da topologia de comunicação para a redução do consumo de energia bem como a redução da interferência causal de cada indivíduo na rede, utilizando conceitos similares aos apresentados nos trabalhos anteriores.

1.1. Motivação

Visando construir um ambiente colaborativo mais eficiente em sistemas multi-robôs, pode-se utilizar estratégias para controlar a topologia de comunicação em prol da tolerância a certas instabilidades que possam prejudicar as atividades colaborativas destes sistemas. Se esse controle de topologia for dinâmico, adaptando-se à variação da topologia de comunicação, tem-se um sistema robusto, com capacidades de atuação bastante extensas.

A teoria de consenso pode fornecer ferramentas suficientes para tornar o controle de topologia adaptável à dinâmica da comunicação entre robôs de forma descentralizada. Para tal, é preciso que o problema de controle de topologia seja formulado como um problema de consenso e, então, alguma lei de controle de movimento seja elaborada de forma que os robôs se movimentem de maneira coordenada para garantir que a propriedade desejada ao grafo de comunicação da rede seja atingida de maneira global. Uma revisão sobre a teoria de consenso em sistemas colaborativos pode ser encontrada em Ren et al. [13] e Olfati-Saber et al. [2], e condições de convergência para problemas de consenso são apresentadas em Ren e Beard [14].

No trabalho de Zavlanos et al. [15], é apresentado um algoritmo de controle de topologia descentralizado para a manutenção da conectividade em redes dinâmicas de robôs móveis durante a realização de flocagem¹. Basicamente, esse algoritmo utiliza uma técnica de leilão para definir um conjunto de *links* desnecessários a serem removidos da rede sem que haja a desconexão entre os robôs. A remoção ocorre quando ligações apresentam-se com um comprimento maior do que uma certa margem de segurança, que é menor do que o raio de comunicação dos robôs. Desta forma, a desconexão controlada ocorre apenas em nível lógico, possibilitando uma rápida adaptação do grupo de robôs às situações instáveis durante a realização da flocagem sem que haja desconexão entre eles. Também é proposto um algoritmo de controle de movimento baseado em campos potenciais para que os robôs ajam de maneira coordenada e não percam a comunicação uns com os outros.

Casteigts et al. [16] apresenta um algoritmo de controle de movimento para robôs com dinâmica de primeira ordem que garante a construção de topologias tolerantes a falhas de comunicação, no entanto, as torna mais densas e sujeitas a interferências causadas pelas diversas sobreposições das áreas de comunicação dos robôs. Seu algoritmo é

¹Flocagem (do inglês *flocking*) é um termo utilizado para designar comportamento grupal coordenado.

descentralizado e se baseia em campos potenciais, utilizando conceitos da *Triangularização de Delaunay* para movimentar os robôs, formando estruturas triangulares que, no contexto geral, tornam a topologia resistente à perda de *links* e robôs, sem que haja a desconexão da rede.

Utilizando consenso e controle preditivo, o trabalho de Ordoñez et al. [17] aborda cooperação entre robôs para que estes possam atingir um ponto de referência conhecido apenas por alguns deles. Sua estratégia suporta falhas parciais na comunicação e se comporta de maneira adequada, mesmo sob a influência de instabilidades nos dispositivos de comunicação. O trabalho de Correia et al. [18], utiliza o mesmo conceito apresentado por Ordoñez, para controlar a formação em um grupo de robôs móveis autônomos de forma descentralizada durante o seguimento de trajetória.

Com base nesses trabalhos, percebeu-se que o problema de controle de topologia poderia ser tratado como um problema de consenso a ser solucionado com técnicas de controle preditivo, como as apresentadas por Ordoñez et al. [17]. Técnicas estas que, como visto nos trabalhos referenciados, são suficientes para que os robôs de um sistema multi-robô ajam de maneira colaborativa e descentralizada para garantir determinadas propriedades ao grafo de comunicação referente à topologia que se quer tratar.

1.2. Objetivos

O principal objetivo deste trabalho é desenvolver, utilizando ferramentas providas pela teoria de consenso e controle preditivo, um algoritmo para o controle de topologia que permita a um grupo de robôs móveis ajustar suas posições de acordo com um critério pré-definido de maneira descentralizada, a fim de que o grafo de comunicação possa ser tolerante à falhas e instabilidades, produza uma menor interferência oriunda da sobreposição de áreas de transmissão diferentes e possua um custo energético mínimo capaz de manter a sua estrutura. Como estes interesses são, em parte, conflitantes, não espera-se que os algoritmos desenvolvidos abordem todas as propriedades descritas anteriormente, podendo apresentar-se de forma modular.

1.2.1. Objetivos Específicos

Os objetivos específicos que constituem este trabalho são os seguintes:

- Desenvolver um método de controle de topologia para minimi-

zar a conectividade, a interferência causada pela sobreposição de áreas de transmissão e o consumo de energia em uma rede de robôs móveis;

- Desenvolver uma estratégia de controle de topologia para garantir tolerância a falhas de comunicação para redes de robôs móveis em ambientes instáveis;
- Adaptar o algoritmo de Ordoñez et al. [17] para garantir a conectividade durante a realização do controle de topologia e que as propriedades a serem inseridas na rede sejam realmente efetivadas;
- Realizar testes e simulações que comprovem a eficiência dos algoritmos propostos;
- Testar os algoritmos propostos em plataformas robóticas experimentais, utilizando robôs terrestres para verificar a factibilidade de implantação destes algoritmos em aplicações reais.

1.2.2. Organização

O restante do documento é organizado da seguinte maneira:

- No capítulo 2 é apresentada uma pequena introdução à teoria de grafos, consenso, otimização e controle preditivo, e o método de Ordoñez et al. [17] é exposto e analisado detalhadamente;
- No capítulo 3, a abordagem proposta nessa dissertação é detalhada;
- No capítulo 4 são elaborados alguns experimentos e simulações a fim de avaliar o desempenho dos algoritmos propostos;
- Finalmente, no capítulo 5 são apresentadas considerações finais a cerca dos algoritmos apresentados e propostas de trabalhos futuros baseados nos resultados até então obtidos.

Capítulo 2

Cooperação e Consenso

Em um ambiente cuja complexidade das tarefas a serem realizadas exige muito esforço físico e computacional, é comum a utilização de técnicas que permitam a divisão dessas tarefas em sub-tarefas que possam ser realizadas de maneira distinta. Estratégias para distribuição das tarefas têm sido aplicadas aos mais diversos segmentos da indústria, e na robótica móvel não é diferente, pois, no geral, a divisão de uma tarefa complexa em várias mais simples permite um barateamento e simplificação dos robôs que a realizarão [5].

A divisão na execução de uma tarefa qualquer entre vários indivíduos (sejam robôs ou não) exige algum tipo de colaboração entre eles, pois as sub-tarefas ainda são partes de um objetivo comum a ser atingido. A cooperação e a coordenação são abordagens colaborativas costumeiramente utilizadas para a resolução de problemas de uma maneira distribuída, e o consenso é uma técnica que permite aos indivíduos de um grupo, obter valores similares para suas ações, concretizando a ação conjunta em prol da realização de alguma tarefa [2, 3, 19].

A seguir é apresentado uma introdução à teoria de consenso e à teoria de grafos, destacando a importância do consenso na realização de tarefas de maneira colaborativa. Então, é apresentado um pequeno esboço a cerca de otimização e suas ramificações aplicadas no controle de sistemas dinâmicos lineares. Finalmente, na seção 2.3, o algoritmo de consenso descentralizado baseado em controle preditivo proposto por Ordoñez et al. [17] para a resolução do problema de *Rendezvous* é apresentado.

2.1. Introdução ao Problema de Consenso

Consenso, segundo a língua portuguesa, designa igualdade, conformidade de opiniões ou pensamento comum a cerca de algo [20]. Já no contexto de controle cooperativo, consenso pode ser definido como o compromisso entre os integrantes de um grupo a cerca de um objetivo ou tarefa cujo conhecimento é coerente entre todos os membros desse grupo [17]. É nesse contexto, que a definição de consenso nos interessa, pois nos permite consolidar a ação cooperativa entre os indivíduos de um grupo.

A principal vantagem do consenso para a realização de tarefas cooperativas está na independência associada a cada membro do grupo no processo de concordância. Isso possibilita o desenvolvimento de algoritmos cooperativos completamente distribuídos, potencializando a ação de vários robôs na resolução de tarefas complexas.

O consenso em um sistema distribuído no geral, pode ser definido de diversas maneiras, de acordo com as variáveis de consenso, que tratam da informação a ser modelada em prol da colaboração entre tais robôs. Essa informação, na literatura, recebe o nome de estado da informação, e pode assumir diversos valores e dimensões de acordo com o problema a ser resolvido [13].

A estrutura de comunicação de um grupo de robôs móveis, por outro lado, pode ser descrita através de um grafo ou suas variações algébricas. Com isso, uma série de propriedades oriundas da teoria dos grafos podem ser utilizadas para construir redes com características específicas.

Seja um grupo de robôs definido pelo grafo da Figura 2.1 em que cada nó representa um robô e cada aresta indica um *link* de comunicação entre eles. A direção da aresta representa o fluxo de informação entre os robôs, assim pode-se ter robôs que recebem e enviam informação (e.g. 1, 2 e 4) e robôs que apenas recebem (e.g. 5) ou apenas enviam dados para seus vizinhos (e.g. 3).

Essa forma de representar uma rede de robôs móveis pode ser modelada algebricamente através da matriz de adjacência, que relaciona cada vizinho através da sua capacidade de se comunicar com os demais membros do grupo. Um grafo arbitrário pode ser definido como $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, onde \mathcal{V} é o conjunto de vértices ou nós desse grafo, no caso de uma rede de robôs, trata-se dos robôs, enquanto que \mathcal{E} é o conjunto de arestas, ou seja, as ligações entre cada robô pertencente à rede. Desta forma, define-se a matriz de adjacência denotada por A para um

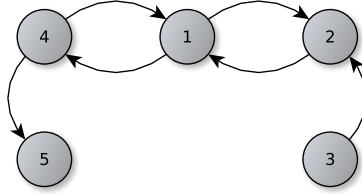


Figura 2.1: Grafo representando a topologia de comunicação entre robôs de um grupo.

grafo \mathcal{G} , como:

$$A = [a_{ij}] \in \mathbb{R}^{n \times n},$$

onde n é o número de nós da rede e a_{ij} representa a relação de vizinhança entre o nó i e o nó j com $a_{ij} > 0$ se $(i, j) \in \mathcal{E}$ e $a_{ij} = 0$, caso contrário [21].

Com isso, para o grafo da Figura 2.1, tem-se a seguinte matriz de adjacência:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

onde cada linha que possui o valor 1 indica que o nó equivalente ao índice da linha recebe informação do nó representado pelo índice da coluna respectiva. O valor “1” pode ser substituído por qualquer outro valor maior que zero, admitindo-se, nesse caso, regras de ponderação que podem ser utilizadas para representar as características físicas da rede.

Genericamente, tem-se que o consenso em um grupo de robôs móveis pode ser definido por um algoritmo que convirja o estado da informação de cada robô para um valor comum desejado. Desta forma, pode-se definir a dinâmica do algoritmo de consenso para um grupo de n robôs móveis representados por uma matriz de adjacência como:

$$v_i[k] = - \sum_{j=1}^n a_{ij} (x_i[k] - x_j[k]), \quad i = 1, \dots, n, \quad (2.1)$$

onde a_{ij} é o valor na matriz de adjacência para os robôs i e j , k é o instante de tempo discreto, e x é a variável sobre a qual é realizado o consenso [2].

A atualização da informação sobre a qual se deseja obter o consenso pode ser modelada como um sistema linear de primeira ordem, que utiliza como entrada o resultado da equação (2.1), resultando em:

$$x_i[k+1] = x_i[k] + v_i[k]\Delta k,$$

onde Δk é o passo de atualização.

É interessante notar que esse algoritmo pode ser elaborado para que apresente comportamento descentralizado, pois cada robô i necessita apenas da informação local a cerca de sua vizinhança (a_{ij}), o que o torna ideal para implementações reais. Para simplificar estas implementações é comum a representação da equação (2.1) através do laplaciano da matriz de adjacência que descreve a rede, originando a seguinte equação:

$$V = -LX,$$

onde $L = \Delta - A$ é a matriz laplaciana, com $\Delta = \text{diag}(\sum_{j=1}^n a_{ij})$ sendo a matriz de valência para o grafo \mathcal{G} , $X = [x_0, \dots, x_n]^\top$ é o vetor de estados da informação e $V = [v_0, \dots, v_n]^\top$ o vetor de sinais de controle para todos os robôs.

Uma vez que o consenso sobre x é atingido, então o estado da informação para todos os robôs da rede será o mesmo, desta forma:

$$\lim_{k \rightarrow \infty} |x_i[k] - x_j[k]| = 0 \quad i, j = 1, \dots, n.$$

No entanto, para se atingir esse consenso global, algumas condições devem ser garantidas. De fato, a maior influência ao processo de consenso sobre uma determinada variável, está na topologia de comunicação adotada pelos robôs da rede. É imprescindível que haja pelo menos uma árvore de extensão orientada na rede para que a informação se propague de forma correta [22].

Uma árvore de extensão é um subgrafo que conecta todos os vértices do grafo de que esta faz parte e é orientada quando esse grafo é direcionado [21]. Assumindo que o estado da informação desejado é conhecido apenas por alguns robôs, então essa informação deve ser a raiz da árvore de extensão orientada. Desta forma, pode-se concluir que em caso de falhas de comunicação permanentes, o consenso não se realizará, sendo esta umas das principais restrições para esse tipo de algoritmo.

2.2. Otimização e Controle Preditivo

Otimização é uma área da matemática aplicada que visa encontrar valores ótimos para variáveis de decisão constituintes de um deter-

minado modelo matemático ao mesmo tempo que assegura a factibilidade de certas restrições impostas por este modelo [23]. Seu uso é muito extenso, indo desde aplicações comerciais, financeiras até aplicações em engenharia de produção e de transportes.

Basicamente, para ser definido como um problema de otimização o mesmo deve conter certas características: possuir uma **função objetivo** que descreva o comportamento a que se quer otimizar; conter **variáveis de decisão**, que poderão ser definidas para otimizar a função objetivo; e possuir **restrições** que caracterizem o espaço de soluções factíveis para as variáveis de decisão, a fim de se garantir alguma propriedade do modelo físico.

O problema de consenso pode ser formulado como um problema de otimização, para isto basta que a função objetivo desta formulação represente a coesão entre os estados da informação disponíveis para um valor desejado. Desta forma, têm se a seguinte formulação para o problema de consenso representado na equação (2.1):

$$\begin{aligned} \min_{v_i} \quad & \sum_{j=1}^n a_{ij}(x_i - x_j) \\ \text{s.t.} \quad & a_{ij} \in \{0, 1\} \\ & j \neq i \end{aligned}$$

onde a função objetivo é definida pela diferença entre os estados da informação do robô i e de todos os seus vizinhos, a_{ij} , x_i e x_j são variáveis de decisão e $j \neq i$ e $a_{ij} \in \{0, 1\}$ são as restrições do problema.

Em teoria de controle pode-se utilizar algoritmos de otimização para se calcular sinais ótimos para um determinado problema dada certas condições, é o chamado controle ótimo. Uma variação desse tipo de controle é chamada de controle preditivo, ou *Model Predictive Control (MPC)* em que são calculadas as entradas ótimas para p instantes futuros dado o estado atual do sistema.

Basicamente o controle preditivo funciona como um otimizador temporal, que utiliza uma determinada faixa de previsões, conhecida como janela temporal, para estipular o comportamento da planta que se deseja controlar, e então calcular, para cada instante dessa janela, qual é o melhor sinal de controle que minimiza o esforço de controle e o erro em relação à referência para p instantes de tempo [1]. Em cada iteração do algoritmo de controle preditivo, apenas a primeira estimação de controle é utilizada como entrada na iteração seguinte, quando a janela temporal avança uma unidade. Esse comportamento de janela temporal deslizante é o que origina uma outra denominação pelo

qual o controle preditivo é conhecido na literatura: *Receding Horizon Control (RHC)*.

A função objetivo pode ser modelada como uma equação linear, mas é comum a utilização de uma formulação quadrática, na forma $\frac{1}{2}x^T Qx + c^T x$, com restrições lineares, o que torna o problema convexo e, portanto, menos custoso computacionalmente [23, 24]. A implementação usual encontrada na literatura segue a estrutura apresentada na Figura 2.2.

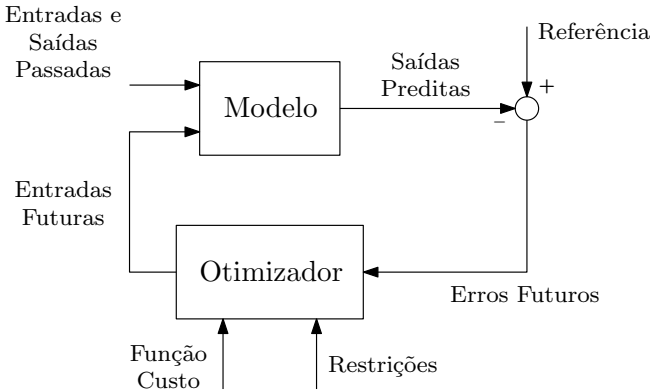


Figura 2.2: Estrutura básica de um controlador preditivo. Adaptado de [1].

Um modelo matemático é utilizado para prever as saídas futuras da planta, baseado nos valores atuais e passados. As ações de controle futuro são calculadas pelo otimizador, levando em consideração a função custo, anteriormente elaborada, assim como suas restrições. Desta forma, minimiza-se o erro da referência em relação ao estado atual e em relação aos estados futuros. A Figura 2.3 descreve esse comportamento para um sistema arbitrário: v indica o sinal de controle enquanto que x é a saída do sistema e \hat{x} é a saída estimada a partir do modelo matemático que representa a planta.

O controle preditivo baseado em janelas deslizantes possui diversas vantagens em relação ao controle tradicional, dentre elas:

- Comportamento intuitivo e ao mesmo tempo de fácil ajuste;
- Capacidade de controlar uma grande variedade de processos, desde modelos mais simples a modelos extremamente complexos, com altas taxas de atrasos e instabilidades comportamentais;
- O controlador resultante é fácil de implementar, por se tratar de uma função linear;

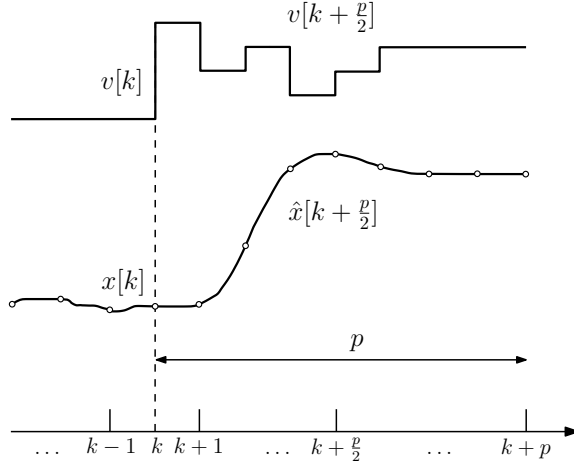


Figura 2.3: Comportamento de um controlador preditivo. Adaptado de [1].

- Extensões para o tratamento de restrições são conceitualmente simples e podem ser sistematicamente incluídos durante o design do controlador.

Essas e outras vantagens o tornam ideal para implementação prática de algoritmos que exijam comportamento cooperativo e ao mesmo tempo descentralizado, é o que o algoritmo proposto por Ordoñez et al. [17] apresenta em sua abordagem, como descrito a seguir.

2.3. Resolvendo o Problema de *Rendezvous* Utilizando Consenso e MPC

O trabalho de Ordoñez et al. [17] apresenta uma formulação baseada em controle preditivo que permite a um grupo de robôs móveis cooperar entre si para atingir um determinado ponto conhecido por apenas alguns deles, problema esse que, na literatura, é referenciado como *rendezvous* [25, 26]. Trata-se de um algoritmo de programação quadrática em que se estima, para uma determinada quantidade de robôs, as p ocorrências da variável de consenso – nesse caso a posição – que não violam as restrições do problema.

Basicamente o algoritmo consiste em um funcional J , modelado de acordo com características cooperativas que se deseja manter ao grupo de robôs, para que, através de uma ação de controle de primeira

ordem, possa-se garantir a convergência da informação em tais robôs para um mesmo estado a respeito do ponto de encontro, ou seja, garantir o *rendezvous*.

A partir da equação (2.1), pode-se adicionar um termo que corresponda ao ponto de encontro para o grupo de robôs, dessa forma incorporando a questão do *rendezvous* ao algoritmo de consenso anteriormente descrito, resultando na equação (2.2).

$$v_i[k] = - \sum_{j=1}^n a_{ij}(x_i[k] - x_j[k]) - a_{ir}(x_i[k] - x_r[k]) \quad (2.2)$$

onde $a_{ir} \geq 0$ pode ser entendido como uma entrada na matriz de adjacência estendida, indicando se o robô i conhece ou não a referência, v_i é o sinal de controle oriundo do consenso e x_r é a coordenada cartesiana desse ponto de encontro.

Reescrevendo-se o algoritmo de consenso para o problema de *rendezvous* como uma função quadrática, tem-se o funcional J composto por 3 termos independentes:

$$\begin{aligned} J_i &= \sum_{j=1}^n \sum_{k=1}^p \|x_i[k] - x_j[k]\|_{a_{ij}}^2 \\ &+ \sum_{k=1}^p \|x_i[k] - x_r[k]\|_{a_{ir}}^2 \\ &+ \sum_{k=1}^p \|\Delta v_i[k]\|_{\gamma_i}^2, \quad i = 1, \dots, n. \end{aligned} \quad (2.3)$$

onde a notação $\|x\|_Q^2 \equiv x^\top Qx$ e γ_i é um fator de ponderação para a variação da velocidade do robô i (Δv_i).

O primeiro termo da equação (2.3) procura penalizar a variação de distância entre os robôs, isso significa que quando $t \rightarrow \infty$, a distância entre i e j tenderá a zero, ou seja, esse termo implementa a questão do *rendezvous* entre os robôs vizinhos, discutido anteriormente, como uma parte da função objetivo da equação. O segundo termo é utilizado para reduzir a distância entre o robô i e a referência para o instante de predição k , assim como o primeiro termo, este busca o *rendezvous*, no entanto, neste caso é em relação a uma referência, que pode ser fixa ou variável, se houver seguimento de trajetória, por exemplo. Já o terceiro termo garante que a velocidade de i seja o mais constante possível, evitando discrepâncias de comportamento o que, na prática, obriga os robôs a se comportarem de maneira ordenada, evitando variações

muito bruscas de velocidade em determinados momentos e suavizando o movimento destes robôs.

A princípio a única restrição a ser adicionada ao problema diz respeito à saturação do sinal de controle, limitando a solução factível a valores máximos e mínimos coerentes com o comportamento desejado para os robôs. Mais adiante, outras restrições envolvendo o raio de comunicação e o raio de cobertura serão adicionadas, com maiores detalhes sobre sua adaptação no algoritmo original proposto.

Como a dinâmica integradora é de primeira ordem, então o valor resultante após a otimização do funcional J_i é a velocidade do robô i para um determinado instante de tempo, desta forma a posição de cada robô i pode ser atualizada conforme:

$$x_i[k] = x_i[k-1] + v_i[k]\Delta k$$

onde k é o instante de tempo discreto e Δk é o passo temporal.

Por se tratar de uma abordagem descentralizada não é possível conhecer os sinais de controle ótimos para os vizinhos do robô i . Assim, utiliza-se uma aproximação para os valores dos robôs vizinhos, de forma que $v_j[k] \approx v_j[0]$ e $x_j[k] \approx x_j[0] \quad \forall k = 1, \dots, p$, com isso pode-se estimar a posição dos robôs vizinhos para todos os instantes de predição.

Opcionalmente pode-se reescrever a função (2.3) na forma matricial para visualizar melhor como o algoritmo é implementado para todos os instantes de predição, com isso a dinâmica de atualização do estado da informação é escrita como:

$$\underbrace{\begin{bmatrix} x_i[1] \\ x_i[2] \\ \vdots \\ x_i[p] \end{bmatrix}}_{\hat{X}_i} = \underbrace{\begin{bmatrix} x_i[0] \\ x_i[0] \\ \vdots \\ x_i[0] \end{bmatrix}}_{X_{i,0}} + \Delta k \underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 \end{bmatrix}}_T \cdot \underbrace{\begin{bmatrix} v_i[1] \\ v_i[2] \\ \vdots \\ v_i[p] \end{bmatrix}}_{V_i}$$

os vetores estimados para a referência e para cada robô vizinho são definidos como:

$$\hat{X}_r = \underbrace{\begin{bmatrix} x_r[0] \\ x_r[0] \\ \vdots \\ x_r[0] \end{bmatrix}}_{X_{r,0}} \quad \hat{X}_j = \underbrace{\begin{bmatrix} x_j[0] \\ x_j[0] \\ \vdots \\ x_j[0] \end{bmatrix}}_{X_{j,0}}$$

e a variação do sinal de controle pode ser escrita como:

$$\Delta V_i = \begin{bmatrix} \Delta v_i[1] \\ \Delta v_i[2] \\ \vdots \\ \Delta v_i[p] \end{bmatrix} = \underbrace{\left(\frac{T}{\Delta k} \right)^{-1}}_{T'} V_i - \underbrace{\begin{bmatrix} v_i[0] \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{V_{i,0}}$$

onde T é uma matriz triangular inferior de ordem p multiplicada por Δk e T' é uma matriz auxiliar de mesma ordem, assim como todos os vetores apresentados. Reescrevendo a função objetivo, descrita pela equação (2.3), tem-se:

$$J_i = \sum_{\substack{j=1 \\ j \neq i}}^n \|X_{i,0} + TV_i - X_{j,0}\|_{a_{ij}}^2 + \|X_{i,0} + TV_i - X_{r,0}\|_{a_{ir}}^2 + \|T'V_i - V_{i,0}\|_{\gamma_i}^2$$

evidenciando-se V_i e escrevendo-se J como uma função quadrática na forma $\frac{1}{2}x^\top Qx + c^\top x$, temos:

$$J_i = \frac{1}{2}V_i^\top H_i V_i + f_i V_i$$

onde

$$H_i = \sum_{\substack{j=1 \\ j \neq i}}^n T^\top a_{ij} T + T^\top a_{ir} T + T'^\top \gamma_i T'$$

e

$$f_i = \sum_{\substack{j=1 \\ j \neq i}}^n (X_{i,0}^\top - X_{j,0}^\top) a_{ij} T + (X_{i,0}^\top - X_{r,0}^\top) a_{ir} T - V_{i,0}^\top \gamma_i T'$$

Pode-se, ainda, definir o problema de otimização quadrática como uma função relacionando os valores das coordenadas do espaço cartesiano onde é realizado o consenso, no caso, em duas dimensões, resultando na seguinte equação:

$$\begin{aligned} \min \quad & \frac{1}{2} \begin{bmatrix} V_i^x & V_i^y \end{bmatrix} \begin{bmatrix} H_i & 0 \\ 0 & H_i \end{bmatrix} \begin{bmatrix} V_i^x \\ V_i^y \end{bmatrix} + \begin{bmatrix} f_i^x & f_i^y \end{bmatrix} \begin{bmatrix} V_i^x \\ V_i^y \end{bmatrix} \quad (2.4) \\ \text{s.t.} \quad & v_{min}^x \leq v_i^x \leq v_{max}^x \\ & v_{min}^y \leq v_i^y \leq v_{max}^y \end{aligned}$$

onde $v_i^x \in V_i^x$ é a velocidade na coordenada x e v_{max} e v_{min} são as velocidades máximas e mínimas, respectivamente.

As velocidades nas coordenadas x e y são desacopladas em relação as restrições aplicadas nessa formulação, isso significa que elas poderiam ser determinadas independentemente, entretanto, como será visto mais adiante, algumas restrições podem acoplar essas coordenadas, devido a isso optou-se por criar uma formulação em que as velocidades em ambas as coordenadas possam ser calculadas simultaneamente.

2.3.1. Exemplo

Nessa seção é apresentado um exemplo simples de aplicação do algoritmo de consenso cooperativo em um grupo de robôs móveis sujeitos ao problema de *rendezvous*, onde há um ponto de encontro e apenas alguns robôs do grupo tem conhecimento dessa referência.

Seja um grupo de seis robôs omnidirecionais com as posições iniciais definidas segundo a Tabela 2.1. Os valores máximos e mínimos

Tabela 2.1: Configuração inicial de posição e velocidade.

robô	x	y	v^x	v^y
1	60	85	0	0
2	100	155	0	0
3	145	170	0	0
4	40	50	0	0
5	70	30	0	0
6	185	140	0	0

para a restrição de saturação do sinal de controle são:

$$v_i^{max} = [1 \quad 1] \quad v_i^{min} = [-1 \quad -1] \quad i = 1, \dots, 6.$$

Os robôs são heterogêneos e, portanto, possuem raios de comunicação diferentes, dados conforme:

$$r_i^{com} = [70, 90, 50, 50, 30, 30] \quad i = 1, \dots, n.$$

A topologia de comunicação é dada pelo grafo na Figura 2.4 e o nodo **ref** = [120 100] indica a posição de referência para o *rendezvous*.

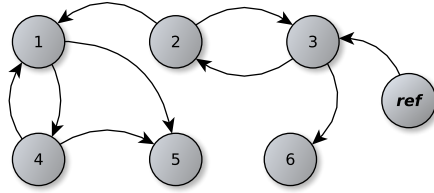


Figura 2.4: Topologia de comunicação proposta.

Todas as coordenadas são em metros e as velocidades em metros por segundo.

Como se observa na Figura 2.4, apenas o robô 3 recebe informação a respeito da referência, porém existe uma árvore direcionada partindo do robô 2 para todos os demais robôs, o que fornece a condição necessária para a realização do consenso entre todos os robôs.

Os parâmetros do algoritmo de controle são definidos como: $p = 5$ e $\Delta k = 0.5s$ onde p é o horizonte de predição e Δk é o passo temporal.

A matriz de adjacência que descreve o grafo da Figura 2.4 pode ser representada por:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

A matriz resultante é assimétrica, pois a topologia de comunicação é direcionada.

A Figura 2.5 representa a topologia descrita e o raio de comunicação de cada robô, juntamente com a referência (o ponto em vermelho). Como os raios de comunicação de cada robô são diferentes, os canais de comunicação descritos pela Figura 2.4 serão habilitados conforme os robôs se aproximam. As linhas em azul indicam comunicação bidirecional, as setas em magenta indicam comunicação direcionada, os nós pretos são nós que recebem e enviam mensagens e os nós azuis apenas recebem, o círculo em torno dos robôs indica o raio de comunicação de cada um.

