ESTRATÉGIA GERAL DE ONDAS DE MENSAGENS PARA DESENVOLVIMENTO DE TAREFAS EM ENXAMES DE ROBÔS

Luneque Del Rio de Souza e Silva Junior

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
Nadia Nedjah

Rio de Janeiro
Agosto de 2017

ESTRATÉGIA GERAL DE ONDAS DE MENSAGENS PARA DESENVOLVIMENTO DE TAREFAS EM ENXAMES DE ROBÔS

Luneque Del Rio de Souza e Silva Junior

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof.ª Nadia Nedjah, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Carlos Eduardo Pedreira, Ph.D.

Prof. Leandro dos Santos Coelho, D.Sc.

Prof. Diego Moreira de Araújo Carvalho, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2017

Silva Junior, Luneque Del Rio de Souza e

Estratégia Geral de Ondas de Mensagens para Desenvolvimento de Tarefas em Enxames de Robôs/Luneque Del Rio de Souza e Silva Junior. – Rio de Janeiro: UFRJ/COPPE, 2017.

XVIII, 134 p.: il.; 29, 7cm.

Orientadores: Felipe Maia Galvão França

Nadia Nedjah

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 108 – 119.

1. Robótica de Enxame. 2. Algoritmos Distribuídos. 3. Tarefas Coletivas. 4. Transmissão de Mensagens. I. França, Felipe Maia Galvão *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

"Inúmeras atividades ainda executadas hoje por mãos humanas serão realizadas por autômatos. Neste momento, cientistas trabalhando nos laboratórios de universidades americanas estão tentando desenvolver o que tem sido descrito como "máquinas de pensar". Eu antecipei esse desenvolvimento."
"Na verdade, eu construí "robôs". Hoje o robô é um fato aceitado, mas o princípio ainda não foi desenvolvido o suficiente. No século 21, o robô vai tomar o lugar que o trabalho escravo ocupou na antiga civilização. Não existe motivo nenhum pelo o qual grande parte disso não se concretize em menos de um século, liberando a humanidade para exercer suas aspirações maiores."

Nikola Tesla,
"A machine to end war", em Liberty Magazine (Fevereiro de 1937)

Agradecimentos

Agradeço a meus orientadores, não apenas pela constante supervisão deste trabalho, mas também pela pela compreensão em eventuais dificuldades enfrentadas no decorrer destes anos. Muito obrigado por tudo.

Agradeço à minha família pelo apoio incondicional.

Agradeço aos professores e funcionários do Programa de Engenharia de Sistemas e Computação pelo suporte na realização deste trabalho.

Agradeço aos colegas de estudo pelos conselhos, pelo auxílio na soluções para problemas diversos, ou mesmo pela conversa nos momentos livres.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ESTRATÉGIA GERAL DE ONDAS DE MENSAGENS PARA DESENVOLVIMENTO DE TAREFAS EM ENXAMES DE ROBÔS

Luneque Del Rio de Souza e Silva Junior

Agosto/2017

Orientadores: Felipe Maia Galvão França
Nadia Nedjah

Programa: Engenharia de Sistemas e Computação

Esta tese propõe uma metodologia geral para o desenvolvimento sistemático de tarefas em enxames de robôs. A robótica de enxame investiga comportamentos coletivos em sistemas de múltiplos robôs. O estudo e desenvolvimento de sistemas robóticos, de uma forma geral, tem por objetivo auxiliar o ser humano em trabalhos difíceis, pesados ou insalubres. Em contraste com sistemas robóticos baseados em um único robô, o uso de enxames robóticos ainda não se traduz em aplicações do cotidiano. Uma possível explicação para este fato está na dificuldade existente no desenvolvimento de controladores distribuídos para tais aplicações. Muitas pesquisas estão direcionadas para a criação de técnicas que auxiliem na elaboração de tarefas coletivas. Na estratégia de ondas de mensagens proposta nesta tese, o enxame opera como uma rede de robôs que interagem por meio de comunicação sem fio. Um algoritmo distribuído de ondas de mensagens foi formulado para uso em enxames de robôs, servindo como base para o controle de diferentes tarefas. Neste trabalho foram implementados os *softwares* de controle para as tarefas conhecidas como básicas, que dizem respeito ao recrutamento, alinhamento, navegação, agregação e dispersão. O funcionamento destas tarefas foi avaliado em um enxame simulado de robôs. Experimentos mostram que o tempo necessário para conclusão destas tarefas está relacionado com o tamanho do enxame, a propagação das mensagens e características próprias de cada tarefa. A implementação destas tarefas demonstra a generalidade da estratégia proposta, indicando seu potencial uso em outras tarefas coletivas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