São executadas 200 iterações do algoritmo, o que implica em 200 rodadas de troca de mensagens entre todos os robôs. Na Figura 2.6 observa-se a movimentação de todos os robôs de suas posições de

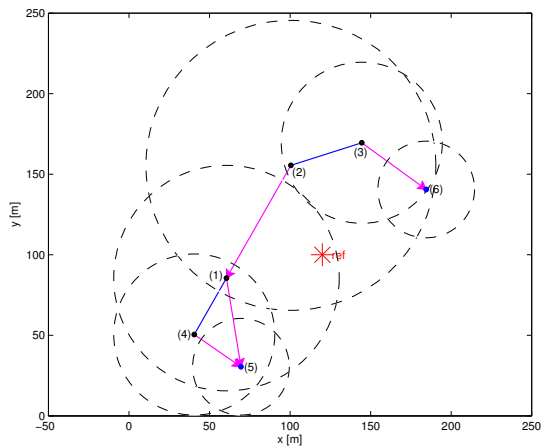


Figura 2.5: Configuração inicial da topologia: os círculos representam áreas de comunicação e as setas o fluxo da informação.

origem até o ponto de encontro.

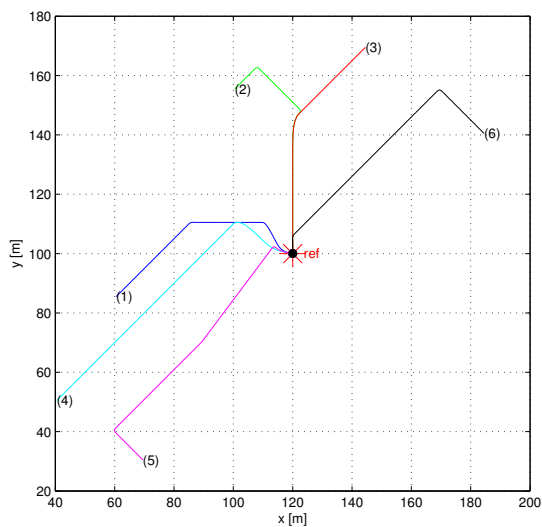


Figura 2.6: Variação da posição no plano após 200 iterações

Como se percebe na Figura 2.7, após aproximadamente 160 iterações todos os robôs atingem o ponto de referência, o que comprova

a eficácia do algoritmo em resolver o problema de *rendezvous* para um grupo de robôs.

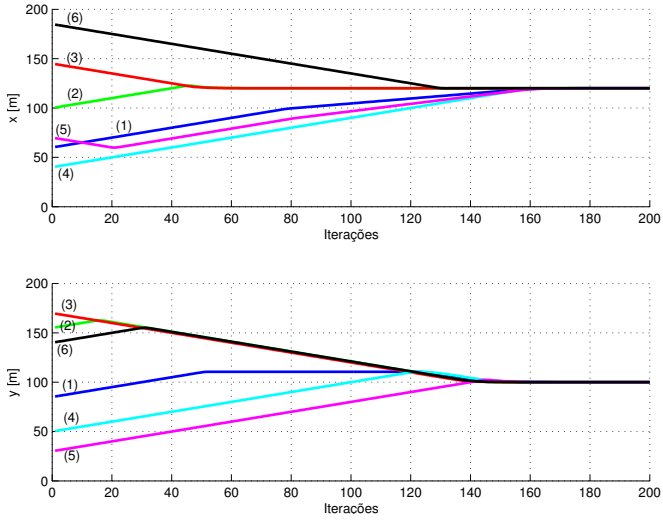


Figura 2.7: Variação da posição no tempo após 200 iterações.

Capítulo 3

Controle de Topologia

Neste capítulo, são apresentadas as contribuições deste trabalho: três algoritmos para controle de topologia em redes de robôs móveis. O primeiro algoritmo utiliza uma estratégia onde se encontra a árvore geradora mínima da rede para reduzir o número de *links* em um grupo de robôs móveis. No segundo e no terceiro algoritmo estrutura-se uma topologia tolerante a falhas de comunicação utilizando propriedades oriundas de teoria dos grafos e procura-se manter essa estrutura mesmo quando há instabilidades nos canais de comunicação dos robôs. Ambas as estratégias são elaboradas para topologias dinâmicas – onde os robôs são móveis – e utilizam um algoritmo de controle de movimento baseado no trabalho de Ordoñez et al. [17] para garantir que o controle de topologia aplicado aos robôs seja efetivo em cumprir os seus objetivos em um ambiente cooperativo.

A seguir, uma série de considerações são feitas, um modelo de comunicação determinístico é apresentado e os três algoritmos propostos para controle de topologia são detalhados. Em seguida, no capítulo 4, são realizadas algumas simulações numéricas e avaliações críticas dos algoritmos apresentados, bem como experimentos com uma plataforma robótica comercial.

3.1. Modelo de Comunicação

Em um sistema multi-robô qualquer, a topologia de comunicação e, portanto, as relações de vizinhança entre os robôs são dadas através da relação entre o raio de comunicação e o raio de cobertura de cada robô. O raio de comunicação de um robô qualquer é a distância máxima que esse robô consegue enviar informação e é proporcional à potência do rádio transmissor desse robô. Já o raio de cobertura de um robô

determina a sua área segura, na qual outro robô não deve estar. Geralmente tem-se um raio de cobertura maior do que o raio do robô e menor do que a metade do seu raio de comunicação [27]. O raio de cobertura é usado para evitar colisões ou para maximizar a cobertura de uma área de monitoramento, por exemplo.

Uma vez que todos os robôs tenham o mesmo raio de comunicação, eles estarão em uma topologia completamente bi-direcional, onde o fluxo de informação ocorre em ambos os sentidos para qualquer par de robôs conectados. No entanto, se a rede for composta por robôs heterogêneos, cujos raios de comunicação diferem, então teremos uma rede de comunicação cujo fluxo de informação pode ser direcionado.

Ambas as redes podem ser representadas por grafos, no entanto, redes direcionadas são representadas por grafos direcionados, os chamados dígrafos (como o apresentado na Figura 2.1). A Figura 3.1 representa o processo de construção da vizinhança para um robô i qualquer através do seu raio de comunicação e do raio de seus vizinhos. O robô

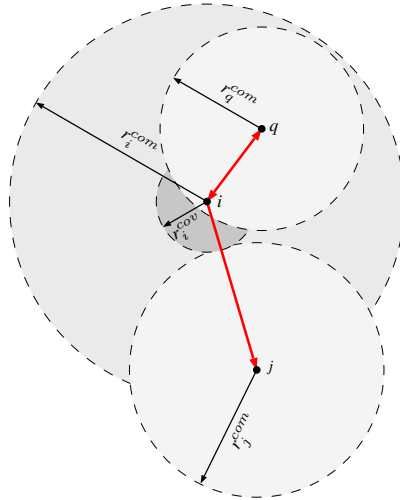


Figura 3.1: Topologia de comunicação definida através do raio de comunicação de um robô e seus vizinhos.

i recebe e envia informação para o robô q , pois ambos encontram-se dentro da área de comunicação um do outro, no entanto, o robô i não recebe informação do robô j embora o contrário ocorra, pois não se encontra dentro da área de comunicação deste, resultando em um *link* direcional de i para j . As setas em vermelho indicam a direção do fluxo

da informação, r^{com} é o raio de comunicação dos robôs e r_i^{cov} é o raio de cobertura do robô i .

Considerando uma topologia de comunicação dinâmica, geralmente encontrada em redes de robôs móveis, tem-se uma maior complexidade na definição do fluxo de informação, já que isso depende da forma como tais robôs se movimentam. O controle de topologia, então, se encarrega de gerenciar como, através da topologia de comunicação, os robôs se movimentarão para garantir certas propriedades ou realizar determinadas tarefas.

A informação a respeito da topologia de comunicação se resume à matriz de adjacência relativa ao grafo que representa esta rede. De fato, para que um robô possa atuar sobre a sua topologia de comunicação, basta que este conheça a matriz de adjacência que descreve essa topologia. Conforme a topologia de comunicação muda devido a movimentação dos robôs que a compõe, adota-se a notação $A[k] = [a_{ij}[k]]_{n \times n}$, com $a_{ij}[k] > 0$ se e somente se $(i, j) \in \mathcal{E}[k]$ para representar a sua dinâmica discreta de forma algébrica.

A dinâmica de atualização da topologia de rede pode ser descrita através da dinâmica de atualização da matriz de adjacência. Nas duas seções seguintes, essa dinâmica de atualização é detalhada sob o contexto da informação global e local.

3.1.1. Escopo de Informação Global

Em uma abordagem onde a informação a respeito da topologia e, portanto, da matriz de adjacência que representa essa topologia é global, existe um consenso sobre o estado da rede em um dado instante para todos os robôs que a compõem. Apesar de ser uma abordagem que pode ser descentralizada, cada robô deve possuir o conhecimento global sobre a rede para que isso seja possível. Entretanto, isto não é pertinente para redes compostas por muitos nodos e geralmente se restringe a aplicações em pequenos grupos de robôs ou sensores estáticos.

A vantagem de se utilizar a informação global é que pode-se atingir a configuração ótima da topologia em um determinado instante de tempo, pois admite-se que a informação em cada nodo poderá ser a mesma em algum momento. Entretanto, manter o estado da informação consistente para todos os membros da rede durante o intervalo de execução da tarefa mostra-se um desafio bastante complexo, principalmente se houver muitos nodos, interferências ou mesmo falhas de comunicação.

De maneira geral, pode-se definir a informação global como sendo

uma função local de atualização da informação sobre a qual se quer manter o consenso. Seja $A_i[k]$ a matriz de adjacência que descreve um grafo qualquer $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$ para um robô i em um instante de tempo k . Como a informação a respeito do estado da rede é global, então $\lim_{k \rightarrow \infty} |A_i[k] - A_j[k]| = 0 \forall i, j \in \{1, \dots, n\}$ e a dinâmica de atualização dessa informação para cada robô em um determinado instante de tempo pode ser descrita como em [15], da seguinte maneira:

$$A_i[k] = \bigvee_{j \in \mathcal{N}_i} (A_i[k-1] \vee A_j[k-1]) \quad (3.1)$$

onde $\bigvee(\cdot)$ é um **OU** lógico cumulativo que compara todas as instâncias de (\cdot) para o conjunto utilizado como domínio deste operador, $(\cdot) \vee (\cdot)$ é um **OU** lógico convencional e \mathcal{N}_i é o conjunto de vizinhos de i , definido como:

$$\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$$

A equação descrita em (3.1) funciona bem caso não haja remoção de *links*, no entanto, em caso de perda de comunicação ou mudanças inesperadas em relação a uma determinada vizinhança, essa dinâmica não descreverá a correta propagação do estado global da rede.

3.1.2. Escopo de Informação Local

Em uma abordagem cujo escopo da informação é local, não contemplando todos os indivíduos da rede, procura-se reduzir a necessidade da informação global através de estimativas locais a respeito de um determinado grau de vizinhança para cada robô. Em geral, restringe-se essa informação ao menor nível possível de vizinhança, no entanto, é comum a utilização de informações um pouco mais completas em determinados casos.

O grau de vizinhança de um robô é descrito através do número de saltos (em inglês *hops*) em nodos necessários para se atingir tal robô. Os vizinhos mais próximos de um robô i qualquer podem ser descritos na forma $\mathcal{N}_i^0 = \mathcal{N}_i$, assim como os vizinhos dos vizinhos de i : $\mathcal{N}_i^1 = \{j \in \mathcal{N}_i^0; q \in \mathcal{V}; q \notin \mathcal{N}_i^0 : (j, q) \in \mathcal{E}\}$. De uma forma genérica, a vizinhança a h -hop para um robô i pode ser descrita como:

$$\mathcal{N}_i^h = \{j \in \mathcal{N}_i^{h-1}; q \in \mathcal{V} \setminus \{\mathcal{N}_i^0 \cup \mathcal{N}_i^1 \cup \dots \cup \mathcal{N}_i^{h-1}\} : (j, q) \in \mathcal{E}\}$$

onde o número sobrescrito representado por h indica a distância em *hops* do escopo da informação, enquanto que o número subscrito representado por i é o índice do robô a que se refere essa vizinhança e a

operação $(\cdot) \setminus (\cdot)$ indica o complemento do conjunto à direita em relação ao da esquerda.

A dinâmica de atualização da matriz de adjacência de i assume, portanto, a seguinte forma:

$$A_i^{(j)}[k] = A_j^{(j)}[k - 1] \quad \forall j \in \mathcal{N}_i^h \quad (3.2)$$

onde $A_i^{(j)} \in \mathbb{R}^{1 \times n}$ representa a linha j da matriz de adjacência de i , levando-se em consideração que, para uma matriz de adjacência qualquer, $a_{ij} > 0$ indica que o robô i recebe informação de j .

Como a propagação da informação requer um menor número de robôs para ocorrer, tem-se uma estrutura da informação mais sólida e coerente em relação ao estado da rede para o escopo local, isso permite uma atualização mais rápida do estado real da rede em cada robô, mesmo em redes com grandes quantidades de nodos ou falhas de comunicação constantes.

3.2. Minimização da Topologia

Segundo Santi [27], em uma rede de sensores ou robôs a maior parte do consumo de bateria se dá pelas ações envolvendo o sistema de comunicação (e.g. enviar e receber mensagens). Existem outras ações, como processamento de informações e movimentação (no caso de veículos), que também consomem bastante energia, no entanto, alterar aspectos dessas atividades pode influenciar negativamente a realização das tarefas relativas a cada nó da rede. Desta forma, seria interessante utilizar alguma estratégia que permita minimizar o consumo de bateria causado pelo processo de comunicação sem prejudicar a atuação dos robôs ou sensores na realização de suas tarefas. Além disso, a interferência causada pelos robôs uns aos outros devido a sobreposição de transmissões é um dos principais motivos de instabilidades e perda de comunicação em redes desse tipo.

Devido a isso, diversos trabalhos têm abordado estratégias de minimização da comunicação em sistemas multi-robôs e redes de sensores sem fio, visando, além da economia de energia, a redução da interferência entre os robôs da rede. Algumas estratégias, como as apresentadas em Casteigts et al. [10], utilizam algoritmos de minimização de grafos para reduzir o número de *links* entre os nodos da rede e minimizar a topologia de comunicação entre eles. O trabalho de Halldórsson e Tokuyama [12] utiliza a minimização do raio de comunicação dos robôs para reduzir a interferência causada pelas transmissões simultâneas,

garantindo o estado de conectividade da rede. Em Bukhart et al. [6], a relação entre a maximização da conectividade e o aumento da interferência na comunicação é relatada e um controle de topologia ótimo, que garante a comunicação e reduz a interferência, é proposto. No trabalho de Johansson e Carr-Motyčková [28], apresenta-se uma nova técnica para determinar a interferência causada pela sobreposição de áreas de transmissão de dados e um controle de topologia que minimiza os efeitos dessa sobreposição. Finalmente, em Hassan e Abuhaiba [11], um controle de topologia para a minimização do consumo de energia e da interferência é proposto. Este algoritmo utiliza a informação de ociosidade dos nós para desligá-los, quando possível, criando uma rede esparsa e reduzindo a interferência de comunicação. Todas essas estratégias baseiam-se na utilização de algoritmos que procuram minimizar o custo dos vértices da rede e resultam em uma árvore de expansão mínima (*MST* do inglês *Minimum Spanning Tree*), o que as diferencia é a forma como a minimização é aplicada para reduzir a interferência ou o consumo de energia e as métricas utilizadas.

Na literatura, há dois tipos de abordagens para o processo de minimização da topologia em um grupo de robôs móveis: abordagem centralizada com informação global e abordagem descentralizada com informação local. A segunda abordagem também pode utilizar informação global, no entanto, isso é menos usual e neste trabalho não será considerado. A abordagem adotada nos algoritmos propostos neste trabalho é descentralizada e com escopo de informação local, embora a abordagem centralizada com informação global seja utilizada como fator comparativo em determinados momentos.

A seguir é apresentado o algoritmo de minimização da topologia de comunicação em um grupo de robôs móveis, que utiliza otimização linear inteira para gerar uma *MST* que será utilizada como base para a redução da interferência e consumo de energia pelos robôs. Logo após, um algoritmo para controle da conectividade em redes dinâmicas é proposto, de forma que este garanta a conectividade e a minimização da rede.

3.2.1. Minimização da Topologia Utilizando Programação Linear

Diversos algoritmos são conhecidos na literatura para a minimização de grafos e geração de árvores de caminho mínimo em tais grafos. Os principais são o algoritmo de **Kruskal** e o algoritmo de **Prim**, originalmente desenvolvidos para lidar com a informação global

sobre o estado da rede [29]. Existem trabalhos onde se altera parcialmente esses algoritmos, permitindo que estes funcionem com um escopo reduzido de informação a respeito do estado da rede [30, 31].

Para a solução do problema de se encontrar a árvore mínima, num contexto local, pode-se desenvolver um algoritmo localizado como o proposto em Li et al. [30], onde se utiliza uma variação do algoritmo de Kruskal (originalmente centralizado) para gerar uma *LMST* (*Localized Minimum Spanning Tree*) com base na informação da vizinhança no escopo de *1-hop*. Como trata-se de uma abordagem local, caso haja ciclos envolvendo pelo menos $2h + 4$ robôs, estes não serão removidos da *LMST*, no entanto, mesmo com *links* redundantes originalmente não encontrados na *MST* global, $MST \subseteq LMST$, o que torna essa solução aceitável para um contexto não muito restritivo.

Neste trabalho optou-se por utilizar uma formulação baseada em programação linear inteira para construir um algoritmo de minimização similar ao algoritmo de Prim. Tal formulação segue a ideia proposta em Dantzig et al. [32] com algumas diferenças nas restrições. As equações (3.3a)-(3.3d) descrevem a função objetivo e as restrições do problema de otimização proposto.

$$\min_{\mathcal{A}} \sum_{i,j \in \mathcal{V}} w_{ij} \alpha_{ij} \quad (3.3a)$$

$$\text{s.t.} \sum_{i,j \in \mathcal{V}} \alpha_{ij} = n - 1 \quad i \neq j \quad (3.3b)$$

$$\sum_{i,j \in \mathcal{S}} \alpha_{ij} \leq |\mathcal{S}| - 1 \quad \forall \mathcal{S} \subseteq \mathcal{V} \quad (3.3c)$$

$$\alpha_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{V} \quad (3.3d)$$

onde α_{ij} é uma variável de decisão que indica a presença de ligação entre os nós i e j , $|\mathcal{S}|$ indica o tamanho do conjunto \mathcal{S} , $\mathcal{A} = [\alpha_{ij}] \in \mathbb{R}^{n \times n} \quad \forall i, j \in \mathcal{V}$ é a matriz de adjacências resultante da solução do problema de otimização e $w_{ij} = \|\tilde{d}_{ij}\|_2$ é a norma euclidiana estimada entre i e j .

A equação (3.3b) representa a restrição que obriga o número de *links* na rede ser igual ao número de robôs menos um, desta forma o grafo resultante do algoritmo de Prim será uma árvore geradora, ou seja, terá o número mínimo de arestas para se manter conexo. Já a equação (3.3c) indica a remoção de ciclos no grafo, esta restrição é interessante porque é a mais influente no desempenho da otimização. No geral, como é necessário avaliar todos os subconjuntos \mathcal{S} contidos em \mathcal{V} , tem-se um número exponencial de restrições em relação ao número

de nós, o que pode tornar o problema demasiadamente difícil de se resolver, ou mesmo infactível dependendo da quantidade de nós na rede. Diversas alternativas são encontradas na literatura para contornar esse problema, as mais conhecidas são a formulação MTZ [33] e as formulações mistas baseadas em corte [34] que conseguem um número polinomial de restrições, no entanto, são mais fracas e suscetíveis a valores não ótimos em determinados casos. Neste trabalho optou-se pela utilização dessa restrição porque, além de sua confiabilidade ser maior, a quantidade de robôs utilizados no processo de minimização é restrito ao escopo local de *1-hop*, reduzindo drasticamente o custo computacional envolvido em comparação com uma abordagem global, por exemplo.

A informação do robô i em relação a seus vizinhos mais próximos (\mathcal{N}_i) é sempre adiantada em um instante de tempo, pois o processo de comunicação e a ação de movimento não ocorrem simultaneamente, o que implica que, para vizinhos a escopo *h-hop*, a informação de i estará adiantada em $h + 1$ instantes de tempo. Com isso a distância vetorial entre i e j é calculada a partir da posição estimada do robô j e da posição do robô i no instante $k - (h + 1)$, como segue:

$$\tilde{d}_{ij}[k] = \tilde{x}_j[k] - x_i[k - (h + 1)],$$

o robô i não conhece a posição de j para o instante k , então estima-se que $\tilde{x}_j[k] = x_j[0]$.

Quando a informação sobre o estado da rede é global, a *MST* gerada pela função (3.3a) é ótima, no entanto é bastante custosa com a restrição (3.3c). Já quando o estado da informação da rede é obtido de forma local, a partir da informação de *h-hop* vizinhos, tem-se uma *MST* não ótima (pois pode conter ciclos), mas com baixo custo computacional, sendo mais eficiente em grupos muito grandes de robôs, onde manter o conhecimento global da rede é praticamente impossível.

Na Figura 3.2 pode-se observar a diferença prática para uma rede com 10 nós e 15 arestas onde existe um laço envolvendo 6 nós. Nesse caso a informação local é de *1-hop*, ou seja, um laço com 6 nós ou mais não é percebido pelo algoritmo de otimização. A aresta tracejada em vermelho indica uma ligação que deveria ser removida, pois não faz parte da *MST* e, embora a *LMST* não seja ótima, percebe-se que *MST* ótima é um subconjunto da *LMST*.

Com a execução do algoritmo de minimização, a matriz de adjacência lógica¹ resultante ($\mathcal{A} = [\alpha_{ij}]$) conterá uma *MST* no escopo global ou uma *LMST* restrita ao escopo de *1-hop* de informação. Essa matriz

¹A matriz de adjacência lógica pode representar ligações virtuais, que não são possíveis fisicamente, mas são definidas entre dois nodos.

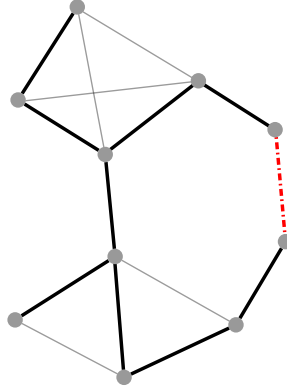


Figura 3.2: *MST* gerada para escopo de informação global (linhas pretas) e local a *1-hop* (linhas pretas e linha tracejada), as linhas em cinza indicam as arestas iniciais do grafo.

será um subconjunto da matriz de adjacência da rede, de forma que $\mathcal{A} \subseteq A$ e com isso, nenhuma ligação poderá ser criada além das que já existiam inicialmente, haverá apenas uma “seleção” das melhores ligações a fim de minimizar esta topologia.

3.2.2. Minimização do Consumo Energético e da Interferência

Uma vez obtida a *MST* ou *LMST*, cada robô pode ajustar o seu raio de comunicação para cobrir apenas a maior distância com outro vizinho da árvore geradora mínima e, com isso, diminuir o consumo de energia utilizado pelo dispositivo de transmissão de dados. Além disso, essa ação reduzirá grande parte da interferência de comunicação causada por sobreposição de áreas de transmissão diferentes. Essa ação para um robô i é descrita através da seguinte equação:

$$r_i^{com} = \max(w_{ij}\alpha_{ij} + \delta_{ij}^{com}) \quad \forall j \in \{1, \dots, n\},$$

onde $\delta_{ij}^{com} \geq 0$ é uma folga utilizada para garantir a conectividade da rede diante da incerteza de posição dos vizinhos de i .

Segundo Santi [27], a energia necessária para transmitir uma mensagem é proporcional ao quadrado da distância que se deseja atingir. Então, reduzir o raio de comunicação de um robô ao seu tamanho ótimo, minimiza consideravelmente o consumo de energia oriundo do

sistema de comunicação. Além disso, de acordo com Johansson e Carr-Motyčková [28], a interferência de comunicação causada por um robô é diretamente proporcional ao número de robôs existentes no raio de comunicação desse robô, logo a redução do raio de comunicação também permite a minimização da interferência na rede.

A ação de reduzir o raio de comunicação dos robôs pode minimizar a topologia no contexto físico, entretanto, em redes onde os robôs têm áreas de cobertura que devem ser monitoradas, por exemplo, essa minimização pode não ocorrer de forma adequada, pois geralmente adota-se uma área de cobertura como sendo $r^{cov} = r^{com}/2$, o que impossibilita qualquer redução no raio de comunicação. Se os robôs forem móveis, pode-se utilizar um algoritmo de movimento que faça com que os robôs não vizinhos na *MST* lógica se afastem, eliminando a ligação física que, outrora, havia entre eles. Na próxima seção, como parte da contribuição deste trabalho, é proposto um algoritmo desse tipo que utiliza consenso e controle preditivo para ajustar a conectividade de maneira descentralizada.

3.2.3. Minimização de *Links* em Redes Dinâmicas

Em redes estáticas, cuja topologia de comunicação não varia no tempo, a manutenção da conectividade entre os agentes durante o processo de minimização é trivial, pois as únicas situações que podem originar perda desta conectividade são falhas nos dispositivos de comunicação ou nos próprios nodos. Entretanto, quando a rede de comunicação é formada por robôs móveis, tem-se uma topologia dinâmica, onde incertezas no deslocamento dos robôs podem quebrar ligações e desconectar o grupo. Então é preciso que o controle de movimento dos robôs se adapte a essa situação, permitindo que a topologia seja minimizada sem perdas de comunicação entre eles.

Com base no algoritmo de Ordoñez et al. [17] apresentado na seção 2.3, foi desenvolvido um algoritmo de controle de conectividade capaz de manter os robôs de um sistema multi-robô conectados enquanto a topologia de comunicação é minimizada. Basicamente, o funcional (2.3) é reformulado para que a ação de controle afaste os robôs de acordo com a área de cobertura de cada um, mantendo-os conectados,

o que resulta na seguinte equação:

$$J_i = \sum_{k=1}^p \sum_{j=1}^n \left(\|x_i[k] - x_{ij}^d\|_{a_{ij}}^2 + \|v_i[k] - v_j[k]\|_{\alpha_{ij}}^2 \right) + \sum_{k=1}^p \|\Delta v_i[k]\|_{\gamma_i}^2, \quad (3.4)$$

onde $\alpha_{ij} \geq 0$ é um entrada da matriz de adjacência lógica resultante do algoritmo de minimização, $a_{ij} \geq 0$ é uma entrada da matriz de adjacência física e x_{ij}^d é a posição a ser alocada para o robô i de acordo com a soma dos raios de cobertura dele e de seu vizinho j .

O primeiro termo dessa nova formulação é bastante similar ao primeiro termo do funcional 2.3, no entanto, nesse caso o robô i deve ocupar uma posição relativa ao seu raio de cobertura e de seus vizinhos. A equação a seguir define como essa posição é obtida por um robô i a partir da posição do seu vizinho j e seus respectivos raios de cobertura:

$$x_{ij}^d = (1 - d_{ij}^m)x_i + d_{ij}^m \tilde{x}_j, \quad (3.5)$$

onde d_{ij}^m é definido conforme a relação de vizinhança entre i e j , da seguinte forma:

$$d_{ij}^m = \begin{cases} [w_{ij} - (r_i^{cov} + r_j^{cov})]/w_{ij} & \text{if } \alpha_{ij} > 0 \\ [w_{ij} - (r_j^{com} + \delta_{ij}^{com})]/w_{ij} & \text{if } \alpha_{ij} = 0 \wedge a_{ij} > 0 \end{cases}$$

e $\delta_{ij}^{com} > 0$ é uma folga suficiente para que os robôs não vizinhos lógicos², mas que estão fisicamente ao alcance um do outro ($\alpha_{ij} = 0 \wedge a_{ij} > 0$) se afastem até perderem a ligação física, minimizando o número de *links* na rede e maximizando a área de cobertura desses robôs.

A Figura 3.3 indica como a posição é alocada para um robô i de acordo com o seu raio de cobertura e o raio de seu vizinho lógico j .

Ja a Figura 3.4 indica como a posição é alocada para um robô i em relação a um vizinho físico j que não é seu vizinho lógico, ou seja, que não tem ligação com i segundo a MST.

Esse termo permite que os robôs ocupem uma posição relativa ao vizinho definida conforme o raio de cobertura de cada um. Isso, garante que no estado final da rede, após a convergência de todos os robôs, não haverá nenhuma violação de posição, ou seja, todos os robôs ocuparão as posições planejadas inicialmente.

²Robôs não vizinhos na MST.

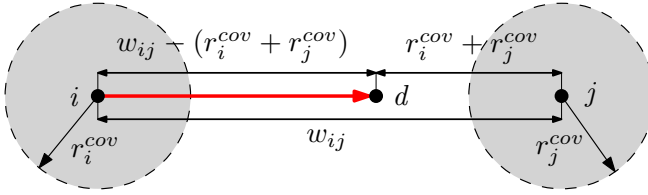


Figura 3.3: Alocação de posição entre dois robôs que são vizinhos na MST.

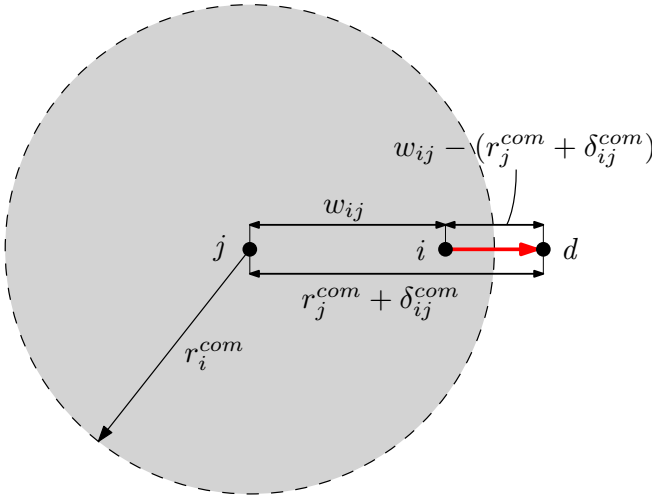


Figura 3.4: Alocação de posição entre dois robôs não vizinhos na MST.

O segundo termo do funcional é utilizado para nivelar as velocidades dos robôs vizinhos, de forma que a diferença de velocidades entre eles se torne a menor possível. Quando se deseja um comportamento grupal normalizado dentro de um sistema multi-robô, é comum a utilização de leis de controle que penalizem a variação da velocidade entre os robôs, criando um “sincronismo” no comportamento destes, e isto é o que se deseja ao adicionar tal termo na função objetivo descrita anteriormente. O terceiro termo é equivalente ao terceiro termo da formulação de Ordoñez et al.

O funcional 3.4 por si só não garante a manutenção da conectividade após a minimização da topologia, pois durante a movimentação

dos robôs, podem ocorrer instabilidades que façam com que vizinhos saiam do raio de comunicação um do outro. Devido a isso, é inserido, no algoritmo de controle de conectividade, uma restrição que limita a ação do controle de forma a garantir que os raios de comunicação de seus vizinhos não sejam violados. Tal restrição é dada como segue:

$$w_{ij} \leq \underbrace{r_j^{com} - \delta_{ij}^{max}}_{r_{ij}^{max}} \quad (3.6)$$

onde $\delta_{ij}^{max} > 0$ é uma folga eventualmente necessária para garantir a não violação das restrições a longo prazo.

Também não há garantias de não colisão entre os robôs durante a alocação de posição, pois apesar do segundo termo do funcional 3.4 permitir que cada robô ocupe uma posição singular ao final da execução do algoritmo, durante a movimentação destes não há restrições que impeçam isso. Por isso, uma restrição para evitar colisões entre os robôs foi formulada a partir dos raios de cobertura de cada um, resultando em:

$$w_{ij} \geq \underbrace{r_i^{cov} + r_j^{cov} + \delta_{ij}^{min}}_{r_{ij}^{min}} \quad (3.7)$$

onde $\delta_{ij}^{min} > 0$ é uma folga utilizada para evitar a violação da restrição.

Como o algoritmo é formulado como uma função de programação quadrática, as restrições devem ser lineares para que o problema seja convexo [24] e, portanto, não se pode simplesmente adicionar as restrições na forma como apresentado na equação (3.6) e na equação (3.7), pois apresentam comportamento não linear devido ao uso do termo w_{ij} , que representa a distância Euclidiana entre os robôs. No trabalho de Correia [35] essas restrições são reescritas considerando a projeção vetorial³, para que possam assumir uma forma linear, resultando em:

$$\begin{aligned} \frac{\Delta k}{\|\tilde{d}_{ij}\|} [\tilde{d}_{ij}^x \quad \tilde{d}_{ij}^y] \begin{bmatrix} v_i^x \\ v_i^y \end{bmatrix} + r_{ij}^{max} - w_{ij} &\geq 0, \\ -\frac{\Delta k}{\|\tilde{d}_{ij}\|} [\tilde{d}_{ij}^x \quad \tilde{d}_{ij}^y] \begin{bmatrix} v_i^x \\ v_i^y \end{bmatrix} + w_{ij} - r_{ij}^{min} &\geq 0, \end{aligned}$$

onde \tilde{d}_{ij}^x e \tilde{d}_{ij}^y são as distâncias vetoriais estimadas para as coordenadas x e y , respectivamente e v_i^x e v_i^y são as velocidades vetoriais para tais coordenadas no primeiro instante de predição.

³A projeção de vetor \vec{u} em direção a um vetor \vec{v} é dada por: $\text{proj}_{\vec{v}}\vec{u} = \vec{u} \frac{\vec{v}}{\|\vec{v}\|}$.

Pode-se escrever a equação (3.4) de forma matricial, seguindo a mesma ideia apresentada para o algoritmo de Ordoñez et al., resultando em:

$$J_i = \sum_{j=1}^n \left(\|X_{i,0} + TV_i - X_{d,0}\|_{a_{ij}}^2 + \|T'V_i - V_{j,0}\|_{\alpha_{ij}}^2 \right) + \|T'V_i - V_{i,0}\|_{\gamma_i}^2,$$

onde $X_{d,0}$ é a matriz de posições alocadas para o instante inicial, definida como:

$$X_{d,0} = \begin{bmatrix} x_{ij}^d[0] \\ x_{ij}^d[0] \\ \vdots \\ x_{ij}^d[0] \end{bmatrix} = (1 - d_{ij}^m) \begin{bmatrix} x_i[0] \\ x_i[0] \\ \vdots \\ x_i[0] \end{bmatrix} + d_{ij}^m \begin{bmatrix} x_j[0] \\ x_j[0] \\ \vdots \\ x_j[0] \end{bmatrix}.$$

Finalmente, isolando-se V_i e escrevendo o funcional na forma quadrática tem-se:

$$J_i = \frac{1}{2} V_i^\top H_i V_i + f_i V_i,$$

com

$$H_i = \sum_{\substack{j=1 \\ j \neq i}}^n \left(T^\top a_{ij} T + T'^\top \alpha_{ij} T' \right) + T'^\top \gamma_i T'$$

e

$$f_i = \sum_{\substack{j=1 \\ j \neq i}}^n \left((X_{i,0}^\top - X_{d,0}^\top) a_{ij} T - V_{j,0}^\top \alpha_{ij} T' \right) - V_{i,0}^\top \gamma_i T'.$$

Essa formulação final deve ser utilizada para calcular a velocidade em uma única coordenada, no entanto, como a restrição de conectividade, apresentada na equação (3.6), relaciona-se com mais de uma coordenada, é necessário formular o algoritmo final para todas as coordenadas envolvidas. No caso do espaço ser um plano, a equação resultante será igual a equação (2.4), com a adição das restrições de conectividade e anticolisão.

3.2.4. Controle de Adição de *Links*

Para evitar a criação de *links* desnecessários utiliza-se uma regra de adição de arestas baseada na informação de distância entre os

vizinhos de um robô e o novo candidato à adição de aresta. A Figura 3.5 indica as duas situações possíveis para a adição de novas ligações entre os robôs: na Figura 3.5a pode-se observar o caso onde a menor distância entre um nó candidato j e um nó q qualquer, que seja vizinho de i , é maior que a distância entre esse robô candidato e o robô i ; já na Figura 3.5b é descrita a situação onde a menor distância entre um robô i e um vizinho q qualquer é maior que a distância entre esse robô i e o robô candidato à adição j .

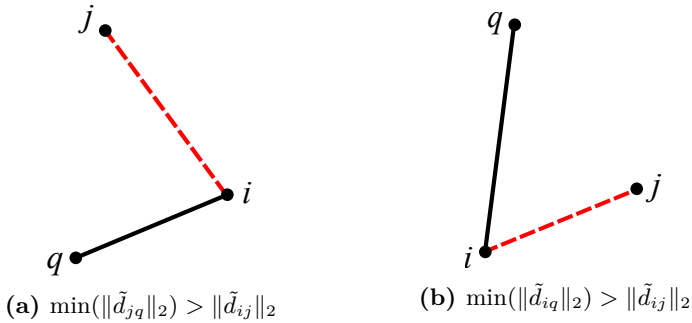


Figura 3.5: Situações para adição de novos *links*.

Com a implementação dessa regra para a adição de novos vizinhos para um robô i , obriga-se que todas as arestas adicionais sejam melhores (tenham menor distância) do que as que já existem, permitindo uma adaptação dinâmica às variações de posição de cada robô da rede. A equação a seguir descreve essa regra em função da matriz de adjacência do robô i .

$$A_i(i, j) = (\min(\|\tilde{d}_{jq}\|_2) > \|\tilde{d}_{ij}\|_2) \vee (\min(\|\tilde{d}_{iq}\|_2) > \|\tilde{d}_{ij}\|_2),$$

onde $q \in \mathcal{N}_i$ e $a_{ij} > 0$ indica que o robô i recebe informação do robô j .

Com a utilização dessa regra, reduz-se a necessidade da minimização da topologia em cada ciclo de comunicação, pois esta só apresentará variação na sua estrutura quando o novo *link* for melhor que um já existente.

De forma genérica, o controle de topologia para a minimização do número de *links* em uma rede de robôs móveis pode ser descrito pelo algoritmo 1. A operação $(\cdot) \leftarrow (\cdot)$ é uma ação de inserção do elemento à direita no conjunto à esquerda, o pacote de comunicação é definido como $msg = [A, x, v, r^{com}, r^{cov}]$ para todos os robôs, as funções **envia**(\cdot) e **recebe**(\cdot) são operações de comunicação, a função

Algoritmo 1 Minimização da topologia para um robô i

```

1: envia( $msg_i$ )
2: for  $j = 1$  to  $n$  do
3:   recebe( $msg_j$ )
4:   if  $msg_j \neq NULL$  then
5:      $\mathcal{N}_i \leftarrow j$ 
6:      $A(i, j) = 1$ 
7:      $\mathcal{N}_i^1 \leftarrow \{\forall q \in A_j^{(j)} | q \notin \mathcal{N}_i \wedge q \neq 0\} \cup \mathcal{N}_i$ 
8:   end if
9: end for
10:  $\bar{A}_i^1 = \text{adjacencia}(\mathcal{N}_i^1)$ 
11:  $\mathcal{A}_i = \text{prim}(\bar{A}_i^1)$ 
12:  $\forall j \in \mathcal{N}_i : r_i^{com} = \max(w_{ij}\alpha_{ij} + \delta_{ij}^{com})$ 
13:  $v_i = \text{CC}(\mathcal{A}_i)$ 

```

adjacencia(\cdot) retorna uma matriz de adjacência ponderada (\bar{A}_i) entre os vértices do conjunto utilizado como entrada, a função **prim**(\cdot) é o algoritmo de minimização de *links* descrito pela equação 3.3a e a função **CC**(\cdot) é o controle de conectividade para a minimização da topologia definido na equação 3.4.

O diagrama esquemático relacionando o controle de topologia para a minimização da comunicação e o controle de movimento de um robô i qualquer é dado conforme a Figura 3.6, onde COM indica o processo de comunicação entre os robôs, PRIM indica o algoritmo de minimização do grafo que representa a rede apresentado na formulação (3.3a), CC é o controle de conectividade baseado no funcional (3.4), CM é o controle de movimento que pode ser interpretado como o controle de baixo nível adequado à dinâmica do robô controlado (R_i) e as variáveis v_i e θ_i representam, respectivamente, a velocidade linear e angular a ser aplicada no robô i oriundas do controle de baixo nível.

Como se observa, o controle de topologia para a minimização da comunicação é, na verdade, o conjunto formado pelo processo de comunicação, algoritmo de minimização e controle de conectividade, sendo cada um deles essenciais para que se possa minimizar o número de *links* em uma rede de robôs móveis.

O controle de conectividade se comporta como um controle de “alto nível”, não dependente da dinâmica comportamental do sistema a ser controlado. Com isso, pode-se utilizar o controle de topologia com diversos tipos de robôs de forma transparente, sem que seja necessário o conhecimento sobre o comportamento específico que o robô apresenta.

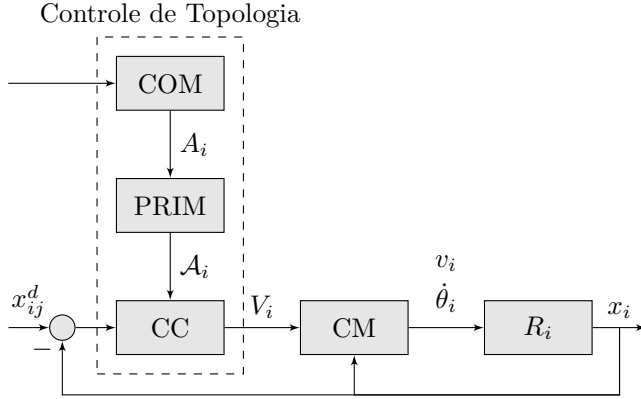


Figura 3.6: Relação entre o controle de topologia para a minimização da comunicação e o controle de movimento de um robô.

A desvantagem deste método está, justamente, na necessidade de se utilizar uma camada extra de controle entre o controle de conectividade e o robô. Para os testes realizados na seção 4.4.1 com robôs comerciais, o controle de “baixo nível” utilizado é apresentado no apêndice B, enquanto que nos demais testes o controle de “baixo nível” não é aplicado.

A seguir é apresentado um exemplo de funcionamento do algoritmo de minimização em uma topologia de comunicação arbitrária.

3.2.5. Exemplo

Para demonstrar o funcionamento do controle de topologia para a minimização da comunicação, tal algoritmo é aplicado a uma rede de robôs móveis arbitrária com topologia variante no tempo e é demonstrado como ocorre a minimização da topologia inicial e a manutenção da conectividade dessa rede.

A rede é constituída por 6 robôs em um espaço bidimensional. Suas posições iniciais, raios de cobertura e raios de comunicação são definidos conforme a Tabela 3.1 e todos os valores são dados em metros.

Inicialmente, todos os robôs estão parados e possuem o mesmo raio de comunicação e raio de cobertura, constituindo uma rede homogênea.

Os parâmetros do controle de conectividade são definidos como

segue:

$$\begin{aligned}
 p &= 5 \quad \Delta k = 0.5s \quad \gamma_i = 0.5 \\
 \alpha_{ij} &\in \{0, 3, \dots, 9\} \\
 a_{ij} &\in \{0, 3, \dots, 10\}
 \end{aligned}$$

A restrição de saturação para a velocidade resultante do controle de conectividade para as coordenadas x e y é definida como:

$$v^{max} = [1 \quad 1] \quad v^{min} = [-1 \quad -1],$$

ambas as velocidade são em m/s .

A topologia em seu estado inicial é representada pela Figura 3.7. Como se pode observar, existem diversos *links* redundantes na rede que reduzem a cobertura total dos robôs e causam um desperdício dos recursos de comunicação.

O escopo da informação é de *1-hop*, então todos os robôs da rede se conhecem. Isso implica que a árvore geradora mínima resultante da execução do algoritmo de minimização da topologia por cada um dos robôs será ótima e não conterá laços.

Após algumas iterações do controle de topologia, uma árvore geradora mínima é construída e os robôs reduzem o seu raio de comunicação e se movimentam para otimizar a sua posição em relação à do vizinho, resultando na topologia apresentada na Figura 3.8. Como se observa, os robôs 2, 4 e 6 tiveram seus raios de comunicação reduzidos significativamente, minimizando o consumo de energia e a interferência na rede causada pela sobreposição de áreas de transmissão.

Na Figura 3.9, o grupo de robôs atinge sua configuração final, com todos os robôs ocupando a posição alocada em relação ao seu raio de cobertura e o dos vizinhos. Na Tabela 3.2, tem-se os valores finais

Tabela 3.1: Configuração inicial dos robôs.

robô	x	y	r^{com}	r^{cov}
1	11.5	14.6	10	3
2	4.6	18.4	10	3
3	7.5	1.8	10	3
4	6.3	12.3	10	3
5	14.4	6.7	10	3
6	14.9	15.2	10	3

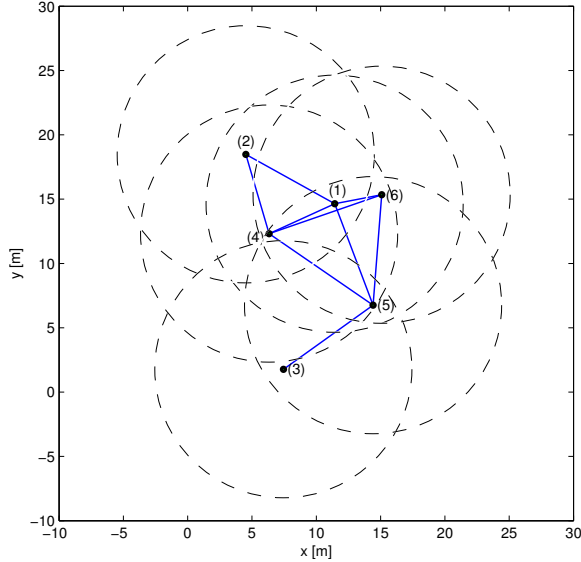


Figura 3.7: Topologia de comunicação no estado inicial: círculos em torno dos robôs indica a área de comunicação de cada um.

para a posição, raio de cobertura e raio de comunicação ao final da simulação.

Tabela 3.2: Configuração final dos robôs.

robô	x	y	r^{com}	r^{cov}
1	11.45	13.45	6.6057	3
2	3.96	18.09	6.6004	3
3	8.86	3.5	6.6016	3
4	5.56	12.3	6.6004	3
5	13.15	7.69	6.6057	3
6	16.93	15.89	6.5994	3

No geral, o raio de comunicação ficou pouco menor do que o apresentado no estágio inicial da topologia, pois o raio de cobertura é bastante grande, ou seja, houve pouca efetividade para a redução do consumo de energia. Entretanto, com relação à redução da interferência, os ganhos foram bem mais significantes, já que apenas os vizinhos da MST permanecem no raio de comunicação uns dos outros.

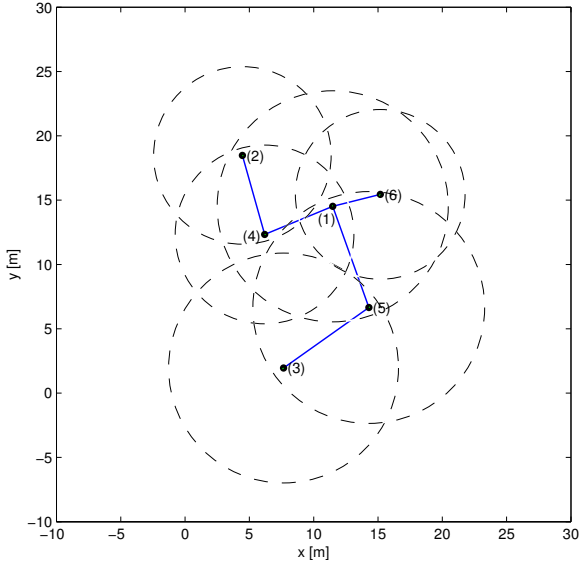


Figura 3.8: Topologia de comunicação após algumas iterações do controle de topologia.

3.3. Manutenção da Conectividade

Nesta seção, o controle de topologia é abordado no contexto de manutenção da conectividade em grupos de robôs móveis, através da criação e manutenção de topologias tolerantes a falhas e instabilidades nos dispositivos de comunicação desses robôs.

Existem na literatura, diversas estratégias para se construir topologias tolerantes aos mais variados tipos de falha de comunicação. Quando se trata de controle de topologia para a manutenção da conectividade, é comum a utilização de uma propriedade conhecida como κ -conectividade (oriunda da teoria de grafos) para construir redes, topologicamente, tolerantes as falhas de comunicação. Nos trabalhos de Li et al. [7] e Ahmadi e Stone [8] utiliza-se a troca de mensagens em um grupo de robôs, para, de forma descentralizada, determinar se uma rede é ou não bi-conexa, estabelecendo critérios de sincronização entre esses robôs para manter a bi-conectividade mesmo em caso de perda nós da rede. Já a solução proposta por Wagenpfeil et al. [9] utiliza o conceito de forças de repulsão e atração para construir uma lei de controle descentralizada que permita movimentar determinados robôs para

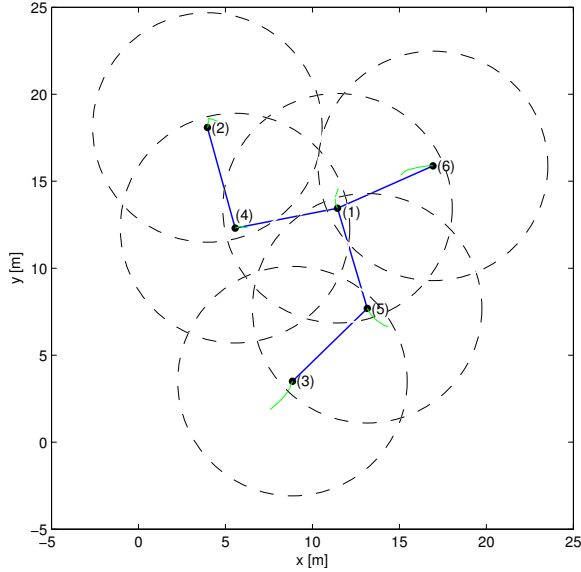


Figura 3.9: Topologia de comunicação em seu estágio final, após a execução do controle de topologia.

o estabelecimento da bi-conectividade. Outro trabalho similar é o de Casteigts et al. [16] em que são utilizadas forças virtuais para se aproximar robôs vizinhos e construir uma estrutura triangular fundamental, que garante a bi-conectividade de forma descentralizada. No trabalho de Fiacchini e Morarescu [36] são apresentadas algumas condições fundamentais para que determinadas topologias sejam preservadas mesmo sob influência de desconexões e falhas na comunicação em sistemas inter-conectados.

Um grafo é dito κ -conexo se para qualquer par de nós u e v existem pelo menos κ caminhos distintos entre eles (p.e. a árvore de extensão mínima de um grafo qualquer é 1 -conexa). Em teoria dos grafos esta propriedade é apresentada formalmente no Teorema de Menger, proposto pelo matemático austríaco Karl Menger em 1927 [37] como *vértice-conectividade* (κ_v).

Teorema 1 (Vértice-conectividade) *Seja \mathcal{G} um grafo formado por um conjunto de arestas definido por \mathcal{E} e um conjunto de vértices definido por \mathcal{V} . Suponha que B e C são subconjuntos de \mathcal{V} e suponha que exista pelo menos um caminho entre B e C . Então o número mínimo de vértices que separam B de C é igual ao número máximo de caminhos dis-*

tintos entre B e C .

Além da κ -conectividade, o teorema de Menger também apresenta a *aresta-conectividade* (κ_e), em que pelo menos κ_e arestas podem ser removidas de um determinado grafo sem que este apresente desconexão.

Teorema 2 (Aresta-conectividade) *Seja \mathcal{G} um grafo não direcionado e x e y dois vértices não adjacentes pertencentes a esse grafo. Então o número mínimo de arestas cuja remoção pode desconectar x e y é igual ao número máximo de caminhos distintos entre x e y .*

A *vértice-conectividade* insere redundância na comunicação de uma rede qualquer quando $\kappa_v > 1$, o que implica na manutenção da conectividade mesmo se até $\kappa_v - 1$ nodos forem perdidos. Logo, o controle de topologia para a manutenção da conectividade se resume em criar uma topologia κ -conexa e garantir que essa topologia não seja alterada no decorrer do tempo, mesmo durante atividades que exijam movimentação ou quando houverem falhas na comunicação.

Para criar uma topologia minimamente tolerante a falhas (*bi-conexa*), deve-se tratar vértices cuja remoção cause desconexão ao grafo, os chamados nós críticos ou vértices de corte. Nas próximas seções são apresentadas, como parte da contribuição deste trabalho, duas técnicas para tratamento de nós críticos: a primeira implícita, baseada no Problema do Caixeiro Viajante; e a segunda explícita, baseada na detecção dos nós críticos por meio da conectividade algébrica do grafo e na criação de *links* virtuais entre vizinhos destes vértices. Por fim, é proposto um algoritmo de controle de conectividade baseado em consenso, para o estabelecimento e manutenção da bi-conectividade em redes dinâmicas.

3.3.1. Correção de Nós Críticos Utilizando o Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (em inglês, *Travelling Salesman Problem - TSP*) é um problema NP-difícil clássico de otimização combinatória, onde um viajante precisa percorrer o menor caminho entre todas as cidades de um determinado local, passando somente uma vez em cada cidade e retornando para o início após atingir a última cidade. Na literatura existem diversos algoritmos para a resolução desse problema, no entanto, a maioria deles possui ordem de complexidade exponencial [38, 39].

O caminho percorrido pelo caixeiro viajante é, na verdade, um ciclo hamiltoniano e este tipo de estrutura é a forma mínima bi-conexa que um grafo pode atingir. Tendo isso em mente, pode-se construir uma

topologia bi-conexa mínima em um grafo completo por aplicar nele um algoritmo para resolução do Problema do Caixeiro Viajante. Para um grafo arbitrário essa ideia também pode ser utilizada, no entanto, é preciso calcular as distâncias entre todos os seus vértices construindo um grafo completo e, só então, aplicar o algoritmo de resolução para este problema.

Neste trabalho será utilizado uma solução para o Problema do Caixeiro Viajante apresentada por Miller et al. [33] que possui ordem de complexidade $\mathcal{O}(n^2)$ e é conhecida como Formulação MTZ. A relaxação efetuada nas restrições originais do problema permitem que a resposta não ótima seja atingida em determinadas situações, mas para cenários menos restritivos como o deste trabalho, esta é uma boa solução. As equações (3.8a)-(3.8d) representam a Formulação MTZ e suas restrições.

$$\min_{\mathcal{A}} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \alpha_{ij} \quad (3.8a)$$

$$\text{s.t.} \quad \sum_{i=1, i \neq j}^n \alpha_{ij} = 1 \quad \forall j \in \mathcal{V} \quad (3.8b)$$

$$\sum_{j=1, j \neq i}^n \alpha_{ij} = 1 \quad \forall i \in \mathcal{V} \quad (3.8c)$$

$$u_i - u_j + n\alpha_{ij} \leq n - 1 \quad 2 \leq i \neq j \leq n \quad (3.8d)$$

onde u_i e u_j são variáveis inteiras de valor singular aleatório.

É interessante notar que a restrição (3.8d) é equivalente à restrição (3.3c), sendo ambas utilizadas para a eliminação de ciclos no grafo. No entanto, a primeira restrição apresentada possui ordem de complexidade exponencial, o que a torna extremamente custosa, enquanto a segunda tem ordem quadrática. É possível, na formulação (3.3a) substituir essa restrição, no entanto, deve-se estar atento as limitações da nova formulação com respeito à ordem em que os nodos aparecem, pois isso pode tornar a solução não ótima.

Para gerar uma topologia bi-conexa a partir de um grafo arbitrário \mathcal{G} inicialmente não bi-conexo, a variável w_{ij} deve representar as distâncias entre todos os nodos desse grafo, então assume-se que $\mathcal{W} = [w_{ij}] \in \mathbb{R}^{n \times n}$ é uma matriz de adjacência ponderada para o grafo completo $\bar{\mathcal{G}}$ construído a partir do grafo inicial. Uma vez que o algoritmo de minimização é executado, a matriz resultante $\mathcal{A} = [\alpha_{ij}]_{n \times n}$ será formada apenas por um ciclo hamiltoniano ótimo envolvendo todos os robôs do grupo. Note-se, nesse caso, que \mathcal{A} é uma matriz de

adjacência lógica, ou seja, nem todas as ligações descritas nelas são possíveis fisicamente (e.g. dois nós marcados como vizinhos em \mathcal{A} podem estar fora do alcance um do outro) e é o controle de conectividade que fará com que esses vizinhos lógicos tornem-se físicos.

Em um contexto descentralizado, a informação global não pode ser obtida de forma satisfatória, então a matriz \mathcal{W} deve ser reduzida a um escopo de informação plausível para um contexto localizado e distribuído de processamento. Dessa forma, \mathcal{W}_i representa a matriz de adjacência ponderada para o grafo completo formado pelos nodos que são vizinhos de i no escopo de 1-hop . Como resultado dessa alteração, a matriz lógica resultante \mathcal{A} conterá o ciclo hamiltoniano localizado, restrito aos robôs da vizinhança a 1-hop de i .

A Figura 3.10 representa a atuação do algoritmo (3.8a) de forma centralizada e descentralizada em um grafo com sete nodos numa configuração inicialmente não bi-conexa (os nodos 3, 5 e 6 são críticos).

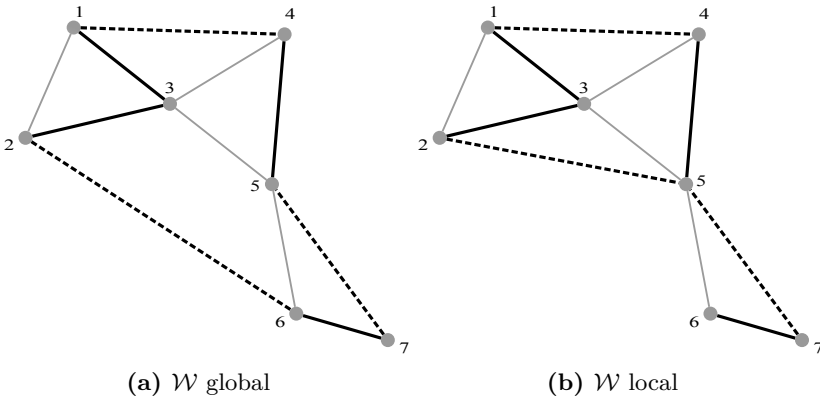


Figura 3.10: Diferenças de atuação do algoritmo (3.8a) no caso de conhecimento global e no caso de conhecimento local.

Como se observa na Figura 3.10a, o grafo resultante da execução do algoritmo no contexto global é um ciclo hamiltoniano ótimo (linhas pretas), contendo ligações lógicas⁴ (marcadas como linhas pretas tracejadas). Já na Figura 3.10b, o algoritmo (3.8a) é executado para o caso onde o conhecimento a respeito do estado da rede é local, res-

⁴Ligações lógicas ou *links* virtuais são arestas que relacionam dois vértices cuja distância um do outro é maior do que ambos os seus respectivos raios de comunicação.

trito a *1-hop*. Há algumas semelhanças entre os resultados das duas execuções, entretanto, no segundo caso não existe a “eliminação” de todos os nós críticos no primeiro instante e possivelmente isso ocorrerá apenas quando pelo menos um dos nós críticos forem corrigidos. Além disso, algumas ligações são direcionadas (entre os nós 2 e 5), o que indica que a matriz \mathcal{A} não é a mesma em todos os robôs.

Uma vez obtida a matriz de adjacência lógica contendo a configuração topológica mínima capaz de tornar a rede bi-conexa, cada robô atualiza sua matriz de adjacência lógica da seguinte maneira:

$$\mathcal{A}_i = A_i \vee \mathcal{A}.$$

Essa matriz de adjacência localizada será a entrada do algoritmo de controle de conectividade, que se incumbirá de aproximar os robôs relacionados através dos *links* virtuais, tornando a rede bi-conexa.

3.3.2. Correção de Nós Críticos Utilizando Ligações Virtuais

Essa abordagem segue alguns conceitos apresentados na anterior, no entanto, nesse caso usa-se somente a informação de nó crítico para tornar a rede bi-conexa. No trabalho de Das et al. [40] é proposto um algoritmo muito similar ao que será discutido nessa sessão, basicamente nele os nós críticos coordenam a aproximação de seus vizinhos mais próximos que não são vizinhos entre si para que um caminho alternativo possa ser criado e o nó deixe de ser crítico. A diferença é relativa à forma como o nó crítico coordena essa aproximação: no trabalho de Das et al. o nó crítico utiliza mensagens de controle de movimento e posição para informar aos robôs vizinhos em que direção se mover, enquanto que no caso discutido neste trabalho é utilizado o conceito de ligação virtual que permite ao algoritmo de consenso cooperativo aproximar robôs não vizinhos a ponto de eles tornarem-se vizinhos.

Essa abordagem pode ser realizada de maneira descentralizada, no entanto, para a detecção ótima dos nós críticos é necessária a informação global, o que não é viável, então uma alternativa baseada em informação local é utilizada. Essa técnica funciona e apesar do conjunto ótimo de nós críticos ser subconjunto do conjunto local de nós críticos, podem existir os chamados “falsos positivos”, que são nós não críticos classificados como críticos. Isso não representa um problema, propriamente dito, no entanto, acarretará em um grafo com valência

média⁵ maior do que o apresentado numa abordagem centralizada.

A Figura 3.11 exhibe os nós críticos detectados sob o escopo de informação à *1-hop*. Os nós em azul indicam nós não críticos que foram marcados como críticos localmente (nós 3, 4, 5, 7 e 9) e os nós em vermelho são os nós críticos reais (nós 6 e 8).

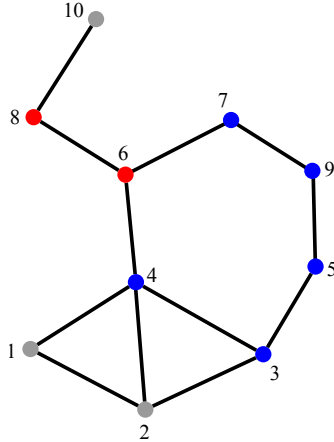


Figura 3.11: Nós críticos detectados sob escopo de informação local.