GENERAL STRATEGY OF WAVE OF MESSAGES FOR TASKS DEVELOPMENT IN SWARM ROBOTICS

Luneque Del Rio de Souza e Silva Junior

August/2017

Advisors: Felipe Maia Galvão França
Nadia Nedjah

Department: Systems Engineering and Computer Science

This thesis proposes a general methodology for a systematic approach of tasks in swarms of robots. Swarm robotics investigates collective behaviors in multiple robot systems. The study and development of robotic systems, in general, aims to assist the human being in difficult, heavy or unhealthy jobs. In contrast to robotic systems based on a single robot, the use of robotic swarms still does not translate into everyday applications. One possible explanation is the difficulty in implementing distributed controllers for such applications. Many research works are directed towards the development of techniques to assist in the elaboration of collective tasks. In the message wave strategy proposed in this work, the swarm operates as a network of robots, which interact through wireless communication. A distributed message wave algorithm is designed to be used in robot swarms, serving as the basis for the control of different tasks. In this thesis, the control software for commonly nown as basic tasks of recruitment, alignment, navigation, aggregation, and dispersion, are implemented. The operation of these tasks is evaluated through a simulated swarm of robots. Experiments show that the time required to complete these tasks is dependent on the size of the swarm, the propagation of messages and the characteristics of each task. The design and execution of such tasks serve as proof of concept for the proposed strategy, indicating its generality so as to be used to implement other collective tasks.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos

C	Canal de comunicação entre dois processos, p. 39
E	Fator de redimensionamento do enxame, p. 73
P	Processo em um sistema de computação distribuída, p. 36
R_i	Robô i em um enxame, p. 41
S_{AD}	Estado no algoritmo de agregação/dispersão, p. 71
S_A	Estado no algoritmo de alinhamento, p. 56
S_R	Estado no algoritmo de recrutamento, p. 56
α	Grau de alinhamento entre dois robôs, p. 92
η_G	Grupo de robôs recrutado pelo inicializador, p. 81
$\overline{\tau_D}$	Tempo médio de dispersão, p. 99
$\overline{\tau_G}$	Tempo médio de agregação, p. 99
$\overline{\tau_R}$	Tempo médio de recrutamento, p. 83
ρ_C	Número de robôs no caminho crítico, p. 89
τ_A	Tempo de alinhamento, p. 88
τ_D	Tempo de dispersão, p. 99
τ_G	Tempo de agregação, p. 99
τ_R	Tempo de recrutamento, p. 81
al_f	Variável lógica de alinhamento dos robôs filhos, p. 58
al_p	Variável lógica de alinhamento do robô pai, p. 58
$d'_{x,y}$	Distância entre um vizinho e o raio máximo de detecção, p. 75

d_F	Distância de um robô filho em relação ao pai, p. 73
$d_{x,y}$	Posição do vizinho no sistema de coordenadas local, p. 74
$d_{x,y}(sep)$	Distância de separação, p. 75
$d_{x,y}(coe)$	Distância de coesão, p. 74
id	Identificador, p. 40
msg_i	Mensagem do tipo i, p. 39
$num\_vizinhos$	Quantidade de vizinhos detectados, p. 74
$raio\_sensor$	Alcance máximo do sistema de detecção, p. 74
red_f	Variável lógica de redimensionamento dos robôs filhos, p. 71
red_p	Variável lógica de redimensionamento, p. 71
v_i	Velocidade associada com a regra de *flocking* i, p. 68
v_{ccw}	Velocidade no sentido anti-horário, p. 75
v_{cw}	Velocidade no sentido horário, p. 75
$vizinho_i$	Robô i vizinho do robô que realiza a detecção , p. 74
w_i	Peso da regra de *flocking* i, p. 68

Lista de Abreviaturas

2D	Bidimensional, p. 17
3D	Tridimensional, p. 20
ABC	*Artificial Bee Colony*, p. 13
ACK	*Acknowledgment*, p. 43
ACO	*Ant Colony Optimization*, p. 12
BSA	*Backtracking Search Optimization Algorithm*, p. 13
E/S	Entrada e Saída, p. 38
FIFO	*First In, First Out*, p. 39
FPGA	*Field Programmable Gate Array*, p. 21
FSM	*Finite State Machine*, p. 44
GA	*Genetic Algorithm*, p. 12
GPS	*Global Positioning System*, p. 32
IR	*Infrared*, p. 15
LED	*Light Emitting Diode*, p. 44
MRS	*Multi-Robot Systems*, p. 8
PCB	*Printed Circuit Boards*, p. 20
PIF	*Propagation of Information with Feedback*, p. 48
PSO	*Particle Swarm Optimization*, p. 12
RF	*Radio Frequency*, p. 15
ROM	*Read Only Memory*, p. 41

SLAM *Self Localization and Mapping*, p. 28

SoC *System-on-Chip*, p. 21

V-REP *Virtual Robot Experimentation Platform,*, p. 17

Capítulo 1

Introdução

A robótica de enxame [1] é uma área multidisciplinar que visa o estudo e desenvolvimento de sistemas capazes de realizar tarefas de forma coletiva. Um enxame é composto por agentes móveis que interagem entre si e com o ambiente. A inspiração para a robótica de enxame é biológica. Várias espécies na natureza estão organizadas em sociedades, como colônias, colmeias e cardumes [2]. Grupos de indivíduos apresentam comportamentos coletivos, que podem emergir a partir de ações realizadas de forma deliberada pelos membros, ou a partir da coordenação entre os mesmos. A compreensão de como estes comportamentos ocorrem em sistemas naturais permite sua replicação em sistemas artificiais. Um desafio no estudo da robótica de enxame é entender como estes comportamentos podem ser convertidos em aplicações [3].