Na abordagem descentralizada, cada robô avalia sua própria criticidade, utilizando sua informação sobre a topologia no escopo de *1-hop*, então retira a si mesmo da matriz adjacência que representa esse escopo e calcula o segundo menor autovalor do espectro Laplaciano dessa matriz, dado por $\lambda_2(\mathcal{L})$, que indica a conectividade algébrica do grafo resultante sem a sua presença [41]. Ao avaliar o valor de $\lambda_2(\mathcal{L})$, o robô verifica sua criticidade, pois se $\lambda_2(\mathcal{L}) > 0$, a rede no contexto local sem a sua presença é conexa, implicando na sua não criticidade, e desconexa caso $\lambda_2(\mathcal{L}) \leq 0$, indicando sua criticidade.

Uma vez que um robô i detecta a si mesmo como nó crítico, ele deve verificar quais são seus vizinhos, que não sejam vizinhos entre si e estejam à menor distância um do outro, para criar uma ligação virtual entre eles. Essa ação é efetuada através do seguinte problema de otimização, resultando em um par de vértices $(j, q) \notin \mathcal{E}$ a serem

⁵A valência média de um grafo é a média do número de arestas incidentes para com os vértices deste grafo.

marcados como parte de um *link* virtual:

$$\begin{aligned}
 & \min_{j,q} \|\tilde{d}_{ij}\|_2 + \|\tilde{d}_{iq}\|_2 & (3.9) \\
 & \text{s.t. } j \neq q \\
 & \quad q \notin \mathcal{N}_j \\
 & \quad j \notin \mathcal{N}_q \\
 & \quad \forall j, q \in \mathcal{N}_i
 \end{aligned}$$

É interessante notar que a informação utilizada para eleger os componentes do *link* virtual a partir de um nó crítico é restrita apenas à vizinhança mais próxima desse nó e que podem ser acessados diretamente por ele. Isso é importante, pois garante um maior controle sobre o estado da informação da topologia no contexto local.

Quando os componentes do *link* virtual são eleitos, cabe ao nó crítico notificá-los sobre a sua nova vizinhança virtual e isso é feito através de uma mensagem de controle definida como **add**(j, q), onde j e q são os membros do *link* virtual. O algoritmo 2 detalha como é

Algoritmo 2 Criação de *links* virtuais para o robô i

```

1:  $A = \mathbf{adjacencia}(\mathcal{N}_i^1)$ 
2:  $\mathcal{A}(t) = M^{(i)} \vee \mathcal{A}(t-1)$ 
3:  $\mathcal{L} = \mathit{diag}(\sum_{j=1}^n \alpha_{ij}) - \mathcal{A}$ 
4: if  $\lambda_2(\mathcal{L}) \leq 0$  then
5:    $\forall j, q \in \mathcal{N}_i : w_{jq} = \min_{j,q} \|\tilde{d}_{ij}\|_2 + \|\tilde{d}_{iq}\|_2$ 
6:    $\mathcal{A}(j, q) = \mathcal{A}(q, j) = 1$ 
7:   add( $j, q$ )
8: end if

```

o processo de detecção e tratamento de um nó crítico dada a topologia inicial. Neste caso, a função **adjacencia**(\cdot) retorna uma matriz de adjacência comum, a partir do conjunto utilizado como entrada. A matriz $M^{(i)}$ é uma submatriz resultante da eliminação da linha e coluna i da matriz de adjacência do robô i . Na linha 5, para todos os vizinhos de i , deve-se encontrar os candidatos a vizinhos lógicos através da equação (3.9) e, então, marcar na matriz de adjacência do robô i a relação virtual entre esses candidatos, o que ocorre na linha 6.

A Figura 3.12 representa o processo de detecção de um nó crítico na rede e criação do *link* virtual para corrigir a sua criticidade. O robô 3 detecta sua criticidade, e procura entre seus vizinhos, aqueles que estão mais próximos e ainda não são vizinhos, satisfazendo a equação (3.9).

Os robôs 1 e 4 são escolhidos como candidatos ao *link* virtual por ainda não serem vizinhos e estarem mais próximos um do outro que os demais robôs não vizinhos. O robô 3 os marca como vizinhos em sua matriz de adjacência lógica e envia a mensagem **add**(1,4) para todos os robôs⁶. Quando os robôs 1 e 4 a recebem, verificam que estão marcados como componentes de uma ligação virtual, criando em suas matrizes de adjacência lógica, tal ligação. Finalmente a ação de controle de conectividade os aproxima até a distância entre eles permitir a comunicação direta de um com o outro e o *link* virtual entre eles tornar-se real. A linha tracejada representa uma ligação virtual, as setas em azul são a ação de comunicação e as setas em vermelho indicam a ação de controle necessária para criar a vizinhança física entre os robôs 1 e 4.

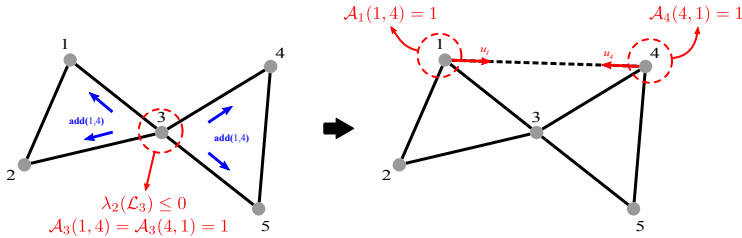


Figura 3.12: O robô 3 detecta sua criticidade e cria uma ligação virtual entre os robôs 1 e 4 (linha tracejada).

Uma vez os *links* virtuais criados, os nós críticos e os demais nós que participam como componentes dessas ligações devem monitorar o comportamento da topologia para remover ou substituir ligações virtuais que por ventura deixem de ser úteis para o processo de estruturação da bi-conectividade na rede. Essa manutenção dos *links* virtuais ocorre em quatro situações distintas, de acordo com o estado em que se encontra a topologia: quando uma possibilidade de *link* virtual melhor surge; quando o nó crítico que mantém um determinado *link* virtual deixa de ser crítico; quando um dos membros de um *link* virtual perde a comunicação com o nó crítico associado à essa ligação; e quando o nó crítico perde a comunicação com os membros do *link* virtual a ele associado. A seguir cada situação e as ações necessárias em cada robô para manter a bi-conectividade lógica da rede são detalhadas.

Muitas vezes um *link* virtual associado a um nó deixa de ser a melhor opção quando outros robôs não vizinhos apresentam-se a uma

⁶Considera-se uma rede de comunicação *Ad-hoc*.

distância menor que a desta ligação. Nesse caso, um novo *link* virtual deve ser criado e o antigo removido. A Figura 3.13 indica como ocorre esse processo: no primeiro instante, o robô 3 cria uma ligação virtual entre os robôs 1 e 4, por serem os não vizinhos entre si pertencentes a vizinhança de 3 que apresentam a menor distância um do outro. No próximo instante, os robôs 2 e 5 se aproximam e a distância entre eles torna-se menor que a distância entre 1 e 4, então o robô 3 cria um novo *link* virtual entre os robôs 2 e 5, enviando a mensagem **add**(2, 5) para os robôs e remove o *link* virtual entre 1 e 4, enviando a mensagem **del**(1, 4) para todos os robôs. Quando os robôs 1 e 4 recebem a mensagem **del**(1, 4), estes removem de sua matriz de adjacência lógica, a ligação com o parceiro virtual e o *link* entre 2 e 5, passa a ser a única ligação virtual da rede. Com isso, a topologia da rede adapta-se de maneira dinâmica às posições dos robôs, permitindo uma construção sempre ótima dos *links* virtuais.

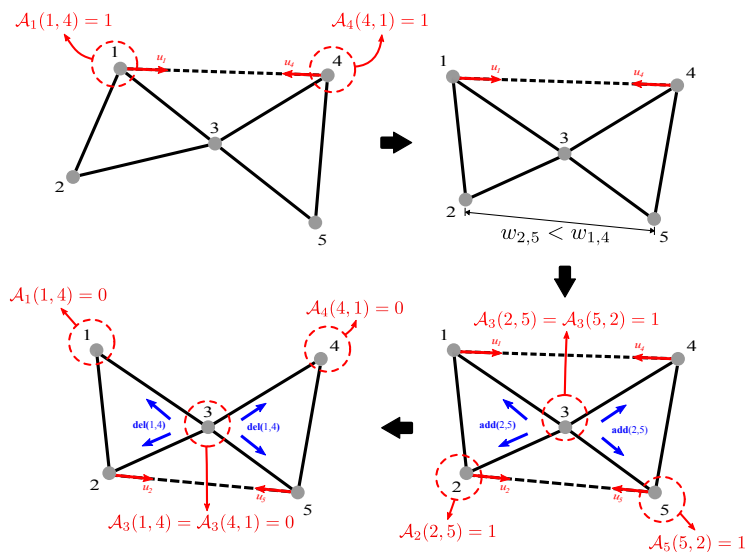


Figura 3.13: Substituição e remoção de *link* virtual quando uma ligação melhor é detectada.

Quando um nó crítico é corrigido, alguns *links* virtuais podem não se tornar físicos, e neste caso estes devem ser removidos para evitar o consumo desnecessário de recursos. Assim como no caso anterior, o gerenciamento da ligação virtual é realizado pelo nó crítico ao qual pertence esta ligação. A Figura 3.14 representa o procedimento de

detecção e remoção de um *link* virtual desnecessário associado a um nó anteriormente crítico: o robô 1 detecta sua criticidade e então informa os robôs 5 e 2 que estes fazem parte de uma ligação virtual para corrigir essa criticidade. Durante a movimentação dos robôs, surge uma ligação física entre os robôs 2 e 3, o que elimina a criticidade do robô 1. Como o escopo de informação é de *1-hop*, o robô 1 percebe que o robô 3 e o robô 2 são vizinhos, então dispara uma mensagem **del**(5, 2) que irá remover a ligação virtual desnecessária entre os robôs 2 e 5.

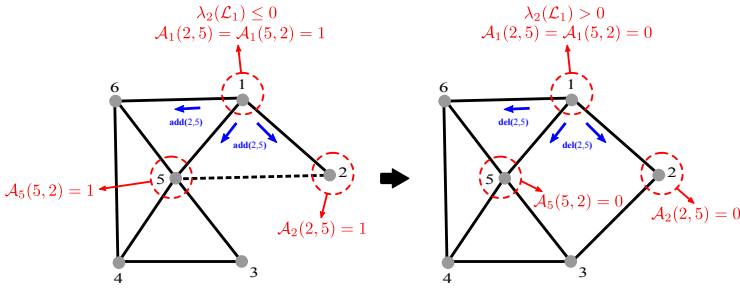


Figura 3.14: Remoção de *link* virtual associado a um nó anteriormente crítico.

Outro estado em que pode haver remoção de *links* virtuais desnecessários é quando um dos membros de uma ligação virtual perde a comunicação com o nó crítico associado à esta ligação. Neste caso, manter a ligação virtual não só é inútil, como também consome recursos da rede, pois o membro restante do *link* virtual continuará se movimentando até atingir a última posição armazenada pelo membro que perdeu a comunicação, na esperança de estabelecer a ligação física pretendida com a criação do *link* virtual. A Figura 3.15 mostra como ocorre o processo de notificação do membro restante e a remoção da ligação virtual pelo robô que é nó crítico. O nó 1 perde comunicação com o robô 3 e com o robô 2 o que é percebido por ambos quando o tempo de resposta do robô 1 (τ_1) excede o *timeout* previamente ajustado para a rede. Então o robô 3 remove a ligação virtual entre os robôs 1 e 4, emitindo uma mensagem **del**(1, 4) para os robôs, e quando o robô 4 recebe essa mensagem, remove de sua matriz de adjacência lógica a ligação virtual com o robô 1. Na iteração seguinte, o robô 3 percebe que tornou-se novamente crítico e uma ligação virtual entre os robôs 2 e 5 é criada, corrigindo a sua criticidade do ponto de vista lógico.

A última situação em que ocorre a remoção de *links* virtuais no

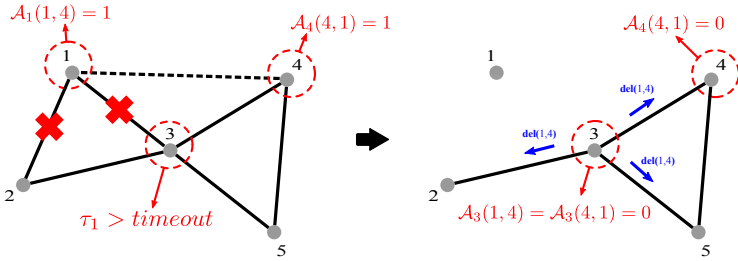


Figura 3.15: Remoção de *link* virtual após um de seus integrantes perder comunicação com o nó crítico associado.

sistema multi-robô é definida quando o nó crítico perde a comunicação com os robôs pertencentes às suas ligações virtuais. Nesse caso, em especial, de acordo com quais robôs perderem comunicação com o nó crítico, pode haver uma desconexão da rede. Ao perderem conexão com o nó crítico, os nodos componentes da ligação virtual deixam de conhecer a posição um do outro e, devido a isso, não poderão estabelecer entre si a ligação física que corrigiria a criticidade do nó crítico. Na Figura 3.16 observa-se os instantes em que há a perda de comunicação entre os robôs componentes do *link* virtual e o nó crítico a ele associado. Assim que as conexões entre os robôs 1 e 3, e 3 e 4 são perdidas, tais robôs eliminam quaisquer ligações virtuais de que façam parte e tenham como nó crítico um dos robôs cujo tempo de resposta excedeu o *timeout* ajustado para a rede. Com isso, os robôs 1 e 4 eliminam a ligação virtual entre eles e, de mesma forma, o robô 3 remove a ligação virtual entre 1 e 4 de sua matriz de adjacência lógica, tornando-se novamente crítico.

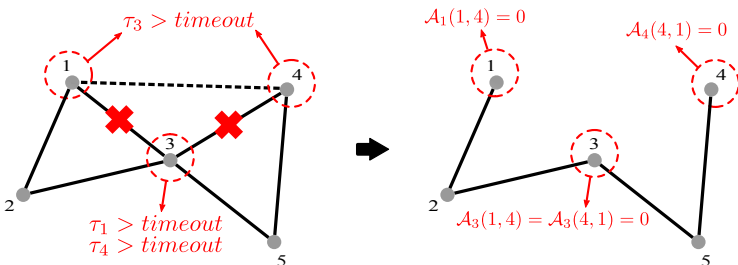


Figura 3.16: Remoção de *link* virtual após o nó crítico a ele associado perder a comunicação com seus membros.

Esse método possui um custo computacional extremamente re-

duzido, se comparado à abordagem anterior, no entanto, a sua principal desvantagem em relação a primeira, está na necessidade de trocas de mensagens especiais entre os nós críticos e os vizinhos candidatos à ligação virtual, pois em caso de falhas de comunicação que prejudiquem a dispersão dessas mensagens, a ligação virtual pode não ser estabelecida e o nó crítico não ser corrigido no tempo correto. Porém, se as falhas forem temporárias, haverá apenas um pequeno atraso na correção do nós críticos, e o estabelecimento da bi-conectividade na rede não será prejudicado.

3.3.3. Controle de Conectividade Utilizando Consenso

Uma vez a criticidade dos nós resolvida no nível lógico, resta aplicar uma ação de controle cooperativo que permita aproximar os robôs que estão fora do alcance um do outro afim de estabelecer a bi-conectividade na rede. Essa ação é provida pelo algoritmo de conectividade baseado na proposta de Ordoñez et al. que será apresentado nessa seção.

A ação de controle em cada robô, deve fazê-lo buscar a posição alocada em relação a seus vizinhos, ao mesmo tempo que atrai os robôs que estão fora do alcance de comunicação um do outro. Tendo estes requisitos como base, o funcional (2.3) é reescrito, resultando em:

$$J_i[k] = \sum_{j=1}^n \left(\|x_i[k] - x_{ij}^d\|_{\alpha_{ij}}^2 + \|v_i[k] + v_j[k]\|_{\bar{\alpha}_{ij}}^2 \right) + \|\Delta v_i[k]\|_{\gamma_i}^2 \quad i = 1, \dots, n, \quad (3.10)$$

onde α_{ij} é uma entrada de \mathcal{A} , e $\bar{\alpha}_{ij} > 0$ se e somente se $r_i^{com} < w_{ij} > r_j^{com} \wedge \alpha_{ij} > 0$, e x_{ij}^d é a posição alocada para o robô i de acordo com o seu raio de cobertura e de seu vizinho j , como descrito na equação (3.5).

O primeiro termo dessa equação é similar ao primeiro termo da equação (3.4), porém, neste caso, o valor de d_{ij}^m na equação (3.5), assume apenas o valor para quando $\alpha_{ij} > 0$. O segundo termo impõe uma penalização para velocidades no mesmo sentido, se os robôs estiverem em movimento. Esse termo é aplicado somente à robôs cuja ligação com o robô i seja virtual, ou seja, que estejam fora do raio de comunicação um do outro. Isto garante que os vizinhos virtuais não se afastem (desde que o primeiro termo seja implementado), pois obriga-os a ter velocidades em sentidos opostos.

Assumindo dinâmica de primeira ordem para os robôs, pode-se reescrever o funcional (3.10) de forma matricial, englobando todos os instantes de predição. Dessa forma tem-se:

$$J_i = \sum_{j=1}^n \left(\|X_{i,0} + TV_i - X_{d,0}\|_{\alpha_{ij}}^2 + \|T'V_i + V_{j,0}\|_{\bar{\alpha}_{ij}}^2 \right) + \|T'V_i - V_{i,0}\|_{\gamma_i}^2,$$

isolando-se V_i e escrevendo J como uma função quadrática na forma $\frac{1}{2}x^\top Qx + c^\top x$, obtém-se:

$$J_i = \frac{1}{2}V_i^\top H_i V_i + f_i V_i,$$

onde

$$H_i = \sum_{\substack{j=1 \\ j \neq i}}^n \left(T^\top \alpha_{ij} T + T'^\top \bar{\alpha}_{ij} T' \right) + T'^\top \gamma_i T'$$

e

$$f_i = \sum_{\substack{j=1 \\ j \neq i}}^n \left((X_{i,0}^\top - X_{d,0}^\top) \alpha_{ij} T + V_{j,0}^\top \bar{\alpha}_{ij} T' \right) - V_{i,0}^\top \gamma_i T'.$$

Para um espaço de coordenadas bidimensional, o algoritmo final de otimização quadrática apresenta-se com a mesma estrutura que o algoritmo mostrado na equação (2.4), inclusive com as mesmas restrições de saturação do sinal de controle.

De maneira genérica, a cada iteração do controle de topologia para a manutenção da conectividade, cada robô executa uma sequência de ações, como descrito no Algoritmo 3. A função **distancias**(\cdot) calcula a norma euclidiana para todos os robôs inseridos como entrada e retorna uma matriz com estas distâncias e as funções **CNC**(\cdot) e **CC**(\cdot) são o algoritmo de correção de nós críticos e o controle de conectividade descrito pela equação 3.10, respectivamente.

A relação entre o controle de topologia para a manutenção da conectividade e o controle de movimento de um robô i é apresentada no diagrama da Figura 3.17, que é similar ao apresentado na Figura 3.6. Entretanto, neste caso, o bloco PRIM é substituído pelo bloco CNC que representa a correção de nós críticos utilizando-se *links* virtuais ou a solução para o Problema do Caixeiro Viajante. Os demais estágios de controle são similares, com exceção do bloco CC, que indica o controle de conectividade adaptado à presença de *links* virtuais, descrito pelo funcional (3.10).

Algoritmo 3 Manutenção da conectividade para o robô i

```

1: envia( $msg_i$ )
2: for  $j = 1$  to  $n$  do
3:   recebe( $msg_j$ )
4:   if  $msg_j \neq NULL$  then
5:      $\mathcal{N}_i \leftarrow j$ 
6:      $A(i, j) = 1$ 
7:      $\mathcal{N}_i^1 = \{\forall q \in A_j^{(j)} | q \notin \mathcal{N}_i \wedge q \neq 0\} \cup \mathcal{N}_i$ 
8:   end if
9: end for
10:  $\mathcal{W}_i = \text{distancias}(\mathcal{N}_i^1)$ 
11:  $\mathcal{A}_i = \text{CNC}(\mathcal{W}_i)$ 
12:  $v_i = \text{CC}(\mathcal{A}_i)$ 

```

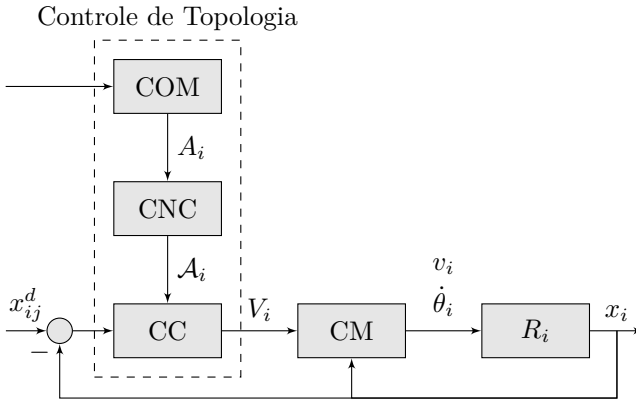


Figura 3.17: Relação entre o controle de topologia para a manutenção da conectividade e o controle de movimento de um robô.

Na próxima seção, alguns exemplos ilustrativos de utilização do controle de topologia para manutenção da conectividade em redes dinâmicas de robôs móveis são apresentados.

3.3.4. Exemplos

Para auxiliar o entendimento dos algoritmos apresentados, são propostos dois exemplos demonstrando como ocorre o processo de construção da topologia bi-conexa em uma rede arbitrária. No primeiro

exemplo, a ação de correção de nós críticos é baseada no Problema do Caixeiro Viajante e no segundo exemplo utiliza-se a detecção de nós críticos explícita e sua correção por meio de *links* virtuais.

A topologia inicial utilizada em ambos os exemplos é a mesma, composta por 6 robôs não holonômicos dispostos em um espaço bidimensional. As posições iniciais e os raios de cobertura e comunicação são definidos, em metros, pela Tabela 3.3.

Tabela 3.3: Configuração inicial de posição, raio de comunicação e raio de cobertura.

robô	x	y	r^{com}	r^{cov}
1	0	0	10	4.5
2	2	7	10	4.5
3	10	0	10	4.5
4	18	7	10	4.5
5	20	0	10	4.5
6	25	5	10	4.5

Inicialmente todos os robôs estão parados e, como pode ser observado na Tabela 3.3, todos possuem o mesmo raio de comunicação e raio de cobertura, resultando em uma topologia homogênea com troca de informação não direcionada.

O controle de conectividade apresenta os seguintes valores para os parâmetros da função objetivo:

$$\begin{aligned}
 p &= 5 \quad \Delta k = 0.5s \quad \gamma_i = 0.5 \\
 \alpha_{ij} &\in \{0, 4, \dots, 10\} \\
 \bar{\alpha}_{ij} &\in \{0, 11, \dots, 20\}
 \end{aligned}$$

A restrição de saturação para o sinal de controle é definida em função da velocidade máxima e mínima (dada em m/s) em cada coordenada para os robôs, da seguinte forma:

$$v^{max} = [1 \quad 1] \quad v^{min} = [-1 \quad -1]$$

A topologia de comunicação inicial é definida pela posição geográfica dos robôs, em função da distância que estes apresentam entre si. A Figura 3.18 indica essa topologia, com as ligações e o raio de comunicação de todos os robôs. As linhas em azul representam comunicação bidirecional e o círculo em torno dos robôs é o raio de comunicação de cada um.

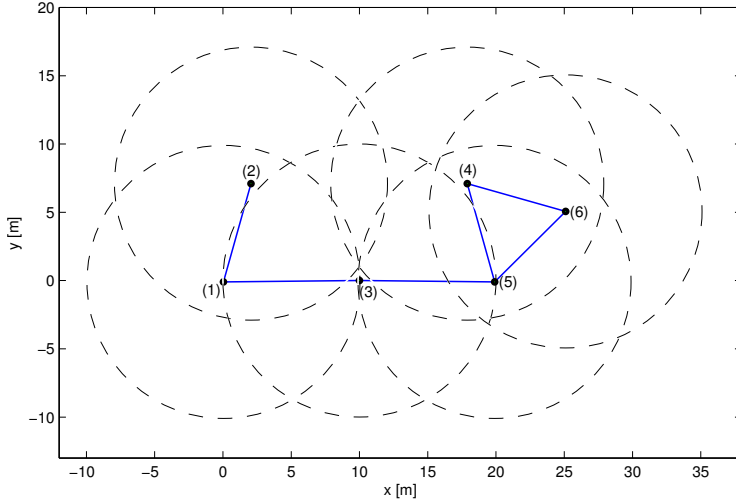


Figura 3.18: Configuração inicial da topologia: os círculos representam a área de comunicação de cada robô.

Inicialmente a topologia apresenta-se numa configuração não bi-conexa com três nós críticos, sendo eles os robôs 1, 3 e 5. A seguir, tais nós críticos são corrigidos utilizando as duas técnicas propostas e o algoritmo de controle de conectividade é aplicado para tornar a rede bi-conexa.

3.3.4.1. Exemplo 1

Neste exemplo o algoritmo de controle de topologia para manutenção da conectividade utiliza uma estratégia baseada no Problema do Caixeiro Viajante para eliminar a criticidade dos nós de corte e tornar a rede logicamente bi-conexa. O algoritmo de controle de conectividade é, então, aplicado ao grupo visando aproximar robôs que sejam apenas vizinhos virtuais uns dos outros.

Como visto na Figura 3.18, os nós 1, 3 e 5 são críticos, e precisa-se corrigi-los para que bi-conectividade seja estabelecida. O escopo de informação é restrito a *1-hop*, o que significa que o robô 2, por exemplo, conhece apenas os robôs 1 e 3. Entretanto, o robô 3, por estar em uma posição privilegiada, conhece todos os robôs, mas suas ações para corrigir a bi-conectividade estão restritas à sua vizinhança mais próxima, o que o impede de estabelecer ligação entre os robôs 2 e

4 para a eliminação da sua criticidade, por exemplo.

Ao executar o algoritmo para resolução do TSP com uma entrada restrita a *1-hop* de informação, cada robô obtém a sua própria versão do ciclo hamiltoniano resultante. Geralmente esse ciclo é diferente para a maioria dos robôs, apenas em determinados casos ou quando o escopo da informação é global, o grafo resultante do algoritmo é igual entre vários robôs.

A Figura 3.19 mostra a topologia de comunicação do ponto de vista de quatro robôs após estes executarem o algoritmo de resolução do TSP com escopo de informação restrito a *1-hop*.

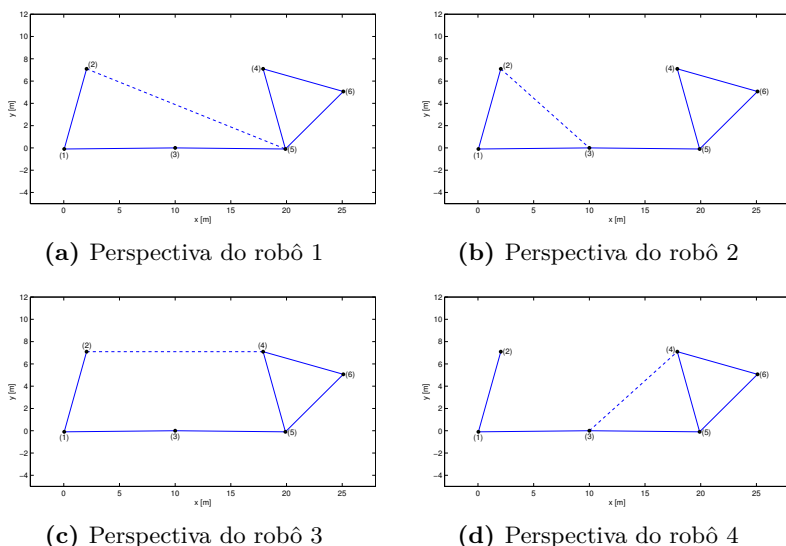


Figura 3.19: Topologia resultante para diferentes perspectivas após a execução do algoritmo de resolução do TSP: linhas tracejadas indicam ligações virtuais.

Da perspectiva do robô 1, com sua vizinhança definida como $\mathcal{N}_1^1 = \{2, 3, 5\}$, quando o algoritmo 3.8a for executado, a matriz de adjacência lógica resultante será igual a:

$$\mathcal{A}_1 = \begin{matrix} & \begin{matrix} (1) & (2) & (3) & (5) \end{matrix} \\ \begin{matrix} (1) \\ (2) \\ (3) \\ (5) \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix},$$

e a topologia de comunicação nessa perspectiva, pode ser observada na Figura 3.19a.

Como o *link* virtual foi criado entre os robôs 2 e 5 (linha tracejada), o robô 1 não pode corrigir a sua criticidade sem enviar mensagens de controle para ambos os robôs, no entanto, como o robô 1 não é vizinho direto do robô 5, não há como garantir, em um ambiente instável e sujeito a falhas na comunicação, que o robô 5 receba a mensagem de controle. Por isso, essa abordagem não faz envios desse tipo de mensagem, visando tornar a estratégia mais robusta a incertezas no processo de comunicação.

Na perspectiva do robô 2, que tem sua vizinhança definida como $\mathcal{N}_2^1 = \{1, 3\}$, a topologia resultante fará com que o robô 2 corrija a criticidade do robô 1. A matriz resultante da execução do algoritmo de resolução do TSP para o robô 2 é dada por:

$$\mathcal{A}_2 = \begin{matrix} & \begin{matrix} (1) & (2) & (3) \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 1 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

e a topologia da rede, de acordo com a perspectiva do robô 2 é dada pela Figura 3.19b.

Na Figura 3.19c, temos a topologia resultante na perspectiva do robô 3 após a execução do algoritmo de correção de nós críticos. Como o robô 3 conhece todos os vizinhos, por estar localizado a *1-hop* de todos eles, a matriz \mathcal{A}_3 resultante do algoritmo para o Problema do Caixeiro Viajante contém uma topologia, em nível lógico, bi-conexa ótima. Entretanto, assim como ocorreu com o robô 1, o robô 3 não pode corrigir a sua criticidade, já que não há envios de mensagens de controle.

Finalmente na Figura 3.19d, o robô 4 executa o algoritmo de correção de nós críticos e obtém uma configuração de topologia que corrige a criticidade do robô 5. Assim como na perspectiva do robô 2, neste caso, o robô tem um *link* virtual, então o algoritmo de controle de conectividade fará com que os robôs 2 e 4 se aproximem do robô 3. Na perspectiva do robô 3, não há *links* virtuais entre ele e qualquer robô, por isso ele não será atraído pelos robôs 2 e 4.

A Figura 3.20 mostra quatro estágios na evolução da topologia no processo de estabelecimento da bi-conectividade. Na Figura 3.20a, percebe-se a presença de ligações virtuais entre os robôs 2 e 3, 4 e 3 (linhas em magenta), no entanto, estas apresentam-se de como ligações

direcionadas, pois para o robô 3, não existe ligação virtual entre eles. A Figura 3.20b indica o exato momento em que uma ligação física bidirecional é estabelecida entre os robôs 2 e 3 e uma ligação física direcionada de 3 para 4 é também estabelecida. Esta ligação direcionada entre 3 e 4 ocorre porque ambos os robôs, nesta iteração, ainda não receberam mensagens um a partir do outro, mas como o robô 4 conhece a posição atrasada de 3, na perspectiva dele o *link* não é mais virtual quando a distância entre eles é menor que o raio de comunicação de 3.

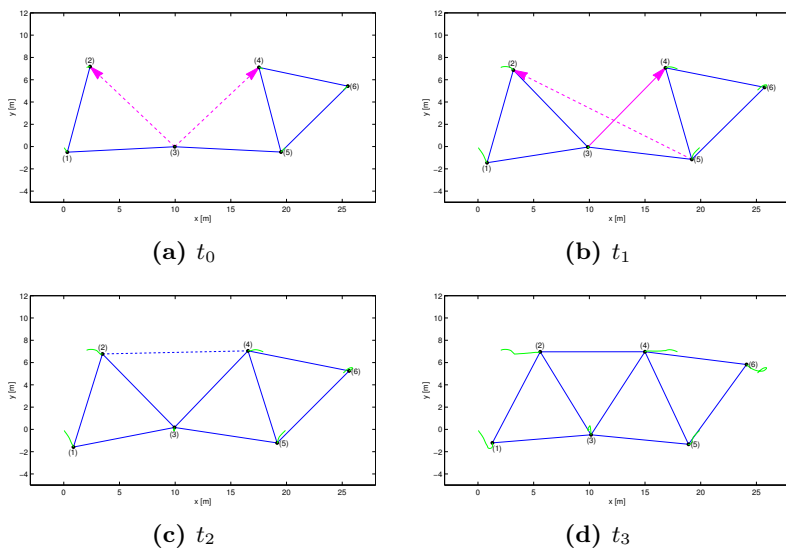


Figura 3.20: Evolução do controle de topologia em quatro momentos distintos: linhas tracejadas com setas indicam ligações virtuais direcionadas.

O robô 2 torna-se vizinho direto de 3, então ele conhece o robô 5, e o novo cálculo do algoritmo de resolução do Problema do Caixeiro Viajante cria uma ligação lógica entre 2 e 5. Entretanto, como observado na Figura 3.20c, quando a ligação entre os robôs 3 e 4 torna-se física, o robô 2 passa a conhecer a posição do robô 4, idem para o robô 4, então o *link* virtual é criado entre os dois, tanto na perspectiva do robô 2 quanto de 4. Finalmente na Figura 3.20d o estado de bi-conectividade é atingido, e todas as ligações entre os robôs tornam-se físicas.

As matrizes componentes do controle de conectividade executado no robô 2, utilizando informações do robô 3 e robô 1 para o instante

descrito pela Figura 3.20a, são definidas como segue:

$$T = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \quad T' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

e os componentes vetoriais são dados como:

$$X_{d_{(2,3)},0} = \begin{bmatrix} 10.9004 & -0.0081 \\ 10.9004 & -0.0081 \\ 10.9004 & -0.0081 \\ 10.9004 & -0.0081 \\ 10.9004 & -0.0081 \end{bmatrix} \quad X_{d_{(2,1)},0} = \begin{bmatrix} 9.0383 & -0.0097 \\ 9.0383 & -0.0097 \\ 9.0383 & -0.0097 \\ 9.0383 & -0.0097 \\ 9.0383 & -0.0097 \end{bmatrix}$$

$$X_{2,0} = \begin{bmatrix} 2 & 7 \\ 2 & 7 \\ 2 & 7 \\ 2 & 7 \\ 2 & 7 \end{bmatrix}$$

para as coordenadas x e y , com $X_{d_{(2,3)},0}$ sendo calculado entre os robôs 2 e 3 e $X_{d_{(2,1)},0}$ sendo calculado entre os robôs 2 e 1. Os vetores iniciais de velocidade são definidos como zero, pois a simulação se inicia com os robôs parados.

O componente matricial H_2 da função objetivo é definido como:

$$H_2 = \begin{bmatrix} 1.10 & -0.42 & 0.06 & 0.04 & 0.02 \\ -0.42 & 1.08 & -0.44 & 0.04 & 0.02 \\ 0.06 & -0.44 & 1.06 & -0.46 & 0.02 \\ 0.04 & 0.04 & -0.46 & 1.04 & -0.48 \\ 0.02 & 0.02 & 0.02 & -0.48 & 0.52 \end{bmatrix}$$

e o componente vetorial f_2 para as coordenadas x e y assume a seguinte forma:

$$f_2^{x,y} = \begin{bmatrix} 0.0307 & 0.0089 \\ 0.0245 & 0.0071 \\ 0.0184 & 0.0053 \\ 0.0123 & 0.0036 \\ 0.0061 & 0.0018 \end{bmatrix}$$

Após uma iteração do algoritmo de controle de conectividade, as velocidades vetoriais ótimas estimadas nas coordenadas x e y para os

5 instantes de predição são:

$$V_2^{x,y} = \begin{bmatrix} -0.0450 & -0.0130 \\ -0.0656 & -0.0190 \\ -0.0722 & -0.0209 \\ -0.0726 & -0.0210 \\ -0.0717 & -0.0208 \end{bmatrix}$$

As velocidades em $V_2^{x,y}$ são sinais de controle ótimos para o robô 2 que garantem a manutenção da conectividade entre ele e o robô 1 e o aproxima de seu vizinho virtual, o robô 3. Apesar da matriz $V_2^{x,y}$ conter os valores para todos os instantes da janela de predição, a posição do robô é atualizada com base apenas no primeiro valor predito para a velocidade nas coordenadas x e y , o que equivale à primeira linha dessa matriz.

3.3.4.2. Exemplo 2

Dada a topologia inicial, anteriormente descrita, utiliza-se o controle explícito de *links* virtuais para corrigir os nós críticos e estabelecer a bi-conectividade lógica dessa topologia. Assim, como no exemplo anterior, o algoritmo de conectividade baseado em consenso constrói a bi-conectividade física na rede de robôs móveis.

Na Figura 3.21 pode-se observar quatro momentos distintos do processo de estruturação da bi-conectividade na topologia inicial, utilizando a correção de nós críticos baseada na construção explícita de *links* virtuais.

Inicialmente, como mostrado na Figura 3.21a, o robô 3 detecta sua criticidade e, através da equação (3.9) encontra os melhores candidatos à ligação virtual que pode corrigir tal criticidade; esses candidatos, no primeiro instante, são os robôs 1 e 5. O mesmo processo ocorre com os robôs 1 e 5, pois ambos são nós críticos: o robô 1 cria uma ligação virtual entre os robôs 2 e 3, e o robô 5 cria uma ligação virtual entre os robôs 3 e 4. Com a aproximação dos robôs, as ligações entre 2 e 3, e 4 e 3 tornam-se físicas, corrigindo a criticidade dos robôs 1 e 5.

Na Figura 3.21b, nota-se que a distância entre os robôs 2 e 4 é menor do que a distância entre os robôs 1 e 5, que é o atual *link* virtual associado ao nó crítico 3. Como os robôs 2 e 4 são ambos vizinhos do robô 3 e não são vizinhos entre si, eles se tornam uma melhor opção como candidatos a uma ligação virtual que substituirá o *link* entre os robôs 1 e 5. Isso ocorre na Figura 3.21c, onde uma ligação virtual entre

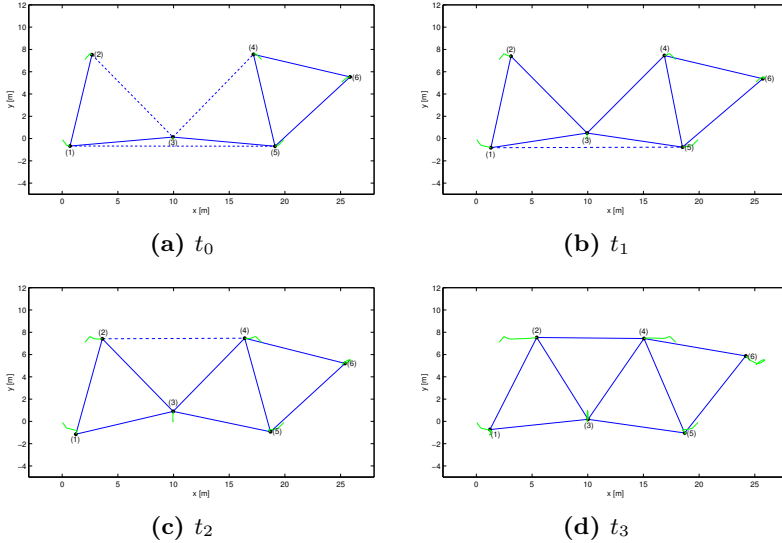


Figura 3.21: Evolução do controle de topologia utilizando *links* virtuais.

os robôs 2 e 4 é criada e a ligação entre os robôs 1 e 5 é removida. Finalmente na Figura 3.21d, o *link* virtual entre os robôs 2 e 5 torna-se físico e a criticidade do robô 3 é corrigida, resultando numa topologia bi-conexa.

Durante a criação e remoção das ligações virtuais, uma série de mensagens de controle é trocada entre os nós críticos e os robôs membros dessas ligações, por exemplo, na Figura 3.21c o robô 3 envia uma mensagem **add**(2, 4) para criar uma ligação virtual entre os robôs 2 e 4, e em seguida envia **del**(1, 5) para eliminar a ligação virtual que fora substituída. Essa é a maior desvantagem desse método para correção dos nós críticos em relação ao anterior, pois se houverem muitas perdas de pacotes devido à instabilidades do ambiente, um robô pertencente à ligação virtual pode não receber uma mensagem para remover esse *link* o que, além de resultar em consumo desnecessário de recursos desse robô, pode até mesmo causar problemas na estruturação da bi-conectividade.

Para a primeira iteração do controle de conectividade executado no robô 2, as velocidades previstas para os 5 instantes de predição são bastante similares às apresentadas no exemplo anterior, pois o estado topológico da rede em relação ao robô 2 é o mesmo nesse período. Já a

partir do tempo descrito na Figura 3.21b os sinais de controle passam a ser diferentes nos dois exemplos, porém mantendo-se equivalentes e mostrando que, no geral, o comportamento do controle de conectividade é o mesmo para as duas estratégias.

Capítulo 4

Resultados e Avaliação

Nesta seção, os algoritmos apresentados anteriormente são submetidos à simulações numéricas e testes em uma plataforma robótica experimental visando a comprovação de seu funcionamento. As simulações são realizadas por uma implementação dos algoritmos em linguagem C++, e utilizam como entrada, um conjunto de grafos aleatórios gerados pela biblioteca NetworkX¹ que é desenvolvida em Python e permite a construção de vários tipos de grafos de uma maneira personalizável. Mais informações a cerca do simulador e do gerador de grafos são encontradas nos apêndices A e D, respectivamente.

Os testes com os robôs reais são efetuados com o auxílio da interface de comunicação para robôs conhecida como ROS (*Robot Operating System*)², que é extremamente poderosa e flexível, permitindo a manipulação dos mais diversos tipos de robôs de uma maneira simples e padronizada.

A análise de robustez é feita através de algumas métricas de desempenho de redes baseadas em conceitos de teoria dos grafos. Os testes serão propostos visando comprovar que as características topológicas oriundas da estratégia utilizada estão entre as melhores possíveis para a configuração inicial da rede apresentada.

A seguir são expostas, de forma breve, as métricas de análise de robustez costumeiramente utilizadas em teoria dos grafos, na sequência são feitas as simulações numéricas e as análises de desempenho e finalmente na seção 4.4 são apresentados os experimentos com os robôs reais, utilizando as plataformas anteriormente mencionadas.

¹<<https://networkx.github.io/>>

²<<http://www.ros.org/>>

4.1. Métricas para Análise de Robustez

A análise de robustez em redes de comunicação visa a determinação de medidas que possam quantificar o desempenho dessas redes de maneira numérica levando em consideração propriedades matematicamente fundamentadas em teoria dos grafos. Essas medidas servem como fatores de comparação qualitativa entre diferentes tipos de redes, para determinar o quanto uma rede é melhor ou pior que outra, por exemplo.

Algumas métricas aqui apresentadas são chamadas de medidas espectrais do grafo de comunicação, pois utilizam os autovalores do espectro Laplaciano derivado da matriz de adjacência que representa o grafo das redes de comunicação. Existem outras métricas de robustez que podem ser utilizadas, no entanto, para o escopo desse trabalho, as apresentadas se comportam de maneira mais adequada às propriedades que se deseja comprovar.

Na sequência são apresentadas as cinco medidas utilizadas nesse trabalho para se avaliar a robustez dos algoritmos de controle de topologia propostos. As três primeiras são espectrométricas e são utilizadas para verificar o desempenho do controle de topologia para a manutenção da conectividade, enquanto que as duas últimas são definidas em função do número de arestas e seu peso relativo e são utilizadas para avaliar a minimização da topologia de comunicação.

4.1.1. Conectividade Algébrica

A matriz Laplaciana de um grafo é positiva semi-definida e a soma de suas linhas é sempre zero, devido a isso seus autovalores são reais, não negativos e o menor deles é zero. Desta forma, pode-se agrupar tais autovalores por ordem de grandeza, tal que $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. O segundo menor autovalor (λ_2) foi chamado por Miroslav Fiedler [41], de Conectividade Algébrica e sua magnitude pode definir o quão conexo é um grafo, da seguinte maneira:

1. A conectividade algébrica de um grafo é igual a zero se e somente se o grafo é desconexo;
2. A conectividade algébrica de um grafo incompleto não é maior que a vértice-conectividade desse grafo, dessa forma:

$$0 \leq \lambda_2 \leq \kappa_v \leq \kappa_e \leq \delta,$$

onde κ_v é a vértice-conectividade, κ_e é a aresta-conectividade e δ é a mínima valência do grafo.

Apesar de a conectividade algébrica detectar características fundamentais de um grafo, ela não é capaz de detectar pequenas variações no número de arestas que não influenciem drasticamente a conectividade desse grafo. Por isso, a sua utilização para análise de robustez só pode ser realizada em conjunto com outras métricas mais precisas, a fim de se ter um resultado mais coerente com a realidade.

4.1.2. Número de Árvores Geradoras

Uma árvore geradora é um subgrafo contendo $n - 1$ arestas, sem ciclos e de acordo com Baras e Hovareshti [42], o número de árvores geradoras de um grafo representando uma rede de comunicação é um indicador de robustez dessa rede. Isso ocorre, porque a confiabilidade de um grafo pode ser mensurada pelo número de caminhos disponíveis entre seus nós constituintes, já que a probabilidade de existir um caminho entre qualquer par de nós é igual a probabilidade de existência de uma árvore geradora no grafo.

Como consequência do Teorema da Matriz-Árvore proposto por Kirchhoff, o número de árvores geradoras de um grafo pode ser definido em função dos $n - 1$ maiores autovalores da matriz Laplaciana não ponderada desse grafo, resultando na seguinte equação:

$$\xi = \frac{1}{n} \prod_{i=2}^n \lambda_i.$$

Diferentemente da conectividade algébrica, o número de árvores geradoras é bastante sensível a mínima variação da conectividade do grafo e acaba se comportando da mesma forma que outra medida de robustez em grafos, a confiabilidade polinomial, aumentando sua magnitude de acordo com a robustez desse grafo.

4.1.3. Resistência Efetiva

A resistência efetiva é definida de acordo com a lei de Kirchhoff para circuitos elétricos assumindo que o grafo se comporta como um circuito elétrico, onde cada aresta (i, j) é equivalente a um resistor $r_{ij} = 1$ Ohm. Assim, como em circuitos elétricos, a resistência efetiva total do grafo pode ser determinada por meio das equações para o cálculo de resistências em paralelo e em série, de forma que se $r_1 = 1$ e $r_2 = 1$, então $r_1 + r_2 = 1 + 1 = 2$ Ohm, se r_1 e r_2 estão conectados em série, e $(r_1^{-1} + r_2^{-1})^{-1} = (1^{-1} + 1^{-1})^{-1} = 1/2$ Ohm, se r_1 e r_2 estão conectados em paralelo.

A resistência efetiva total do grafo é dada pela soma das resistências efetivas de cada aresta. Mas, de acordo com Klein e Randić [43] a resistência efetiva total de um grafo, também pode ser obtida a partir dos autovalores não nulos da matriz Laplaciana desse grafo, da seguinte maneira:

$$R = n \sum_{i=2}^n \frac{1}{\lambda_i}.$$

No trabalho de Ellens et al. [44] é provado que a resistência efetiva de um grafo é estritamente relacionada ao grau de conectividade desse grafo, decrescendo seu valor de acordo com esse grau de conectividade. Desta forma, quanto menor a resistência efetiva de um grafo que representa uma rede de comunicação, mais robusta é esta rede.

4.1.4. Custo Energético

Segundo Santi [27], o consumo energético de uma rede pode ser analisado a partir do seu custo energético, que é definido em função da distância característica associada à rede obtida a partir da soma dos raios de comunicação de todos os robôs elevado a um fator de potência que caracteriza o ambiente onde tais robôs se encontram. A sua magnitude é obtida por meio da seguinte equação:

$$\Gamma = \sum_{i=1}^n (r_i^{com})^\phi,$$

onde ϕ é o fator de atenuação definido de acordo com o grau de interferência do ambiente, para uma área livre de obstáculos, por exemplo, $\phi = 2$.

4.1.5. Cobertura Média das Arestas

No trabalho de Johansson e Carr-Motyčková [28] a interferência oriunda do processo de comunicação na rede é obtida a partir da análise da cobertura média das arestas da rede (*Average Edge Coverage*), definida como:

$$\bar{C} = \sum_{i=1}^n \sum_{j=1}^n a_{ij} \frac{C_{ij}}{n},$$

onde C_{ij} é a cobertura da aresta (i, j) , dada pelo número de vértices que estão posicionados a uma distância de i e j menor que o comprimento

desta aresta, em outras palavras:

$$\mathcal{C}_{ij} = |\{q \in \mathcal{V} : w_{iq} \leq w_{ij} \wedge w_{jq} \leq w_{ij}\}|.$$

4.1.6. Valores para Grafos Conhecidos

Usualmente, para se analisar a robustez de um grafo qualquer com relação à sua conectividade, utiliza-se as medidas espectralmétricas apresentadas anteriormente para grafos já conhecidos, como valores de referência, e avalia-se o quão próximo de um grafo regular com maior robustez para uma configuração similar a esse grafo, estão esses valores.

A seguir, na Figura 4.1, são apresentados 6 grafos usualmente utilizados na literatura como fontes de comparação para índices de robustez em análises quantitativas. Todos os grafos apresentados são não direcionados e possuem 10 nós, variando o número de arestas e a estrutura de conexão destas. Na Figura 4.1a é apresentado um grafo linear, na Figura 4.1b um grafo cíclico, na Figura 4.1c o grafo de Petersen, que é regular com grau 3, na Figura 4.1d um grafo regular de grau 3, na Figura 4.1e um grafo regular em malha e finalmente na Figura 4.1f um grafo completo.

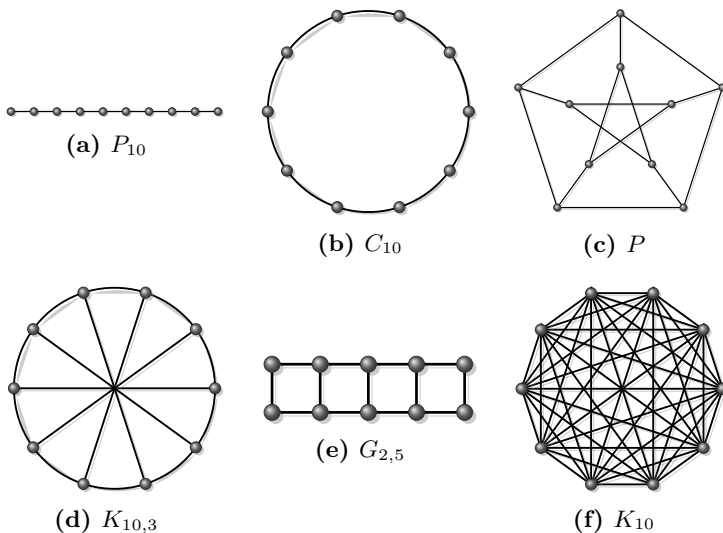


Figura 4.1: Grafos utilizados como referência.

A Tabela 4.1 contém os valores para a conectividade algébrica (λ_2), resistência efetiva (R), número de árvores geradoras (ξ), aresta-conectividade (κ_e), número de arestas (ϵ) e vértice-conectividade (κ_v) para todos os grafos apresentados na Figura 4.1.

Tabela 4.1: Medidas espectrométricas para grafos conhecidos.

grafo	λ_2	R	ξ	κ_e	κ_v	ϵ
P_{10}	0.098	165	1	1	1	9
C_{10}	0.382	82.5	10	2	2	10
P	2	33	2000	3	3	15
$K_{10,3}$	1.382	34.39	1815	3	3	15
$G_{2,5}$	0.382	56.1	209	2	2	13
K_{10}	10	9	10^8	9	9	45

Como se observa, para o caso do grafo cíclico e do grafo regular em malha, a conectividade algébrica não é sensível o bastante para detectar a variação no número de arestas não determinantes para a conectividade do grafo, e devido a isso, ambos os grafos possuem a mesma conectividade algébrica, mesmo sendo ligeiramente diferentes um do outro.

Além dos valores aqui apresentados, em [45, 46] são encontrados outros valores para a conectividade algébrica em diversos grafos, em [44] há valores para a resistência efetiva em alguns grafos e em [42, 45], valores para o número de árvores geradoras em certos grafos podem ser encontrados.

4.2. Simulações do Controle de Topologia para a Minimização do Número de *Links*

Nesta seção são apresentadas algumas simulações para avaliar a efetividade do controle de topologia para a minimização do número de *links* de comunicação, auxiliando na redução do consumo energético e na interferência causada pela comunicação entre os robôs. São realizados dois testes pontuais: no primeiro, uma rede de robôs móveis arbitrária é submetida ao controle de topologia e, em seguida, verifica-se a redução no consumo de energia e na interferência média para a topologia resultante; no segundo, uma rede com um grande número de robôs é submetida ao mesmo algoritmo e, após algumas iterações, avalia-se o estado da rede, bem como o seu grau de conectividade e a

eficiência na redução do consumo energético e interferência relativas a comunicação entre os robôs.

4.2.1. Teste 1 - Avaliação da Redução no Consumo Energético e Interferência

Este teste é realizado em uma topologia arbitrária com 10 robôs não holonômicos dispostos num plano e cujas posições, raio de comunicação e raio de cobertura são apresentados na Tabela 4.2.

Tabela 4.2: Configuração inicial de posição, raio de comunicação e raio de cobertura (valores em metros).

robô	x	y	r^{com}	r^{cov}
1	10.15	30.95	7	2.5
2	21.66	12.06	15	3.5
3	8.41	21.87	10	3.5
4	27.24	12.54	10	3.5
5	19.98	22.43	10	3.5
6	20.41	32.64	15	3.5
7	24.9	18.76	10	3.5
8	15.37	23.61	10	3.5
9	24.96	39.97	10	4.5
10	18.95	32.75	12	3.5

Como se percebe os raios de comunicação dos robôs 1, 2, 6 e 10 são diferentes dos demais, assim como o raio de cobertura dos robôs 1 e 9. Com isso a topologia resultante será heterogênea, apresentando ligações direcionadas. A Figura 4.2 mostra essa topologia. As setas em magenta representam ligações direcionadas e o círculo em torno dos robôs é a área de comunicação de cada um. Os nós em azul são robôs que não enviam informação para nenhum outro, pois todos estão fora do seu raio de comunicação.

O algoritmo de controle de conectividade possui os seguintes parâmetros para sua função objetivo:

$$\Delta k = 0.5s \quad p = 5 \quad \gamma_i = 0.5$$

$$\alpha_{ij} \in \{0, 1\} \quad a_{ij} \in \{0, 1\}$$

A saturação do sinal do controle de movimento assume os se-

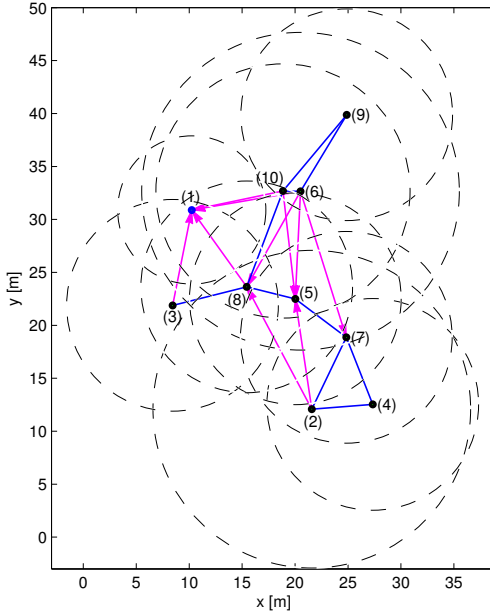


Figura 4.2: Configuração inicial da topologia: círculos representam a área de comunicação de cada robô e as setas indicam troca de informação direcionada.

guintes valores para as velocidades lineares em m/s:

$$v^{max} = [1 \quad 1] \quad v^{min} = [-1 \quad -1].$$

Após a execução de 100 iterações do controle de topologia, a rede resultante apresenta-se completamente minimizada, constituída apenas pela árvore geradora mínima, o que pode ser visto na Figura 4.3. O raio de comunicação de todos os robôs também sofreu uma significativa redução, como se observa na Tabela 4.3.

Na Figura 4.4 é possível perceber a variação do custo energético da rede durante a simulação. Houve uma grande redução no consumo energético da rede até a vigésima iteração do algoritmo, logo após há pequenos aumentos e quedas sucessivas, de forma que a partir da iteração 80, aproximadamente, o custo energético se estabiliza.

A interferência oriunda da sobreposição de áreas de comunicação da rede pode ser analisada através da cobertura média das arestas do grafo desta rede e na Figura 4.5 é apresentada a variação dessa cobertura durante a simulação. Como se observa, o valor da cobertura média

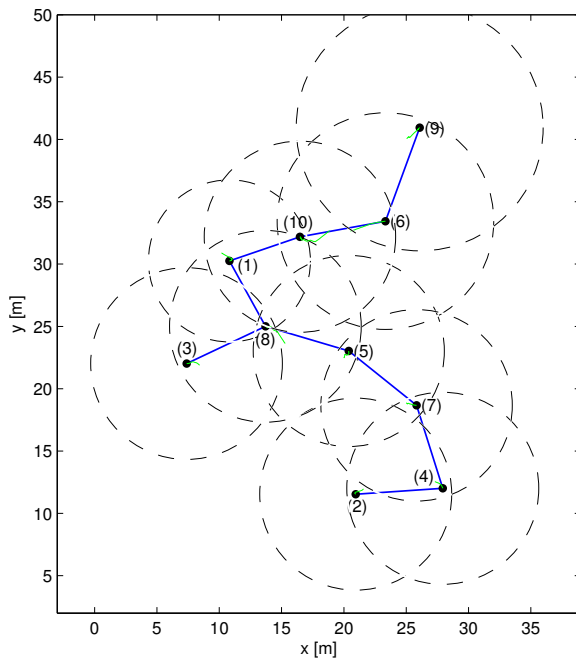


Figura 4.3: Topologia resultante após 100 iterações do controle de topologia.

Tabela 4.3: Configuração final de posição, raio de comunicação e raio de cobertura (valores em metros).

robô	x	y	r^{com}	r^{cov}
1	10.81	30.25	6.4963	2.5
2	20.95	11.54	7.7032	3.5
3	7.38	22.02	7.6936	3.5
4	27.93	12.01	7.7032	3.5
5	20.39	23.03	7.6704	3.5
6	23.33	33.44	8.6927	3.5
7	25.83	18.67	7.6793	3.5
8	13.7	25	7.6936	3.5
9	26.08	40.94	9.9	4.5
10	16.47	32.19	7.6729	3.5

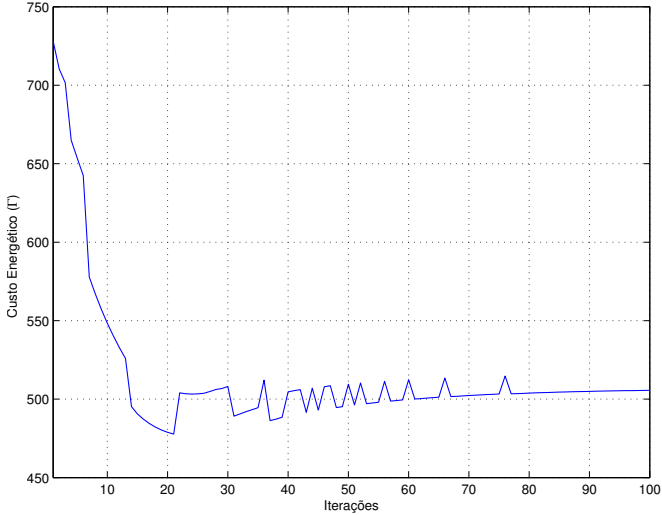


Figura 4.4: Custo energético da rede durante 100 iterações.

decrece consideravelmente durante a simulação, estabilizando-se após a iteração 80 no valor zero.

Analisando os valores do custo energético ao final da simulação, percebe-se uma redução de aproximadamente 30.55% nesse valor, o que indica uma redução no consumo de energia da rede proporcional. Também houve uma grande redução da cobertura média das arestas ao final da simulação, tal valor decaiu de 0.6 no início da simulação para 0 ao final desta, o que indica a redução completa da interferência causada pela sobreposição de áreas de transmissão nesta rede. Esses valores comprovam a efetividade do algoritmo em reduzir o consumo energético envolvido nos processos de comunicação e a interferência causal oriunda do conflito transmissões concorrentes de cada robô numa rede de comunicação arbitrária.

Também pode-se verificar a variação do número de árvores geradoras do grafo da rede durante a execução do controle de topologia. A Figura 4.6 apresenta esta variação durante as 100 iterações do controle de topologia. Como se percebe, nas primeiras 10 iterações, o número de árvores geradoras decresce rapidamente, ocorrendo pequenas variações e estabilizando-se em 1, após a iteração 80. Esse resultado é o esperado, já que não restaram subciclos na topologia otimizada.

Analisando a matriz de adjacências ponderada resultante \bar{A} em relação à matriz de distâncias entre todos os robôs \mathcal{W} , tem-se que as

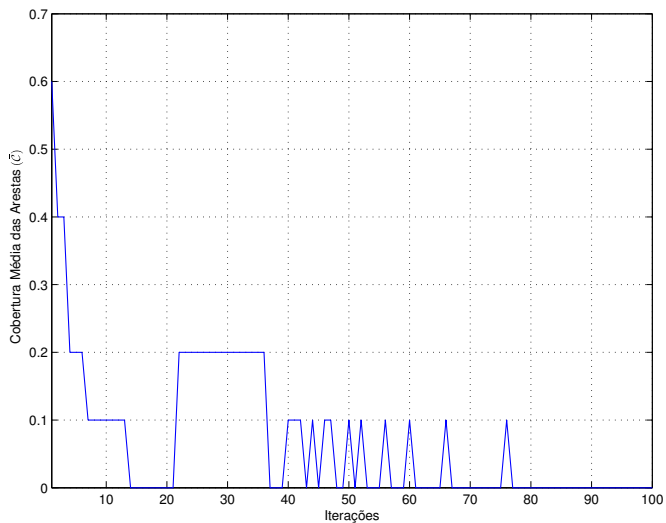


Figura 4.5: Cobertura média das arestas da rede durante 100 iterações.

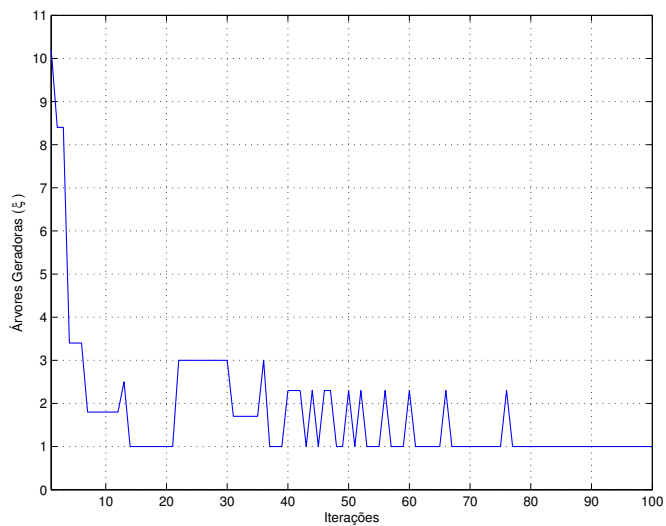


Figura 4.6: Variação do número de árvores geradoras da rede durante 100 iterações.

arestas escolhidas são realmente as melhores possíveis³, o que demonstra que a árvore geradora sob a qual se encontra a rede é mínima.

$$\bar{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.99 & 0 & 5.98 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6.99 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 6.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6.96 & 6.97 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.99 & 6.97 \\ 0 & 0 & 0 & 6.97 & 6.96 & 0 & 0 & 0 & 0 & 0 \\ 5.99 & 0 & 6.99 & 0 & 6.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7.99 & 0 & 0 & 0 & 0 \\ 5.98 & 0 & 0 & 0 & 0 & 6.97 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{W} = \begin{bmatrix} 0 & 21.2 & 8.9 & 25 & 12 & 12.9 & 18.9 & 5.99 & 18.6 & 5.98 \\ 21.2 & 0 & 17.1 & 7 & 11.4 & 22 & 8.6 & 15.2 & 29.9 & 21.1 \\ 8.9 & 17.1 & 0 & 22.8 & 13 & 19.6 & 18.7 & 6.99 & 26.6 & 13.6 \\ 25 & 7 & 22.8 & 0 & 13.3 & 21.9 & 6.97 & 19.2 & 28.9 & 23.2 \\ 12 & 11.4 & 13 & 13.3 & 0 & 10.8 & 6.96 & 6.97 & 18.7 & 9.9 \\ 12.9 & 22 & 19.6 & 21.9 & 10.8 & 0 & 14.9 & 12.7 & 7.99 & 6.97 \\ 18.9 & 8.6 & 18.7 & 6.97 & 6.96 & 14.9 & 0 & 13.6 & 22.2 & 16.4 \\ 5.99 & 15.2 & 6.99 & 19.2 & 6.97 & 12.7 & 13.6 & 0 & 20.1 & 7.6 \\ 18.6 & 29.8 & 26.6 & 28.9 & 18.7 & 7.99 & 22.2 & 20.1 & 0 & 13 \\ 5.98 & 21.1 & 13.6 & 23.2 & 9.9 & 6.97 & 16.4 & 7.6 & 13 & 0 \end{bmatrix}$$

4.2.2. Teste 2 - Efetividade em Redes Densas

Nesta simulação um grupo de 100 robôs não holonômicos dispostos em um plano são submetidos ao controle de topologia para a minimização da comunicação, e verifica-se o comportamento da topologia em termos da cobertura média das arestas e do custo energético da rede em seu estado final.

A rede é homogênea, todos os robôs tem raio de comunicação $r^{com} = 10m$ e raio de cobertura $r^{cov} = 3m$. Os parâmetros do controle de conectividade são definidos como no teste anterior.

Na Figura 4.7a, observa-se a disposição inicial dos robôs e a topologia de comunicação antes da aplicação do controle de topologia.

³O elemento zero nestas matrizes indica inalcançabilidade.

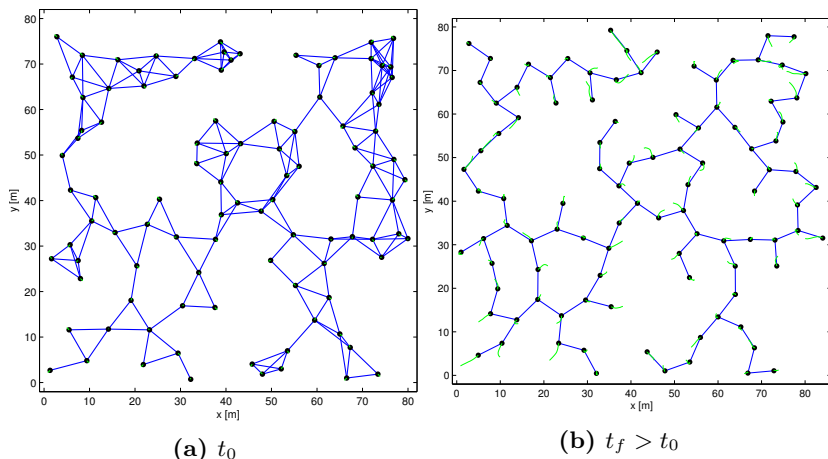


Figura 4.7: Topologia de comunicação em dois instantes diferentes.

Após a execução de 100 iterações, a topologia final apresenta-se minimizada, como se observa na Figura 4.7b. A topologia resultante não é composta apenas por uma árvore geradora mínima, pois como o algoritmo utiliza somente informação localizada à 1-hop para minimizar a comunicação, ciclos com 6 nós ou mais não são percebidos, originando uma LMST, que apesar de não ser ótima, cumpre o seu papel, minimizando o número de ligações na rede.

Na Figura 4.8, nota-se a variação do custo energético da rede durante a simulação. Como o esperado, esse custo é reduzido significativamente durante a execução do controle de topologia e ao final da simulação a redução é de cerca de 45.98% em relação ao valor encontrado no início da simulação.

A Figura 4.9 refere-se a variação da cobertura média das arestas da rede durante a execução das 100 iterações. Como se observa, assim como no caso do custo energético, há uma grande redução na cobertura média das arestas, atingindo aproximadamente 5.36% do valor inicial no final das 100 iterações, o que implica em uma diminuição de 94.64% na interferência no processo de comunicação oriunda da sobreposição de áreas de transmissão.

Assim como na primeira simulação, neste caso o controle de topologia se mostrou efetivo na ação de minimizar o consumo energético da rede e reduzir a interferência causada pela comunicação.

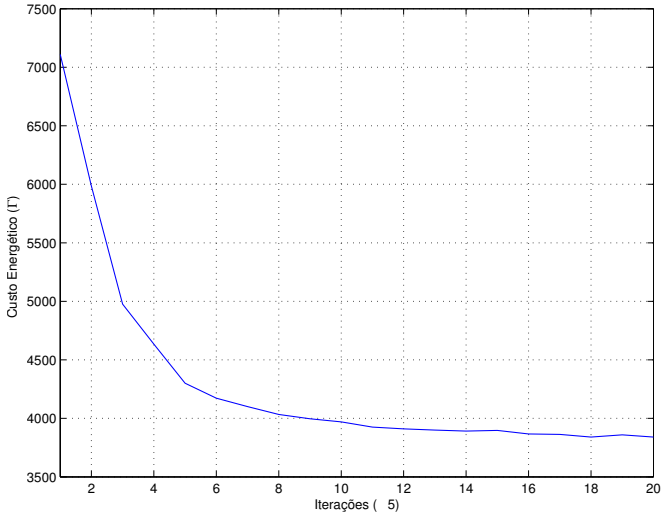


Figura 4.8: Variação do custo energético da rede.

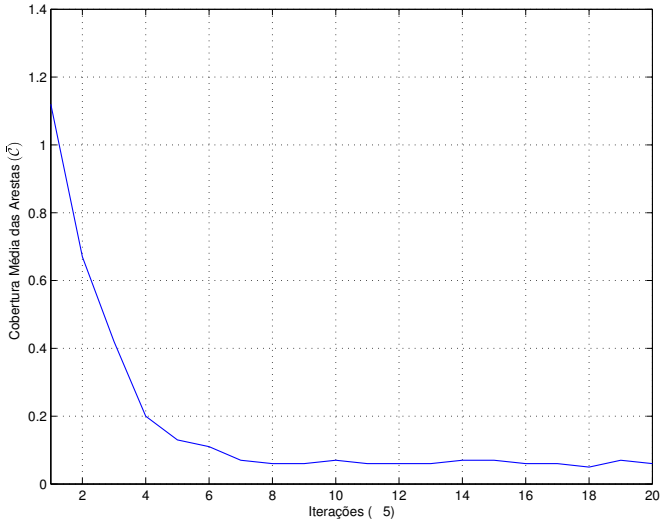


Figura 4.9: Variação da cobertura média da rede.

4.3. Simulações do Controle de Topologia para Manutenção da Conectividade

Para avaliar a atuação do algoritmo de controle de topologia para a manutenção da conectividade utilizando as duas abordagens de correção de nós críticos apresentadas, foram realizados 3 testes específicos: no primeiro uma rede, cuja topologia inicial é não bi-conexa, é submetida à algumas iterações dos algoritmos propostos, e ao final avalia-se a robustez da rede resultante em relação aos grafos utilizados como referência; no segundo teste, a topologia testada sofre com perdas de comunicação por parte de alguns robôs e ocorre a entrada de novos indivíduos na rede; e finalmente no terceiro teste, as duas abordagens para correção de nós críticos são avaliadas em uma rede com muitos robôs, e compara-se o desempenho de ambas as abordagens nesse cenário.

Para simplificar a referência aos algoritmos propostos, o controle de topologia utilizando a correção de nós críticos baseada em *links* virtuais poderá ser chamado de controle de topologia VLINK ou simplesmente VLINK, e o controle de topologia utilizando a correção de nós críticos através do Problema do Caixeiro Viajante poderá ser referenciado como controle de topologia TSP ou apenas TSP.

4.3.1. Teste 1 - Robustez da Topologia Resultante

Nesse teste, são utilizados 6 robôs não holonômicos dispostos em um plano, com suas posições iniciais sendo definidas conforme a Tabela 4.4. Todos possuem o mesmo raio de comunicação e raio de cobertura e, portanto, trata-se de uma rede homogênea, tais valores também são definidos na Tabela 4.4 e todos os valores são apresentados em metros.

A topologia inicial de comunicação é descrita pela Figura 4.10. O círculo em torno dos robôs é a área de cobertura de cada um.

O algoritmo de conectividade possui os seguintes valores para o passo de atualização e horizonte de predição:

$$\Delta k = 0.5s \quad p = 5.$$

E os parâmetros para a função objetivo são definidos como segue:

$$\begin{aligned} \gamma_i &= 0.5 \\ \alpha_{ij} &\in \{0, 1\} \\ \bar{\alpha}_{ij} &\in \{0, 1\} \end{aligned}$$

Tabela 4.4: Configuração inicial de posição, raio de comunicação e raio de cobertura para o teste 1.

robô	x	y	r^{com}	r^{cov}
1	0	0	10	4.5
2	0	6	10	4.5
3	5	-0.3	10	4.5
4	2.5	-0.9	10	4.5
5	10	-5.2	10	4.5
6	11	5	10	4.5
7	11	12	10	4.5
8	11	16	10	4.5
9	15	5	10	4.5
10	24	5	10	4.5

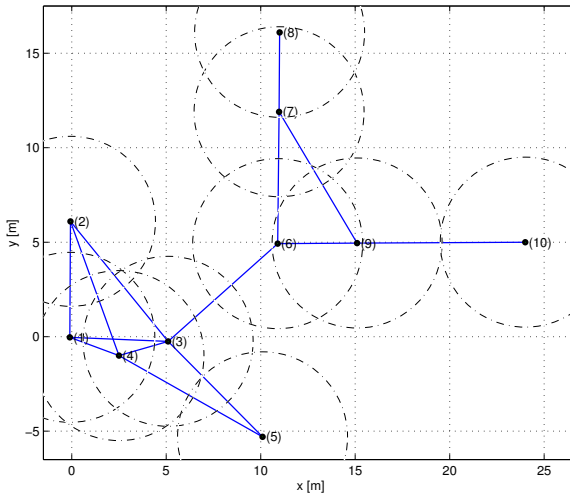


Figura 4.10: Configuração inicial da topologia: círculos em torno dos robôs indicam a área de cobertura de cada um.

Após aproximadamente 200 iterações entre os robôs, a topologia resultante para ambas as abordagens de correção de nós críticos utilizando o controle de conectividade com os parâmetros definidos anteriormente é exibida na Figura 4.11. Na Figura 4.11a observa-se o estado da rede para controle de topologia com VLINK e na Figura 4.11b é apresentado o estado da rede para o controle de topologia uti-

lizando TSP. Nota-se que os grafos finais para as duas abordagens são diferentes, e aparentemente, a utilização do algoritmo de resolução do Problema do Caixeiro Viajante para a eliminação dos nós críticos é mais eficiente, neste primeiro instante.

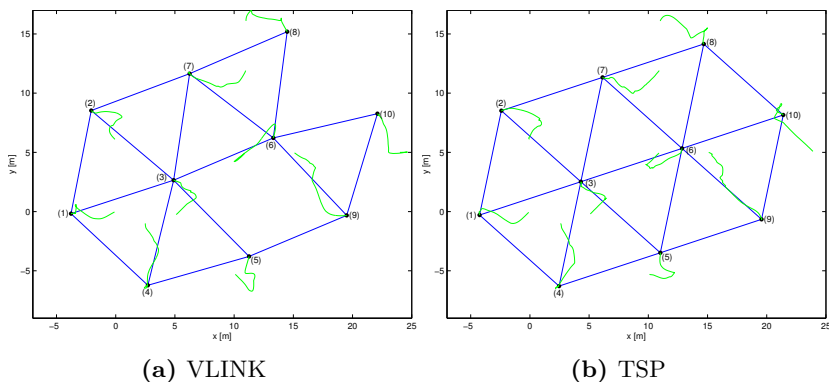


Figura 4.11: Estado da rede após 200 iterações.

Na Figura 4.12 são apresentadas as variações da conectividade algébrica para as duas abordagens durante 250 iterações de comunicação entre os robôs. Nota-se, mais uma vez, que a abordagem utilizando o Problema do Caixeiro viajante (linha), atinge um grau de conectividade maior em um tempo menor do que a abordagem utilizando *links* virtuais (tracejado), no entanto, as duas topologias atingem o mesmo grau de conectividade no fim da simulação.

A Figura 4.13 expõe a variação da resistência efetiva da rede de robôs durante a execução do algoritmo de controle de topologia utilizando as duas abordagens para correção de nós críticos. Os resultados se assemelham aos apresentados pela conectividade algébrica, onde a abordagem para correção de nós críticos utilizando o Problema do Caixeiro Viajante (linha) faz a rede atingir sua melhor conectividade em tempo menor – exatamente na iteração 117 – que a abordagem utilizando *links* virtuais (tracejado) – que atinge o mesmo valor na iteração 227.

A variação do número de árvores geradoras para a rede durante a execução do algoritmo utilizando as duas abordagens é descrita na Figura 4.14: em tracejado é apresentada a variação para a abordagem baseada em *links* virtuais e em linha contínua, para a abordagem utilizando o Problema do Caixeiro Viajante. Percebe-se que o número de árvores geradoras para a execução do controle de topologia para a

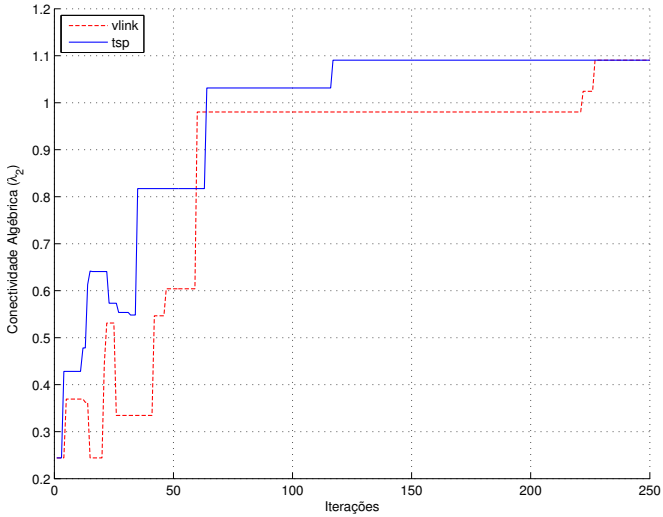


Figura 4.12: Variação da conectividade algébrica durante 250 iterações.

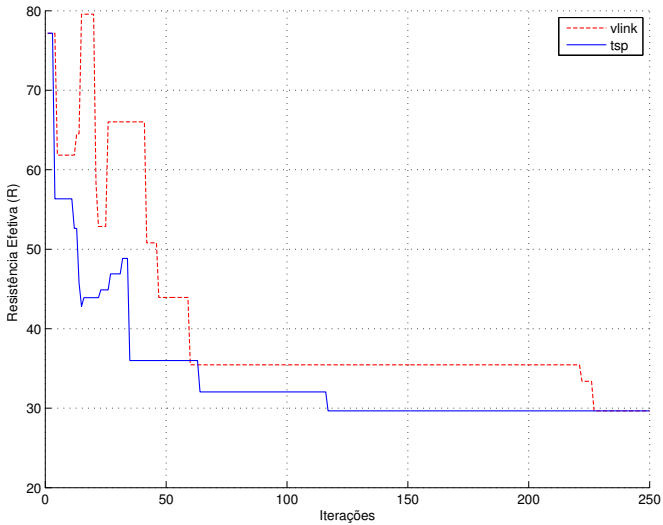


Figura 4.13: Variação da resistência efetiva durante 250 iterações.

primeira abordagem é bem maior do que o apresentado para a segunda, o que indica uma maior quantidade de *links* entre os robôs nesta abor-

dagem.

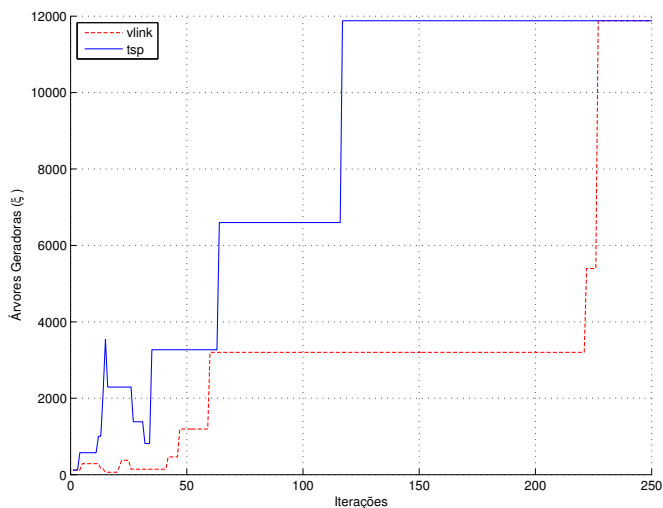


Figura 4.14: Variação do número de árvores geradoras durante 250 iterações.

Na Figura 4.15, nota-se a variação do número de arestas na rede de comunicação durante a execução do controle de topologia com as duas abordagens de correção de nós críticos. Durante a execução do controle de topologia baseado no TSP, o número de *links*, no início da atuação da simulação é muito maior do que o apresentado pelo controle de topologia utilizando o VLINK, o que é reflexo da forma como os nós críticos são corrigidos.

Analisando as métricas de robustez citadas para cada caso, percebe-se que o tempo de convergência do algoritmo de controle de topologia para a manutenção da conectividade utilizando como técnica de correção de nós críticos o Problema do Caixeiro Viajante é cerca de 48.46% menor do que o tempo de convergência apresentado quando o controle de topologia utiliza a abordagem baseada em *links* virtuais. Isso não significa, necessariamente, que a técnica de *links* virtuais para a correção de nós críticos seja ruim, apenas que sua velocidade de convergência não é apropriada para situações mais instáveis, em que a topologia da rede deve atingir um estado bi-conexo o mais rápido possível.

A rede, durante a simulação, apresenta uma topologia robusta, é notavelmente bi-conexa, atingido a tri-conectividade no final da simula-

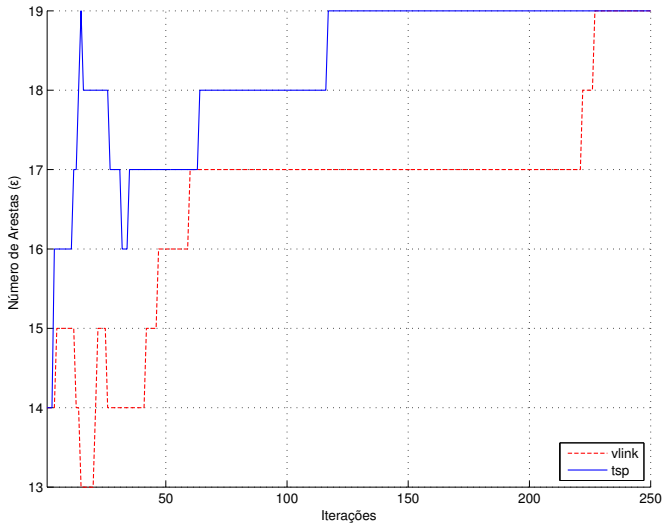


Figura 4.15: Variação do número de *links* durante 250 iterações.

ção. Isso implica que até dois nós adjacentes podem ser desconectados, sem que haja uma desconexão na rede como um todo. Em termos comparativos, a resistência efetiva da topologia final é de $R = 29.66$ o que corresponde aproximadamente a um valor 10.1% menor que a resistência efetiva do grafo de Petersen, um dos melhores grafos bi-conexos com 10 nós. O número de árvores geradoras também é bastante elevado, com $\xi = 11880$ árvores geradoras, o que é cerca de 83.16% maior do que o valor apresentado pelo grafo de Petersen. Entretanto, em relação ao grafo completo K_{10} , os valores se mostram bastante tímidos, com o número de árvores geradoras sendo 99.9% menor que o valor apresentado pelo grafo completo com 10 vértices, o que de fato é esperado, afinal o grafo completo é a forma mais robusta de um grafo com relação à conectividade.

Para comprovar o estado de bi-conectividade atingido pela rede de robôs utilizada nesta simulação, pode-se avaliar a matriz de adjacências que descreve esta rede para os dois métodos de correção de nós críticos, verificando a conectividade algébrica durante a remoção de cada robô e buscando a presença de vértices de corte ou nós críticos. Na ausência destes elementos, a topologia é comprovada bi-conexa.

A seguir são apresentadas as matrizes de adjacência para as topologias mostradas na Figura 4.11 após a execução do controle de topo-

gia utilizando o TSP e o VLINK para a correção dos nós críticos:

$$\underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}}_{A^{\text{tsp}}} \quad \underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{A^{\text{vlink}}}$$

A avaliação da bi-conectividade foi realizada testando a conectividade algébrica da rede resultante durante a remoção de cada um dos robôs individualmente. Se a conectividade algébrica é menor que zero quando se remove um determinado robô, então este robô é um nó crítico e a topologia não é bi-conexa.

Essa avaliação foi feita a partir das matrizes de adjacências apresentadas anteriormente e resultaram no gráfico da Figura 4.16, onde as barras em preto indicam a conectividade algébrica para o controle de topologia VLINK, enquanto que as barras em cinza representam esse mesmo valor para o controle de topologia TSP. O eixo horizontal indica quais robôs são removidos da rede durante a avaliação e o eixo vertical a conectividade algébrica.

Como se observa, a conectividade algébrica se mantém sempre acima de zero para todos os robôs avaliados, o que indica a inexistência de vértices de corte e, portanto, comprova o estado de bi-conectividade atingido pela rede após a execução do controle de topologia com ambas as abordagens para correção dos nós críticos.

4.3.2. Teste 2 - Convergência em Topologias Dinâmicas

Neste teste, a mesma topologia utilizada nas simulações anteriores é processada pelo algoritmo de controle de topologia quando ocorre instabilidades e robôs são desconectados da rede. É avaliado, também, como o algoritmo se comporta quando há inserção de novos robôs na rede e como a topologia se adapta à essa situação para manter o estado de bi-conectividade.

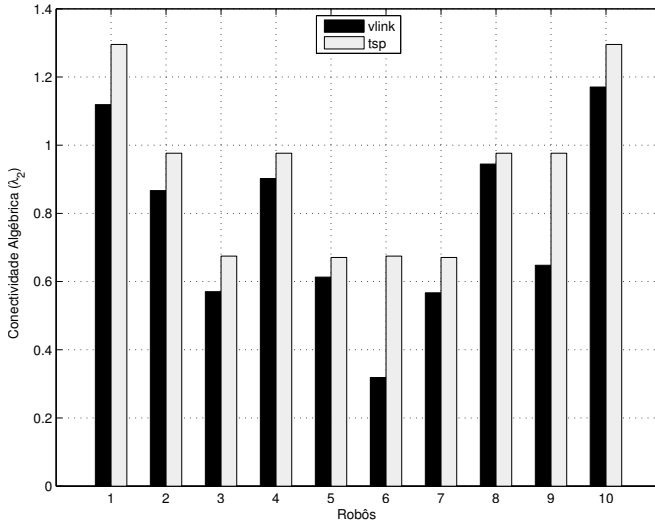


Figura 4.16: Variação da conexão algébrica durante a remoção de cada um dos robôs.

A Figura 4.17 refere-se a rede de comunicação no instante imediatamente posterior ao apresentado na Figura 4.11, onde o robô 3 perde a conexão com todos os outros e o algoritmo de controle de topologia tenta restaurar a conectividade.

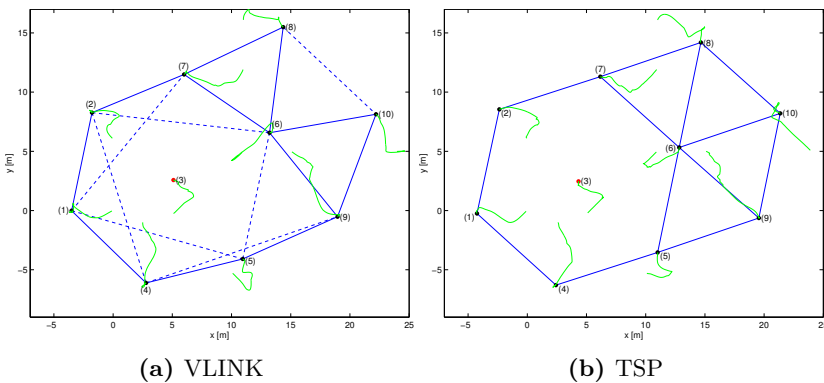


Figura 4.17: Topologia resultante quando o robô 3 perde a comunicação com todos os outros.

Na Figura 4.17a, observa-se o algoritmo de controle de topologia utilizando a abordagem de *links* virtuais para a correção dos nós críticos, enquanto que na Figura 4.17b, é mostrada a mesma situação para a abordagem utilizando o algoritmo de resolução do TSP. Nota-se que a ação do controle de topologia nas duas abordagens é diferente, pois enquanto na primeira abordagem os nós críticos locais tentam corrigir sua criticidade, criando uma série de ligações virtuais entre seus vizinhos mais próximos, na segunda abordagem, não há qualquer ação visível do algoritmo de controle de topologia, embora este esteja sendo executado.

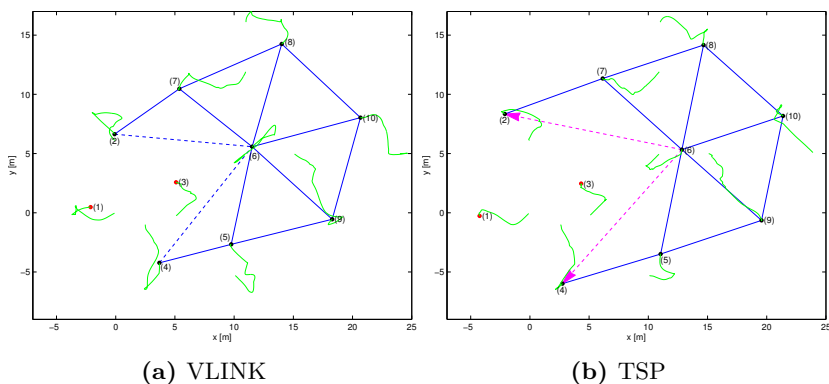


Figura 4.18: Topologia resultante quando o robô 1 perde a comunicação com todos os outros.

Na Figura 4.18, o robô 1 perde a conexão com todos os demais, tornando a rede não bi-conexa, com os robôs 7 e 5 nós críticos. As figuras 4.18a e 4.18b representam a ação do controle de conectividade sujeito a cada uma das abordagens de correção de nós críticos. As ligações direcionadas na Figura 4.18b ocorrem porque a abordagem para remoção de nós críticos utilizando o TSP gera um grafo ótimo diferente para cada robô, devido à informação limitada a que cada um tem acesso, enquanto que na abordagem baseada em *links* virtuais, é o nó crítico quem cria e coordena a manutenção da ligação virtual entre os vizinhos lógicos, que têm a mesma informação sobre as ligações virtuais a que pertencem.

Finalmente, na Figura 4.19, a bi-conectividade é restaurada na rede, e todos os nós críticos são corrigidos. Os *links* virtuais na Figura 4.19a se mantém, porque ainda não houve tempo suficiente para os robôs 7 e 5 receberem informação sobre a nova ligação que se formou

entre os robôs 2 e 4, e enviarem a mensagem de deleção dos *links* virtuais para seus respectivos vizinhos, isso possivelmente acontece na iteração seguinte.

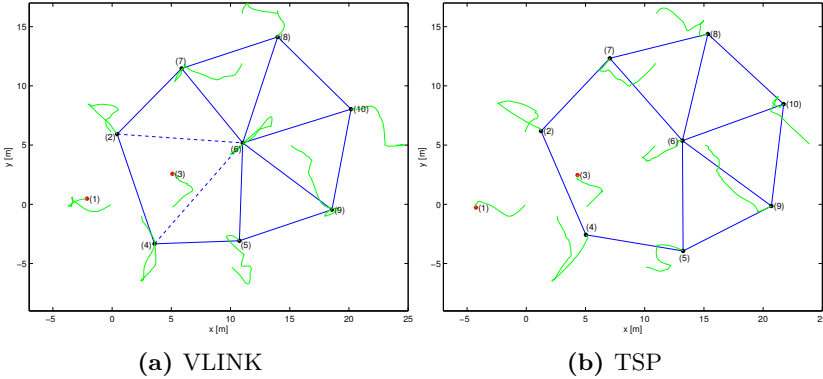


Figura 4.19: A bi-conectividade é restaurada na rede.

Na Figura 4.20 um novo robô aparece na vizinhança do robô 8 e é criada uma ligação entre os dois, o robô 8 torna-se um nó crítico e um *link* virtual é criado entre os robôs 10 e 11 para corrigir essa criticidade. Em ambas as abordagens para a correção dos nós críticos, a ação de correção é a mesma.

Como observado nas simulações, as abordagens para correção de nós críticos funcionam de maneira adequada em ambientes dinâmicos e o controle de conectividade proposto, garante que a topologia mantenha seu estado de conectividade, mesmo quando robôs perdem a conexão com o restante do grupo ou novos robôs aparecem.

4.3.3. Teste 3 - Desempenho em Topologias Densas

Nesta simulação, o algoritmo de controle de topologia para a manutenção da conectividade é analisado em uma rede com um grande número de robôs. Em um espaço bi-dimensional são dispostos, de maneira randômica, 100 robôs não holonômicos, de forma que inicialmente a topologia de comunicação do grupo é altamente esparsa e não é bi-conexa, como mostra a Figura 4.21. Todos os robôs possuem raio de comunicação $r^{com} = 10$ m e raio de cobertura $r^{cov} = 4.5$ m, os parâmetros da função objetivo do controle de conectividade são os mesmos apresentados no teste 1.

Após a execução de 500 iterações de comunicação entre os robôs,

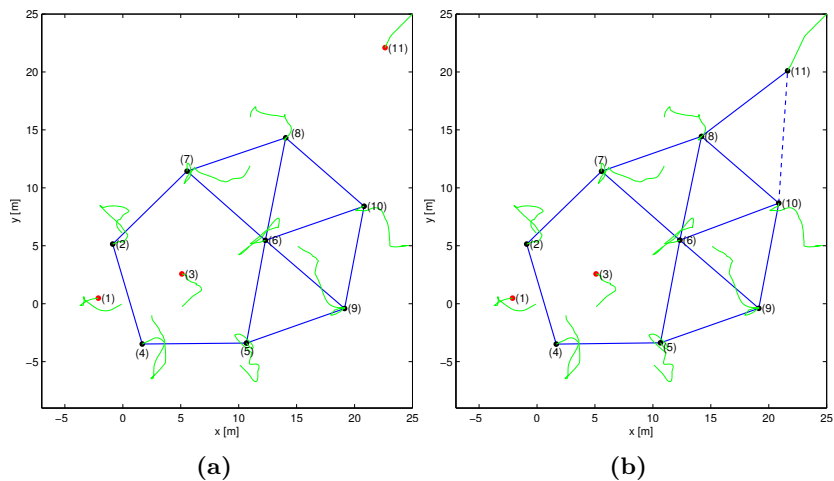


Figura 4.20: Topologia resultante quando um novo robô é detectado.

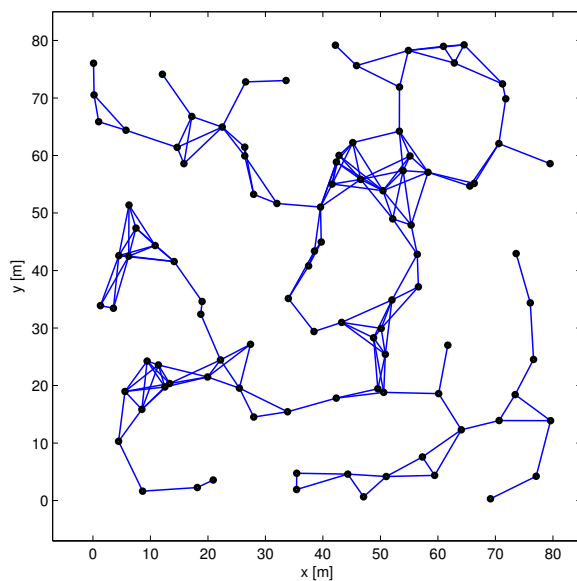


Figura 4.21: Configuração inicial da topologia com 100 robôs dispostos de maneira aleatória.

nota-se na Figura 4.22, o estado final da rede para o controle de topologia utilizando a correção de nós críticos baseada em *link* virtual (Figura 4.22a) e para o controle de topologia utilizando a correção de nós críticos com o TSP (Figura 4.22b).

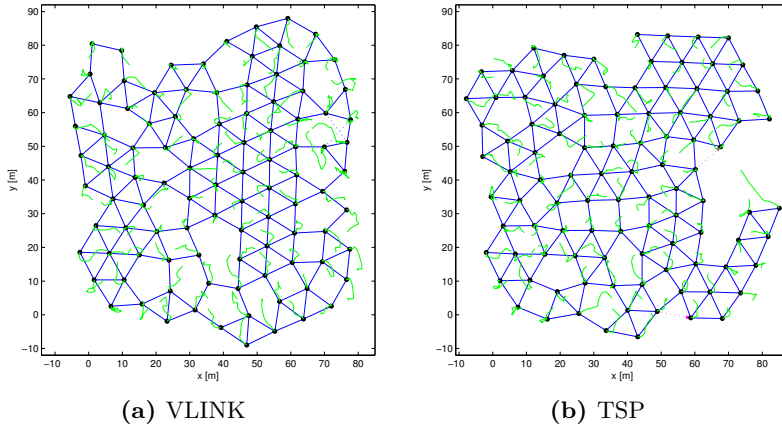


Figura 4.22: Topologia resultante após 500 iterações do controle de topologia.

Na Figura 4.23 tem-se a variação da conectividade algébrica para a rede. Como é perceptível, o grafo após a execução do controle de topologia VLINK é levemente mais conexo que o grafo resultante do controle de topologia TSP.

A Figura 4.24 apresenta a variação da resistência efetiva do grafo da rede durante a simulação, a abordagem para correção de nós críticos utilizando *links* virtuais se comporta melhor que a abordagem utilizado o TSP, apresentado uma menor resistência efetiva durante a simulação.

Na Figura 4.25 é possível analisar a variação no número de vértices durante as 500 iterações do algoritmo de controle de conectividade. Neste caso, percebe-se um comportamento interessante para a topologia, mesmo com mais arestas, o grafo resultante do controle de topologia com a abordagem TSP apresenta conectividade menor do que o derivado da abordagem VLINK, isso ocorre porque, como se percebe pela Figura 4.22, o grafo resultante da primeira abordagem (Figura 4.22b) é menos agrupado, enquanto que o gerado pela segunda abordagem (Figura 4.22a) é menos denso, porém mais agrupado, isso influencia a conectividade algébrica, pois quanto mais agrupado é um

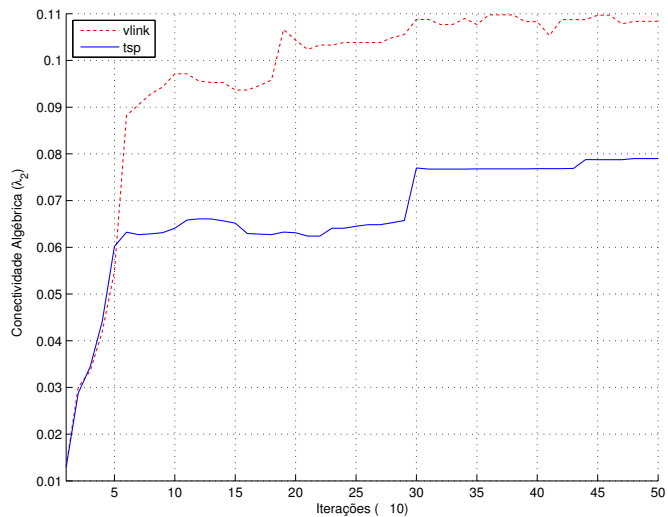


Figura 4.23: Variação da conectividade algébrica durante a simulação.

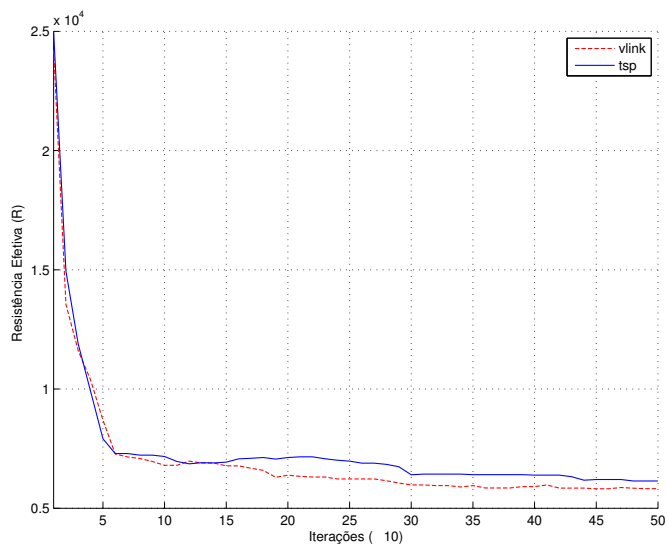


Figura 4.24: Variação da resistência efetiva durante a simulação.

grafo, mais conexo este tende a ser.

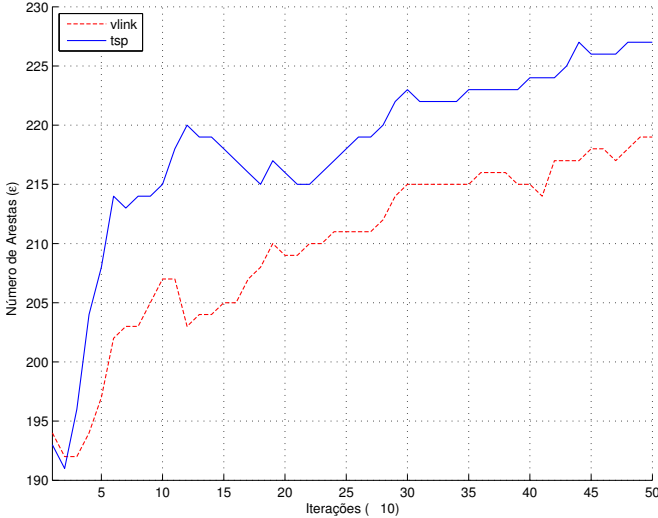


Figura 4.25: Variação do número de *links* durante a simulação.

A rede final resultante da aplicação do controle de topologia com as duas abordagens para correção de nós críticos, atingiu o estado esperado de conectividade e, no geral, ambas as topologias são equivalentes, apresentando apenas algumas variações resultantes da ação de correção de nós críticos. Em termos computacionais, o controle de topologia utilizando TSP é mais custoso que a versão baseada *links* virtuais, o que pode ser verificado quando se analisa o tempo gasto por cada um: enquanto a primeira abordagem consumiu cerca de 768 segundos para executar 500 iterações, a segunda abordagem levou somente 334 segundos para concluir a mesma simulação, o que implica que a abordagem para correção de nós críticos VLINK foi 56.5% mais rápida que a abordagem utilizando TSP.

4.4. Testes em Plataforma Experimental

Nesta seção, os algoritmos para controle de topologia anteriormente propostos são aplicados em robôs reais visando verificar a sua efetividade nestes dispositivos. Para tal, foi desenvolvido uma biblioteca de comunicação que permite integrar o simulador dos algoritmos de controle de topologia com o ROS (detalhes sobre a implementação dessa biblioteca podem ser encontrados no apêndice C). Optou-se por essa

abordagem devido a flexibilidade da interface de comunicação provida por este ambiente e ao fato de que é utilizado em diversos robôs comerciais, possuindo uma vasta compatibilidade com os mais variados periféricos disponíveis para tais robôs.

Foram utilizados dois robôs comerciais, o P3AT⁴ e o P3DX⁵ que são fabricados pela *Pioneer*. A Figura 4.26 mostra ambos os robôs e o ambiente onde os testes foram realizados. O método de localização utilizado por eles é baseado em odometria.



Figura 4.26: Robôs e ambiente de testes: à esquerda P3DX e à direita P3AT.

Os testes também são realizados com robôs virtuais, simulados no Stage⁶, que possui uma versão integrada ao ROS e permite que tais robôs se comportem de forma similar aos robôs reais utilizados.

Como os tanto os robôs simulados no Stage, quanto os robôs comerciais da *Pioneer* são diferenciais, um controle de movimento de “baixo nível” que traduz as velocidades omnidirecionais provenientes do controle de conectividade em velocidades adequadas à dinâmica destes robôs foi desenvolvido e detalhes de sua implementação estão disponíveis no apêndice B.

A seguir, são apresentados os testes com os três algoritmos propostos para o controle de topologia.

⁴<<http://www.mobilerobots.com/ResearchRobots/P3AT.aspx>>

⁵<<http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx>>

⁶<<http://playerstage.sourceforge.net/>>

4.4.1. Minimização da Comunicação Utilizando a Plataforma Experimental

O teste realizado nesta seção leva em consideração a mesma topologia apresentada na seção 4.2.1. A diferença básica é que, neste caso, os robôs são diferenciais e os robôs 6 e 10 foram substituídos pelos robôs da *Pioneer*, apresentados anteriormente.

Os parâmetros para o controle de conectividade são os mesmos apresentados na seção 4.2.1, com exceção do passo temporal, que assume o valor $\Delta k = 0.1s$. No entanto, há a adição dos parâmetros do controle de “baixo nível”, definidos como segue:

$$\begin{aligned}\delta_{max}^{\theta} &= 0.35 \text{ rad}, & \delta_{min}^{\theta} &= 0.01 \text{ rad}, \\ \delta_{max}^x &= 0.1 \text{ m}, & \delta_{min}^x &= 0.01 \text{ m}, \\ v_{max}^{\theta} &= 1 \text{ rad/s}.\end{aligned}$$

Como o passo temporal é o mesmo para o controle de conectividade e para o controle de movimento de “baixo nível”, o valor de $\delta_{max}^x = \Delta k \cdot v_{max}^x = 0.1m$, já que a velocidade linear máxima é $1m/s$.

A Figura 4.27 exibe a topologia de comunicação para o grupo de robôs após 100 iterações do controle de topologia. Como se observa, a topologia resultante é praticamente idêntica à topologia resultante na simulação com robôs virtuais, apresentada anteriormente. Existem, apenas, algumas alterações no caminho percorrido pelos robôs, mas isso se dá por suas restrições holonômicas, aqui consideradas.

4.4.2. Manutenção da Conectividade na Plataforma Experimental

Neste experimento, os robôs apresentados anteriormente, são submetidos às mesmas condições apresentadas pela simulação realizada na seção 4.3.2, em que um grupo de 10 robôs dispostos em uma topologia inicial aleatória, precisa construir uma topologia bi-conexa e garantir que esta estrutura se mantenha, ainda que esta esteja sob influência de instabilidades e perdas de nós da rede.

Os valores para o controle de conectividade são os mesmos apresentados na simulação referida e os valores para o controle de movimento são definidos conforme o teste anterior. Os robôs 2 e 4 são substituídos pelos robôs P3DX e P3AT, respectivamente.

A Figura 4.28 mostra o estado da topologia de comunicação após 100 iterações do controle de topologia para as duas estratégias de correção de nós críticos. Como se observa, a topologia resultante apresenta

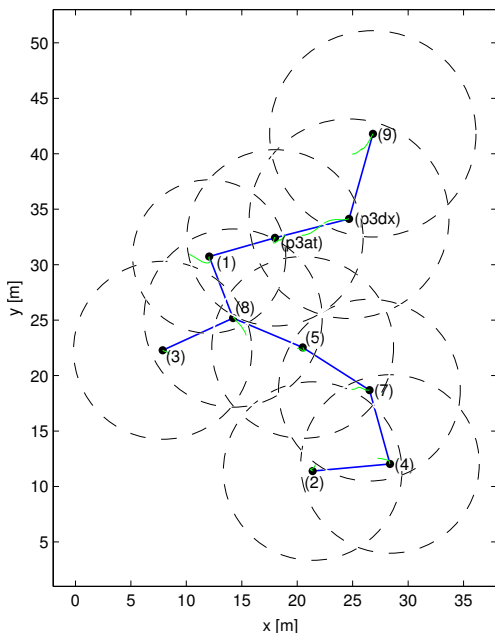


Figura 4.27: Topologia resultante após 100 iterações do controle de topologia.

uma estrutura bi-conexa, e os robôs P3DX e P3AT, assim como os demais, assumem as posições esperadas, dada a topologia inicial.

A Figura 4.29 descreve a topologia de comunicação no instante em que o robô 1 perde comunicação com todos os demais, anteriormente o robô 3 havia também perdido a comunicação com todos os demais.

Após a ação do controle de topologia, em construir uma nova estrutura virtual bi-conexa, o controle de conectividade atrai os vizinhos virtuais e, como se observa na Figura 4.30, a bi-conectividade é restaurada na rede.

Os robôs 2 e 4, foram substituídos pelos robôs *Pioneer* justamente porque apresentam um papel essencial na restauração da bi-conectividade após os robôs 1 e 3 perderem comunicação com os demais. Percebe-se, em ambos os experimentos, que tais robôs comportaram-se de maneira satisfatória, efetivando a estratégia de controle em “baixo nível” e os parâmetros escolhidos.

Nota-se que o resultado final da simulação é diferente se comparado à simulação realizada na seção 4.3.2, onde os robôs se com-

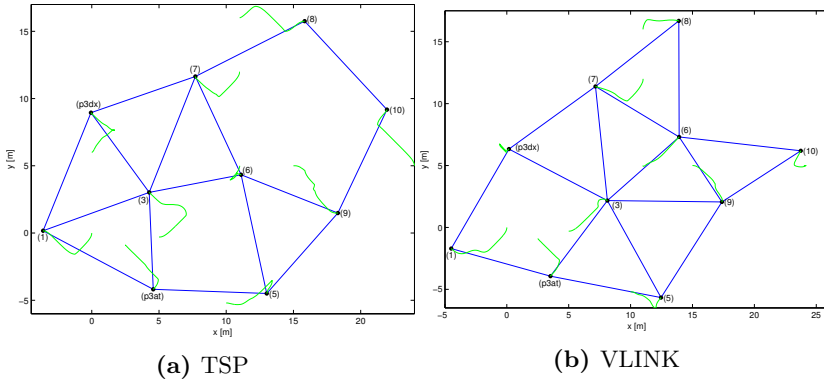


Figura 4.28: Topologia resultante após 100 iterações do controle de topologia.

portam de maneira mais uniforme e há poucas diferenças entre os resultados das duas estratégias de correção de nós críticos. Neste caso, o estado final da rede, apesar de bi-conexo, foi influenciado também pelas restrições holonômicas dos robôs, que favoreceram o comportamento invariante da abordagem de correção de nós críticos baseada no gerenciamento explícito de *links* virtuais, em detrimento da correção de nós baseada no problema do Caixeiro Viajante, que apresenta comportamento variável, de acordo com a vizinhança de cada robô.

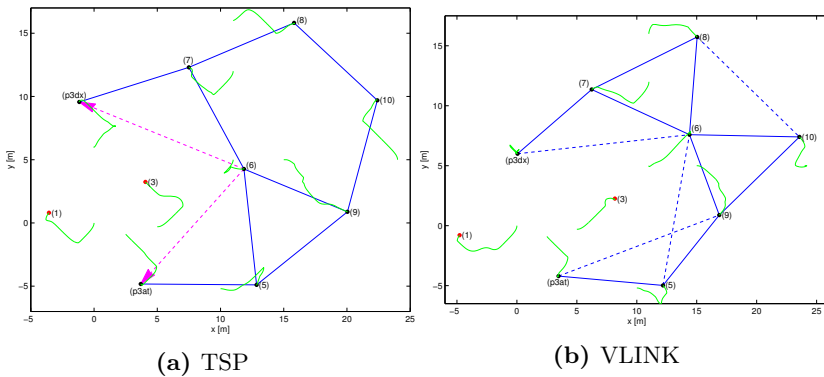


Figura 4.29: Topologia resultante após os robôs 1 e 3 perderem a comunicação com todos os demais.

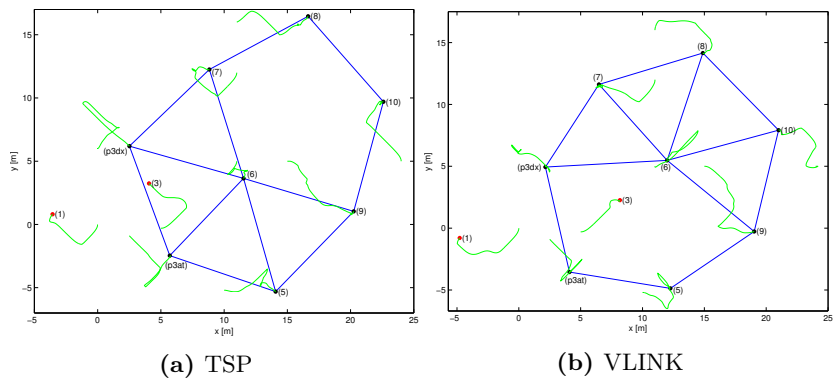


Figura 4.30: Estrutura bi-conexa restaurada após ação do controle de topologia.

Capítulo 5

Considerações Finais

Neste trabalho foram abordados aspectos relativos ao controle de topologia em redes de robôs móveis, sujeitas a alterações dinâmicas no número de robôs e em canais de comunicação entre eles. Foram analisadas as duas formas em que o controle de topologia ocorre na literatura: o controle de topologia para a minimização no número de *links* da rede de comunicação e o controle de topologia para a manutenção da conectividade em uma rede arbitrária.

Os objetivos iniciais deste trabalho foram atingidos de forma plena, e as abordagens resultantes estão de acordo com o previsto. O primeiro desses objetivos era desenvolver um método de controle de topologia para a minimização do número de *links* na rede, resultando em uma menor interferência oriunda da sobreposição de áreas de transmissão e redução do consumo energético com os dispositivos de comunicação. E foi cumprido com um algoritmo que utiliza estratégias de minimização de grafos para reduzir o número de *links* em uma rede de robôs móveis, permitindo que a interferência causada pela sobreposição de áreas de transmissão seja reduzida e o custo energético necessário para manter a rede seja minimizado através do ajuste dinâmico do raio de comunicação dos robôs na rede.

O segundo objetivo era desenvolver uma estratégia de controle de topologia para garantir a tolerância a falhas em redes de robôs móveis sujeitas à instabilidades. Este objetivo foi concluído com o desenvolvimento de dois algoritmos descentralizados para controle de topologia: o primeiro, utilizando o Problema do Caixeiro Viajante para corrigir a criticidade de nós que possam, eventualmente, desconectar a rede e um controle de movimento baseado em consenso e controle preditivo para garantir a estruturação da bi-conectividade nesta rede; e o segundo, utilizando o mesmo controle de movimento do algoritmo anterior, no

entanto, corrigindo os nós críticos através do gerenciamento explícito de ligações virtuais entre os robôs.

O terceiro objetivo era adaptar o método desenvolvido por Ordoñez et al. [17] para garantir a conectividade durante a realização do controle de topologia e mover os robôs de forma que as propriedades desejadas realmente fossem atingidas. O controle de movimento utilizado nas três abordagens para o controle de topologia basearam-se na ideia apresentada por Ordoñez et al., utilizando um algoritmo de consenso formulado como um problema de controle preditivo descentralizado para ajustar as posições dos robôs de acordo com as ligações da rede construídas pela abordagem de controle de topologia utilizada. O controle de conectividade, como foi chamado esse algoritmo, é diferente para cada estratégia do controle de topologia, apresentado um comportamento quando é realizada a minimização da topologia e outro comportamento durante a manutenção da conectividade.

O quarto e quinto objetivos visavam a realização de simulações e testes em plataformas robóticas experimentais que comprovassem a eficiência dos algoritmos propostos em gerenciar a topologia de um grupo de robôs móveis e a factibilidade de implementação destes algoritmos em aplicações reais. Tais objetivos foram cumpridos, de forma que as diversas simulações efetuadas confirmaram a eficácia dos algoritmos propostos para o controle da topologia em redes dinâmicas e os testes com robôs comerciais comprovaram a factibilidade de implementação destes algoritmos em ambientes e problemas reais.

A seguir, as principais contribuições técnicas deste trabalho são expostas, juntamente com algumas limitações e sugestões de trabalhos futuros dada estas limitações.

5.1. Produções Técnicas

Durante a realização desse trabalho, dois artigos técnicos foram elaborados e submetidos à dois eventos distintos:

- *Topology Control for Maintenance of the Connectivity in Cooperative Mobile Robot Networks*: descreve o controle de topologia para a manutenção da conectividade, utilizando como abordagem para correção de nós críticos o Problema do Caixaieiro Viajante, foi submetido ao *11th IFAC Symposium on Robot Control (SYROCO)*, que será realizado na cidade de Salvador-BA nos dias 26 a 28 de Agosto de 2015;
- *Controle de Topologia para a Manutenção da Conectividade em Sistemas Multi-Robôs Utilizando Consenso e Controle Preditivo*:

apresenta o mesmo algoritmo abordado no artigo anterior, mas insere a questão das restrições holonômicas e os experimentos com robôs comerciais; foi submetido ao *Simpósio Brasileiro de Automação Inteligente* (SBAI) a ser realizado nos dias 25 a 28 de outubro de 2015, na cidade de Natal-RN.

5.2. Limitações

Este trabalho apresenta algumas limitações em decorrência das estratégias utilizadas e dos objetivos inicialmente estabelecidos, tais como:

- Os três algoritmos para o controle de topologia propostos não se adaptam à presença de obstáculos no ambiente;
- A estratégia de controle de topologia para a manutenção da conectividade não garante esta propriedade se mais de um robô perder a comunicação com os outros simultaneamente;
- A minimização do número de *links* não garante a redução no custo energético ou a minimização de áreas de transmissão conflitantes da rede se o raio de cobertura tem um valor muito próximo a metade do raio de comunicação;
- O custo computacional para a redução no número de *links* da rede é bastante elevado graças as restrições para eliminação de sub-ciclos, o que pode prejudicar a atuação desse algoritmo em grupos que contenham muitos robôs.

5.3. Trabalhos Futuros

Com base nas limitações e deficiências dos algoritmos apresentados, existe uma gama de trabalhos que podem ser realizados aproveitando o que foi desenvolvido até aqui, entre eles, temos:

- *Adaptar o controle de topologia para detectar a presença de obstáculos no ambiente:* de forma que a topologia construída em nível lógico leve em consideração a presença de obstáculos e outras perturbações do ambiente que possam causar desconexões ou inviabilizar a estrutura planejada;
- *Melhorar o algoritmo de minimização da topologia:* o algoritmo de minimização apresentado neste trabalho é baseado em uma clássica formulação para o Problema do Caixeiro Viajante adaptada para a construção da árvore geradora mínima, não sendo totalmente otimizado, em função disto. Desta forma, pode-se utilizar outras estratégias para a minimização de maneira des-

centralizada, que obtenham um melhor comportamento para geração da MST numa rede muito densa e possuam um menor custo computacional, tais como a formulação de Miller et al. [33], adaptada para esse contexto;

- *Inserir a questão da holonomia no controle de conectividade:* a inserção de restrições holonômicas diretamente na função objetivo do controle de conectividade, eliminaria a necessidade do controle de baixo nível, que trata a velocidade processada antes de aplicá-la aos robôs.
- *Permitir aos robôs efetuarem seguimento de trajetória:* manter os aspectos apresentados relativos ao controle de topologia enquanto o grupo de robôs percorre uma determinada trajetória é uma possibilidade interessante, que adicionaria mais flexibilidade à abordagem apresentada, para atuar nos mais diversos tipos de redes;
- *Integrar o controle de topologia para a minimização da comunicação e o para a manutenção da conectividade:* de forma que os robôs, em conjunto, possam decidir qual a melhor opção para o estado da rede dado as suas posições atuais e a presença de obstáculos no ambiente em que se encontram, bem como a tarefa alvo a ser realizada;
- *Verificar as condições de estabilidade do controle de conectividade proposto:* com as alterações dos termos originais da função objetivo apresentada por Ordoñez et al., as condições de estabilidade desse algoritmo podem ter sido alteradas, e uma verificação minuciosa poderia determinar se há e quais são as novas condições de estabilidade.

Referências

- 1 CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. 1st. ed. London: Springer, 1998. 280 p.
- 2 OLFATI-SABER, R.; FAX, J. A.; MURRAY, R. M. Consensus and cooperation in networked multi-agent systems. **Proceedings of the IEEE**, v. 95, n. 1, p. 215–233, jan. 2007.
- 3 MURRAY, R. M. Recent Research in Cooperative Control of Multivehicle Systems. **Journal of Dynamic Systems, Measurement, and Control**, American Society of Mechanical Engineers, v. 129, n. 5, p. 571, set. 2007. ISSN 00220434.
- 4 LUCAS, N. P.; PANDYA, A. K.; ELLIS, R. D. Review of multi-robot taxonomy, trends, and applications for defense and space. **Unmanned Systems Technology XIV**, v. 8387, p. 1 – 10, 2012. ISSN 0277786X.
- 5 CAO, Y.; YU, W.; REN, W.; CHEN, G. An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination. **IEEE Transactions on Industrial Informatics**, v. 9, n. 1, p. 427–438, 2013.
- 6 BURKHART, M.; RICKENBACH, P. VON; WATTENHOFER, R.; ZOLLINGER, A. Does topology control reduce interference? In: **Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '04**. New York, New York, USA: ACM Press, 2004. p. 9. ISBN 1581138490.
- 7 LI, N.; HOU, J. C. FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks. In: **Proceedings of the 10th**

annual international conference on Mobile computing and networking - MobiCom '04. New York, New York, USA: ACM Press, 2004. p. 275. ISBN 1581138687.

8 AHMADI, M.; STONE, P. Keeping in Touch: Maintaining Biconnected Structure by Homogeneous Robots. In: **The Twenty-First National Conference on Artificial Intelligence (AAAI)**. [S.l.: s.n.], 2006. p. 580–85.

9 WAGENPFEIL, J.; TRACHTE, A.; HATANAKA, T.; FUJITA, M.; SAWODNY, O. A distributed minimum restrictive connectivity maintenance algorithm. **9th International Symposium on Robot Control**, v. 9, n. 1, p. 499–504, 2009.

10 CASTEIGTS, A.; NAYAK, A.; STOJMENOVIC, I. Topology control in sensor, actuator and mobile robot networks. In: NAYAK, A.; STOJMENOVIC, I. (Ed.). **Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication**. 1st. ed. Hoboken: John Wiley & Sons, Inc., 2010. cap. 7, p. 185 – 206.

11 HASSAN, M. A.; ABUHAIBA, I. S. Interference reduction in mobile ad hoc and sensor networks. **Journal of Engineering and Computer Innovations (JECI)**, v. 2, n. September, p. 138–154, 2011.

12 HALLDÓRSSON, M. M.; TOKUYAMA, T. Minimizing interference of a wireless ad-hoc network in a plane. **Theoretical Computer Science**, v. 402, n. 1, p. 29–42, jul. 2008. ISSN 03043975.

13 REN, W.; BEARD, R. W.; ATKINS, E. M. Information consensus in multivehicle cooperative control. **IEEE Control Systems Magazine**, v. 27, n. 2, p. 71–82, abr. 2007. ISSN 0272-1708.

14 REN, W.; BEARD, R. W. Consensus seeking in multiagent systems under dynamically changing interaction topologies. **IEEE Transactions on Automatic Control**, v. 50, n. 5, p. 655–661, 2005.

15 ZAVLANOS, M. M.; TANNER, H. G.; JADBABAIE, A.; PAPPAS, G. J. Hybrid Control for Connectivity Preserving Flocking. **IEEE Transactions on Automatic Control**, v. 54, n. 12, p. 2869–2875, 2009.

- 16 CASTEIGTS, A.; ALBERT, J.; CHAUMETTE, S.; NAYAK, A.; STOJMENOVIC, I. Biconnecting a Network of Mobile Robots Using Virtual Angular Forces. In: **2010 IEEE 72nd Vehicular Technology Conference - Fall**. [S.l.]: IEEE, 2010. p. 1–5. ISBN 978-1-4244-3573-9. ISSN 1090-3038.
- 17 ORDOÑEZ, B.; MORENO, U. F.; CERQUEIRA, J.; ALMEIDA, L. Generation of Trajectories Using Predictive Control for Tracking Consensus with Sensing. **Procedia Computer Science**, v. 10, p. 1094–1099, 2012. ISSN 18770509.
- 18 CORREIA, F. L. d. B.; ORDOÑEZ, B.; CERQUEIRA, J.; ALMEIDA, L.; MORENO, U. F. Controle de Veículos Autônomos em Formação com Seguimento de Referência Utilizando Consenso e RHC. **Anais do XX Congresso Brasileiro de Automática**, p. 3445–3452, 2014.
- 19 FARINELLI, A.; IOCCHI, L.; NARDI, D. Multirobot systems: A classification focused on coordination. **IEEE Transactions on Systems Man and Cybernetics Part B**, v. 34, n. 5, p. 2015–2028, 2004.
- 20 SANTIAGO-ALMEIDA, M. M. **Minidicionário Livre da Língua Portuguesa**. São Paulo: Hedra, 2011. ISBN 9788577152346.
- 21 GODSIL, C.; ROYLE, G. **Algebraic graph theory**. New York: Verlag, 2001. 439 p.
- 22 MOALLEMI, C. C.; ROY, B. V. Consensus propagation. **IEEE Transactions on Information Theory**, v. 52, n. 11, p. 4753–4766, nov. 2006.
- 23 NOCEDAL, J.; WRIGHT, S. J. **Numerical Optimization**. New York: Springer, 1999. 634 p. ISBN 0387987932.
- 24 BOYD, S.; VANDENBERGHE, L. **Convex Optimization**. Cambridge: Cambridge University Press, 2004. 716 p. ISBN 9780511804441.
- 25 LIN, J.; MORSE, A.; ANDERSON, B. The multi-agent rendezvous problem. In: **42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)**. [S.l.]: IEEE, 2003. v. 2, p. 1508–1513. ISBN 0-7803-7924-1. ISSN 0191-2216.

- 26 DIMAROGONAS, D. V.; KYRIAKOPOULOS, K. J. On the Rendezvous Problem for Multiple Nonholonomic Agents. **IEEE Transactions on Automatic Control**, v. 52, n. 5, p. 916–922, maio 2007. ISSN 0018-9286.
- 27 SANTI, P. **Topology Control in Wireless Ad Hoc and Sensor Networks**. 1st. ed. Chichester: John Wiley & Sons, Inc., 2005. 252 p. ISBN 9780470094532.
- 28 JOHANSSON, T.; CARR-MOTYCKOVÁ, L. Reducing interference in ad hoc networks through topology control. In: **Proceedings of the 2005 joint workshop on Foundations of mobile computing - DIALM-POMC '05**. New York, New York, USA: ACM Press, 2005. p. 17. ISBN 1595930922.
- 29 CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 3rd. ed. Cambridge: The MIT Press, 2009. 1312 p. ISSN 15580768. ISBN 0262033844.
- 30 LI, N.; HOU, J. C.; SHA, L. Design and analysis of an MST-based topology control algorithm. In: **IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)**. [S.l.]: IEEE, 2003. v. 3, p. 1702–1712. ISBN 0-7803-7752-4. ISSN 0743-166X.
- 31 HU, B.; LEITNER, M.; RAIDL, G. R. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. **Journal of Heuristics**, v. 14, n. 5, p. 473–499, out. 2007. ISSN 1381-1231.
- 32 DANTZIG, G. B.; FULKERSON, D. R.; JOHNSON, S. M. Solution of a Large-Scale Traveling-Salesman Problem. In: JÜNGER, M.; LIEBLING, T. M.; NADDEF, D.; NEMHAUSER, G. L.; PULLEYBLANK, W. R.; REINELT, G.; RINALDI, G.; WOLSEY, L. A. (Ed.). **50 Years of Integer Programming 1958-2008**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. cap. 1, p. 7–28. ISBN 978-3-540-68274-5.
- 33 MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. Integer Programming Formulation of Traveling Salesman Problems. **Journal of the ACM**, ACM, v. 7, n. 4, p. 326–329, out. 1960. ISSN 00045411.

- 34 PATAKI, G. Teaching Integer Programming Formulations Using the Traveling Salesman Problem. **SIAM Review**, v. 45, n. 1, p. 116–123, 2003.
- 35 CORREIA, F. L. d. B. **Controle de Formação com Seguimento de Referência para Grupo de Veículos Autônomos Utilizando Consenso e RHC**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2014.
- 36 FIACCHINI, M.; MORARESCU, I.-C. Convex Conditions on Decentralized Control for Graph Topology Preservation. **IEEE Transactions on Automatic Control**, v. 59, n. 6, p. 1640–1645, jun. 2014. ISSN 0018-9286.
- 37 MENGER, K. Zur allgemeinen Kurventheorie. **Fundamenta Mathematicae**, Institute of Mathematics Polish Academy of Sciences, v. 10, n. 1, p. 96–115, 1927. ISSN 0016-2736.
- 38 PADBERG, M.; SUNG, T.-Y. An analytical comparison of different formulations of the travelling salesman problem. **Mathematical Programming**, v. 52, n. 1-3, p. 315–357, maio 1991. ISSN 0025-5610.
- 39 LETCHFORD, A. N.; NASIRI, S. D.; THEIS, D. O. Compact formulations of the Steiner Traveling Salesman Problem and related problems. **European Journal of Operational Research**, v. 228, n. 1, p. 83–92, jul. 2013. ISSN 03772217.
- 40 DAS, S.; LIU, H.; NAYAK, A.; STOJMENOVIĆ, I. A localized algorithm for bi-connectivity of connected mobile robots. **Telecommunication Systems**, v. 40, n. 3-4, p. 129–140, 2009. ISSN 1018-4864.
- 41 FIEDLER, M. Algebraic connectivity of graphs. **Czechoslovak Mathematical Journal**, v. 23, p. 298–305, 1973.
- 42 BARAS, J. S.; HOVARESHTI, P. Efficient and robust communication topologies for distributed decision making in networked systems. In: **Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference**. [S.l.]: IEEE, 2009. p. 3751–3756. ISBN 978-1-4244-3871-6. ISSN 0191-2216.

- 43 KLEIN, D. J.; RANDIĆ, M. Resistance distance. **Journal of Mathematical Chemistry**, v. 12, n. 1, p. 81–95, dez. 1993. ISSN 0259-9791.
- 44 ELLENS, W.; SPIEKSMAN, F. M.; Van Mieghem, P.; JAMAKOVIC, A.; KOUIJ, R. E. Effective graph resistance. **Linear Algebra and Its Applications**, v. 435, p. 2491–2506, 2011. ISSN 00243795.
- 45 ABREU, N. M. M. de. Old and new results on algebraic connectivity of graphs. **Linear Algebra and its Applications**, v. 423, n. 1, p. 53–73, maio 2007. ISSN 00243795.
- 46 GUIXIAN, T.; TINGZHU, H.; SHUYU, C. Bounds on the Algebraic Connectivity of Graphs. **Advances in Mathematics**, v. 41, n. 2, p. 217–224, 2012.
- 47 O’KANE, J. M. **A Gentle Introduction to ROS**. 1st. ed. Columbia: Independently published, 2014. 153 p. ISBN 9781492143239.
- 48 GOEBEL, R. P. **ROS By Example**. 1st. ed. [S.l.]: Independently published, 2012. 223 p. ISBN 5800085311092.

Apêndice A

Simulador para o Controle de Topologia

Nesta seção o simulador desenvolvido para os testes com os algoritmos de controle de topologia é detalhado quanto ao seu funcionamento e estrutura básica. O simulador foi desenvolvido em C++ e compilado com o *GNU Compiler Collection (GCC)*¹ que é um compilador bastante popular disponível para as principais plataformas computacionais. Todos os códigos fontes utilizados para a construção do simulador aqui apresentado possuem licença aberta e estão disponíveis em <https://github.com/SIDNEYRDC/topology_control>.

A figura A.1 indica como o simulador se comporta durante a execução de um dos algoritmos de controle de topologia. Durante a execução do simulador, verifica-se a utilização do ambiente provido pelo ROS para controlar robôs externos e caso esta opção seja confirmada, o simulador deve criar um nó de comunicação utilizando a biblioteca *ros_interface*, alocar e subscrever os tópicos pertinentes à esses robôs, coletar a posição e enviar a velocidade para cada um deles. Neste caso, nota-se que os robôs externos são apenas um reflexo dos robôs virtuais criados no simulador, que efetivamente interagem entre si durante a troca de mensagens simulada, resultando em um algoritmo centralizado, entretanto, que simula um comportamento descentralizado.

A ação de manipulação da matriz de adjacência e o controle de conectividade são definidos conforme o tipo de controle de topologia realizado. Se o controle de topologia é para a minimização no número de *links* e redução do consumo de energia ($TC = 1$), executa-se o algoritmo de minimização de links descrito na equação (3.3a), assim como o controle de conectividade representado pela equação (3.4). Se o controle de topologia deve manter a conectividade da rede, executa-se um dos dois métodos utilizados para a correção de nós críticos ($TC = 2$ para

¹<<https://gcc.gnu.org/>>

TSP ou TC = 3 para VLINK) e o controle de conectividade descrito na equação (3.10).

A seguir, é apresentado um exemplo do arquivo de configuração utilizado como entrada para o simulador durante os experimentos realizados com robôs comerciais na seção 4.4.2. Linhas que iniciam com # são comentários.

```
1 # Topology configuration file
2
3 [simulation-configuration]
4 # Number of iterations
5 n_iter: 500
6
7 # Data capture frequency
8 dc: 1
9
10 # Simulation step
11 dt: 0.1
12
13 # Prediction horizon
14 ph: 5
15
16 # Velocity variation gain
17 gamma: 0.5
18
19 # Maximum velocities
20 vx_max: 1
21 vy_max: 1
22 va_max: 1
23
24 # Minimum velocities
25 vx_min: -1
26 vy_min: -1
27 va_min: -1
28
29 # Information scope
30 # 0=local 1=global
31 info_scope: 0
32
33 # Topology control type
34 # 0=off 1=mst 2=tsp 3=vlink
35 opt_type: 2
36
37 # Enable communication radius constraint
38 # 0=off 1=on
39 com_c: 0
40
41 # Enable security radius constraint
42 # 0=off 1=on
43 sec_c: 1
44
```



```
45 # Enable ROS interface
46 # 0=off 1=on
47 ros: 1
48
49 [robot-positions]
50 # bot: x y
51 0: 0 0
52 1: 0 6
53 2: 5 -0.3
54 3: 2.5 -0.9
55 4: 10.0 -5.2
56 5: 11.0 5.0
57 6: 11.0 12.0
58 7: 11.0 16.0
59 8: 15.0 5.0
60 9: 24.0 5.0
61
62 [robot-ranges]
63 # bot: R r
64 0: 10.0 4.50
65 1: 10.0 4.50
66 2: 10.0 4.50
67 3: 10.0 4.50
68 4: 10.0 4.50
69 5: 10.0 4.50
70 6: 10.0 4.50
71 7: 10.0 4.50
72 8: 10.0 4.50
73 9: 10.0 4.50
74
75 [external-robots]
76 # bot: type 'name'
77 # 1=real 2=stage 3=turtle
78 0: 2 'robot_0'
79 1: 1 'p3dx'
80 2: 2 'robot_2'
81 3: 1 'p3at'
82 4: 2 'robot_4'
83 5: 2 'robot_5'
84 6: 2 'robot_6'
85 7: 2 'robot_7'
86 8: 2 'robot_8'
87 9: 2 'robot_9'
88
89 [timeout]
90 # bot: timeout (iter)
91 0: 160
92 2: 140
```

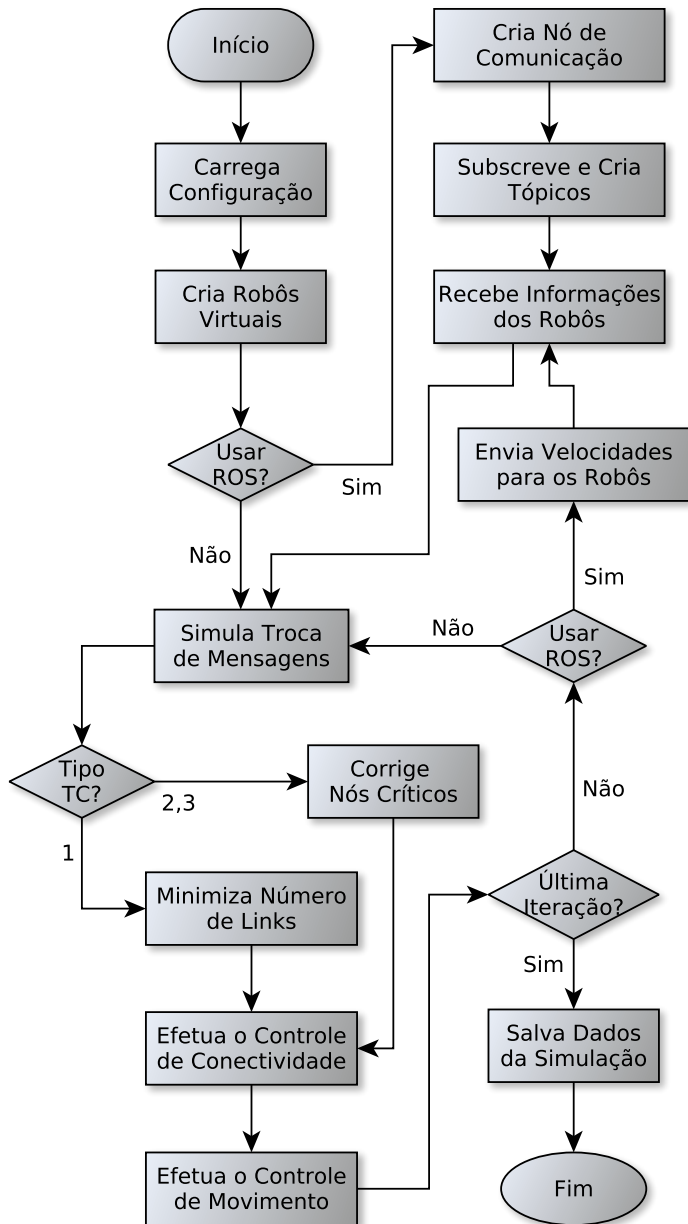


Figura A.1: Fluxo de execução do simulador.

Apêndice B

Controle de Movimento de “Baixo Nível”

O controle de conectividade proposto anteriormente, considera somente robôs com comportamento não holonômico, ou seja, que podem se mover livremente em qualquer direção. No entanto, para executar testes em robôs reais é preciso considerar as restrições holonômicas que tais robôs, por ventura, apresentem.

Nesta seção é apresentado um controle de movimento que considera as restrições holonômicas dos robôs comerciais utilizados nos experimentos. Tais robôs são do tipo diferencial, o que indica que eles possuem restrições de movimento que limitam sua ação aos eixos x e z , somente. Desta forma, o controle de baixo nível deve traduzir os sinais de velocidade omnidirecionais, provenientes do controle de conectividade, em velocidade linear no eixo x e velocidade angular no eixo z (yaw), para que os robôs possam efetuar o comportamento desejado pelo controle de topologia, ao mesmo tempo que respeitam suas restrições holonômicas.

O controlador de baixo nível proposto é um algoritmo que ajusta o ângulo do robô para o ângulo desejado e aplica uma velocidade no eixo x proporcional ao erro em relação à referência. Na prática, esse controle provê um comportamento onde o robô para, ajusta a sua direção em relação à referência e então se move para frente.

A ação de controle linear e angular é definida conforme as seguintes equações:

$$\begin{aligned}\dot{\theta} &= K^\theta v_{max}^\theta \\ \dot{x} &= K^x v_{max}^x (1 - |K^\theta|)\end{aligned}$$

onde v_{max}^θ e v_{max}^x são os limites superiores de velocidade angular para o eixo z e linear para o eixo x , respectivamente; e K^θ e K^x são variáveis de proporcionalidade, definidas em função do erro angular e de posição,

como segue:

$$K^\theta = \begin{cases} \Delta\theta/\delta_{max}^\theta & \text{if } \delta_{min}^\theta \leq |\Delta\theta| \leq \delta_{max}^\theta \\ 1 & \text{if } |\Delta\theta| > \delta_{max}^\theta \text{ and } \Delta\theta > 0 \\ -1 & \text{if } |\Delta\theta| > \delta_{max}^\theta \text{ and } \Delta\theta < 0 \\ 0 & \text{if } |\Delta\theta| < \delta_{min}^\theta \end{cases}$$

$$K^x = \begin{cases} \sqrt{\Delta x^2 + \Delta y^2}/\delta_{max}^x & \text{if } \delta_{min}^x \leq \Delta x \leq \delta_{max}^x \\ 0 & \text{if } \Delta x < \delta_{min}^x \\ 1 & \text{if } \Delta x > \delta_{max}^x \end{cases}$$

com $\Delta\theta = \theta_1 - \theta_0$, $\Delta x = |x_1 - x_0|$ e $\Delta y = |y_1 - y_0|$ sendo o erro angular e linear em relação à referência e δ_{min} e δ_{max} o valor mínimo e máximo para esses erros, respectivamente.

A figura B.1 mostra como os valores para o erro angular e linear são obtidos em função da disposição do robô em relação ao referencial inercial: d é o ponto desejado, θ_0 é o ângulo do robô em relação ao eixo X inercial para o instante atual e θ_1 é o ângulo desejado em relação ao eixo X .

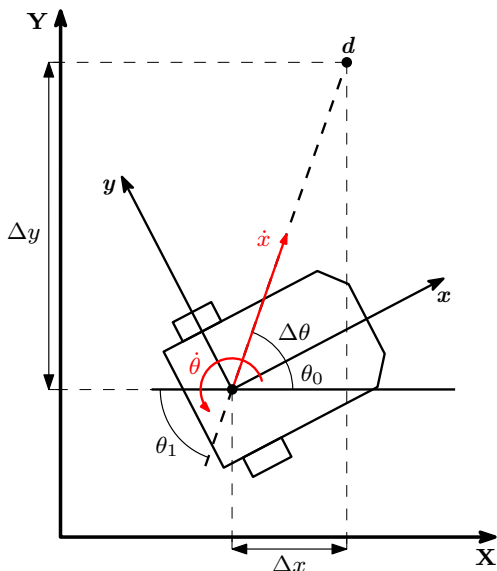


Figura B.1: Determinação do erro angular e linear em relação à referência.

Apêndice C

Interface de Comunicação com o ROS

A interface de comunicação com o ROS fornece uma série de funções que podem ser utilizadas para acessar o ambiente de comunicação provido por ele, permitindo que cada robô simulado possa publicar e receber informações de diversos tópicos diferentes de uma forma simplificada.

A maior vantagem de se utilizar uma interface de integração, nesse caso, é que não existe a necessidade da implementação explícita de funções do ROS dentro do simulador, ao mesmo tempo que a estrutura desse ambiente é encapsulada. O código fonte da implementação da biblioteca de integração é de licença aberta e está disponível online via [github](#)¹ e as seguintes funções são implementadas pela biblioteca:

- **add_node(id, type, target, init_pose)**: permite a inclusão de um novo robô na interface, **type** define o tipo de robô, para robôs que utilizam a biblioteca ROSARIA o tipo é 1, **target** é o nome do nó a que se deseja enviar e receber informação;
- **node_send(id, msg)**: permite o envio da velocidade para um nó existente na interface e cujo identificador único é **id**;
- **node_receive(id)**: retorna a posição de um robô da interface com o identificador único igual a **id**;
- **clock(time)**: define o passo da simulação.

A figura C.1 mostra o nó representando a interface de comunicação e os tópicos apresentados pelos dois robôs utilizados nos experimentos realizados na seção 4.4. O tópico **pose** fornece a informação da odometria do nó associado, enquanto que o tópico **cmd_vel** é utilizado para enviar o comando de velocidade para o robô a ele associado. Para mais detalhes a cerca da estrutura de comunicação provida pelo ROS, consultar [47] e [48].

¹<https://github.com/SIDNEYRDC/ros_interface.git>

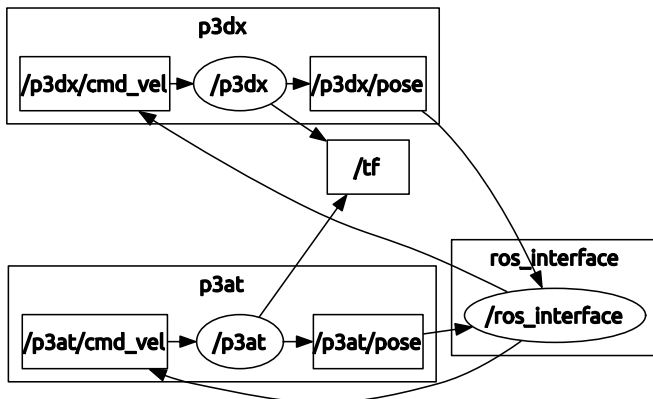


Figura C.1: Grafo de comunicação entre a *ros_interface* e os robôs.

A seguir o código fonte principal da biblioteca escrito em C++ é apresentado:

```

1  /*****
2  * ROS Communication Interface declaration
3  * Written by Sidney RDC, 2015.
4  * Last Change:2015 Abr 09 01:29:37
5  *****/
6
7  #ifndef ROS_INTERFACE_H
8  #define ROS_INTERFACE_H
9
10 #include <string>
11 #include <vector>
12
13 // Type of robots supported in the ROS
14 #define T_REAL 1
15 #define T_STAGE 2
16 #define T_TURTLE 3
17
18 // Position data type
19 struct position {
20     double x;        // position in x
21     double y;        // position in y
22     double z;        // position in z
23     double roll;     // lateral axis (X)
24     double pitch;    // longitudinal axis (Y)
25     double yaw;      // vertical axis (Z)
26 };
27
28 // Velocity data type
29 struct velocity {

```

```

30     double x;           // velocity in x
31     double y;           // velocity in y
32     double z;           // velocity in z
33     double roll;        // lateral axis angular velocity
34     double pitch;       // longitudinal axis angular velocity
35     double yaw;         // vertical axis angular velocity
36 };
37
38 // Node class to ROS interface
39 class ros_interface {
40 public:
41     // Default initializer
42     ros_interface();
43
44     // Initialize the ROS communication interface
45     ros_interface(int argc, char** argv, std::string node_name);
46
47     // Destructor
48     ~ros_interface();
49
50     // Add nodes to the interface
51     void add_node(int id, int type, std::string target, position
        init_pose);
52
53     // A specific node send a message
54     void node_send(const int id, velocity msg);
55
56     // A specific node receive a message
57     position node_receive(const int id);
58
59     // Data capture frequency
60     void clock(const float dt);
61
62     // Return true if ros_interface is running
63     bool ros_ok();
64 private:
65     // Nodes pointer array
66     std::vector<void*> nodes_ptr;
67
68     // Verify if the node id exist in nodes array and return its
        index
69     bool check_node(const int id, unsigned int &index);
70 };
71
72 #endif

```


Apêndice D

Gerador de Grafos Aleatórios

Nesta seção é apresentado o código fonte para o gerador de grafos aleatórios conexos que foram submetidos aos testes com os algoritmos de controle de topologia. Este programa consiste em um script em Python que utiliza funções da biblioteca *NetworkX*¹ para gerar grafos conectados em função da distância entre seus nodos, os chamados grafos geométricos.

Para executar o gerador de grafos pela linha de comando utiliza-se os seguintes parâmetros:

```
$ python rand_graph.py [-h] -n N -r R [-s S] [-e E] [-f F]
```

onde *N* é o número de nós que o grafo irá ter, *R* é o raio de comunicação, *S* é a área onde os nodos serão gerados, *E* é o número de vértices que o grafo apresentará e *F* é um multiplicador de coordenadas para ampliar a escala do grafo gerado. Como resultado da execução deste script, é gerado o arquivo de configuração para o simulador no formato apresentado no apêndice A e um arquivo .pdf contendo o grafo resultante.

O código fonte do script para geração dos grafos é apresentado a seguir:

```

1 #####
2 # Random Geometric Graph Generator
3 # Written by SIDNEY RDC using NetworkX library.
4 # Last Change: 2015 Mar 24 17:34:31
5 #####
6
7 import argparse
8 import numpy as np
9 from scipy import spatial
10 import networkx as nx
11 import matplotlib.pyplot as plt
12 import math, random, sys

```

¹<<https://networkx.github.io/>>

```

13
14 # generate a random geometric graph in a determined area
15 def random_geometric_graph(n, radius, size=1, ne=None, dim=2,
16     pos=None):
17     G=nx.Graph()
18     G.name="Random Geometric Graph"
19     G.add_nodes_from(range(n))
20     if pos is None:
21         # random positions
22         for n in G:
23             G.node[n]['pos']=[random.random() * size for i in
24                 range(0,dim)]
25     else:
26         nx.set_node_attributes(G,'pos',pos)
27     # connect nodes within "radius" of each other
28     # n^2 algorithm, could use a k-d tree implementation
29     nodes = G.nodes(data=True)
30     e=0
31     while nodes:
32         u,du = nodes.pop()
33         pu = du['pos']
34         for v,dv in nodes:
35             pv = dv['pos']
36             d = sum(((a-b)**2 for a,b in zip(pu,pv)))
37             if d <= radius**2:
38                 G.add_edge(u,v)
39                 e = e + 1
40
41         if e == ne:
42             break
43
44     if e == ne:
45         break
46     return G
47
48 if __name__ == "__main__":
49     parser = argparse.ArgumentParser(description='rand_graph.py
50     - Random Geometric Graph Generator')
51     parser.add_argument('-n', '--nodes', dest='nodes', type=int,
52         required=True, help='Number of nodes on the graph')
53     parser.add_argument('-r', '--radius', dest='radius', type=
54         float, required=True, help='Radius of communication')
55     parser.add_argument('-s', '--size', dest='size', type=float,
56         required=False, help='Size area of random generate')
57     parser.add_argument('-e', '--edges', dest='edges', type=int,
58         required=False, help='Number of edges on the graph')
59     parser.add_argument('-f', '--factor', dest='scale', type=int
60         , required=False, help='Scale factor of size')
61     args = parser.parse_args()
62
63     # Verify the scale factor
64     if args.scale > 1:

```

```

57         scale = args.scale
58     else:
59         scale = 1
60
61     # Verify the size
62     if args.size > 1:
63         size = args.size
64     else:
65         size = 1
66
67     # Verify the number of edges constraint
68     if ((args.edges < args.nodes - 1) or (args.edges > (args.
69         nodes * (args.nodes - 1) / 2))) and args.edges != None:
70         print 'ERROR: The e must be: n-1 <= e <= n*(n-1)/2'
71         exit(1)
72
73     # Generate a connected graph
74     print 'Generating a connected graph...'
75     while True:
76         G = random_geometric_graph(args.nodes, args.radius, size,
77         args.edges)
78         pos = nx.get_node_attributes(G, 'pos')
79         if nx.is_connected(G) == True:
80             print 'is connected: %s' % (nx.is_connected(G))
81             break
82
83     # Configuration output file
84     f = open('topology.conf', 'w')
85
86     # Initial headers
87     f.write('# Topology configuration file\n')
88
89     f.write('\n[simulation-configuration]\n')
90     f.write('# Number of iterations\n')
91     f.write('n_iter: 100\n')
92
93     f.write('\n# Data capture frequency\n')
94     f.write('dc: 1\n')
95
96     f.write('\n# Information scope')
97     f.write('\n# 0=local 1=global\n')
98     f.write('info_scope: 0\n')
99
100    f.write('\n# Topology control type')
101    f.write('\n# 0=off 1=mst 2=tsp 3=vlink\n')
102    f.write('opt_type: 0\n')
103
104    f.write('\n# Enable reference pass')
105    f.write('\n# 0=off 1=on\n')
106    f.write('ref_pass: 0\n')
107
108    f.write('\n# Enable communication radius constraint')

```

```

107     f.write('\n# 0=off 1=on\n')
108     f.write('com_c: 0\n')
109
110     f.write('\n# Enable security radius constraint')
111     f.write('\n# 0=off 1=on\n')
112     f.write('sec_c: 0\n')
113
114     f.write('\n# Enable ROS interface')
115     f.write('\n# 0=off 1=on\n')
116     f.write('ros: 0\n')
117
118     # Headers for position
119     f.write('\n[robot-positions]\n')
120     f.write('# bot: x y\n')
121
122     # Write positions on archive file
123     for n in pos:
124         f.write('%d: %f %f\n' % (n, pos[n][0]*scale, pos[n][1]*
125             scale))
126
127     # Headers for velocity
128     f.write('\n[robot-velocities]\n')
129     f.write('# bot: vx vy\n')
130
131     # Headers for radius
132     f.write('\n[robot-ranges]\n')
133     f.write('# bot: R r\n')
134
135     # Write radius on archive file
136     for n in xrange(0, args.nodes):
137         f.write('%d: %.1f 0.0\n' % (n, args.radius*scale))
138
139     # Headers to external robots
140     f.write('\n[external-robots]\n')
141     f.write('# bot: type \'name\'\n')
142     f.write('# 1=real 2=stage 3=turtle\n')
143
144     # Headers to reference
145     f.write('\n[reference]\n')
146     f.write('# bot: x y\n')
147
148     # Headers to timeout configuration
149     f.write('\n[timeout]\n')
150     f.write('# bot: timeout (iter)\n')
151
152     # Close file
153     f.close()
154
155     # Draw graph
156     nx.draw(G, pos)
157
158     # Draw labels

```

```
158 nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans
    -serif')
159
160 # Draw nodes
161 nx.draw_networkx_nodes(G, pos, node_color='#E0E0E0')
162
163 # Show and save figure
164 plt.savefig("graph.pdf")
165 plt.show()
```