Os estudos de robótica possuem várias abordagens interdependentes. O projeto mecânico de um robô especifica as características físicas do mesmo, como sua capacidade de movimentação e manipulação de objetos. O projeto de *hardware* eletrônico define a capacidade de processamento, bem como os sistemas de detecção e comunicação, que permitirão a interação entre robôs. O projeto do *software* de controle deve interpretar as observações feitas do ambiente externo, permitindo que cada robô reaja à presença de outros robôs ou objetos, de forma que um trabalho coletivo seja realizado.

1.1 Desenvolvimento de Tarefas Coletivas

O estudo de tarefas coletivas é uma das áreas de pesquisa em robótica de enxame. Uma tarefa é uma operação realizada por um robô. Por exemplo, o deslocamento por uma determinada distância ou período de tempo pode ser interpretado como uma tarefa individual de um robô. As tarefas coletivas ocorrem quando a operação individual depende não somente daquilo realizado pelo robô, mas também pelo realizado por outros robôs. Um exemplo de tarefa coletiva é o deslocamento de dois robôs, com a movimentação só sendo interrompida após ambos terem percorrido

certa distância. Neste caso, os robôs compartilham a informação da distância percorrida. Tarefas coletivas usualmente lidam com a interação entre os robôs. Esta interação pode ocorrer através de comunicação, quando os robôs são capazes de transmitir informações entre si. Também é possível a interação por meio de detecção, quando cada robô possui sensores capazes de identificar a presença de outros membros do enxame.

Diversas tarefas são estudadas no contexto da robótica de enxame [3, 4]. Na Seção 2.4 desta tese são apresentadas as tarefas mais abordadas na literatura especializada. Uma característica observada nestes estudos é que muitos são direcionados na implementação do *software* de controle para alguma tarefa específica. Seguindo esta metodologia, o desenvolvimento de tarefas ocorre de forma independente umas das outras. Havendo a necessidade de implementação de diferentes comportamentos, um controlador deverá ser projetado e desenvolvido para cada tarefa, considerando as características específicas da tarefa e do *hardware* do robô empregado. Embora estudos baseados nesta metodologia obtenham resultados, tal abordagem também reflete uma dificuldade na robótica de enxame, tendo em vista o esforço e tempo necessários para o projeto e implementação de cada tarefa.

1.2 Metodologias e Modelagem de Enxames

Tendo em vista a dificuldade existente no desenvolvimento individual de diferentes comportamentos, muitos estudos são direcionados para a implementação automática de controladores distribuídos para enxames de robôs. Também há um direcionamento para o estudo do enxame em níveis de abstração mais elevados.

A *robótica evolutiva* é um exemplo de metodologia de projeto que emprega conceitos de computação evolucionária no desenvolvimento de sistemas robóticos [5]. Neste tipo de abordagem, a execução de algoritmos evolucionários visa o aperfeiçoamento de diferentes aspectos de robôs, tal como a calibração do *software* de controle em relação a sinais de entrada (leitura de sensores) e saída (movimentação do robô e manipulação de objetos). Por exemplo, em Duarte *et al.* [6], a computação evolucionária foi usada na criação de controladores para enxames de robôs aquáticos. Segundo os autores, a maior dificuldade de seu trabalho foi a modelagem do ambiente usado para a evolução dos controladores.

O projeto de comportamentos complexos é necessário para o desenvolvimento de aplicações eficientes em enxames. As denominadas *estratégias de planejamento* visam definir como uma determinada tarefa pode ser dividida em um conjunto de operações. No trabalho de Ziparo *et al.* [7], os autores usam redes de Petri para formular planos de execução de tarefas, permitindo o projeto de comportamentos de múltiplos robôs. O objetivo de tal estratégia é auxiliar a concepção e implementação

de comportamentos de robôs a partir de descrições de alto nível. Esta metodologia de modelagem oferece características tais como concorrência, execução distribuída e a análise formal.

A modelagem de sistemas robóticos é necessária para a avaliação de diferentes aspectos. Modelos podem ser empregados, por exemplo, para verificar se uma tarefa é possível, ou ainda, avaliar quantos robôs são necessários para que uma tarefa possa ser realizada. Os modelos de enxames podem ser expressados em diferentes níveis de abstração. A modelagem de um enxame pode ser *microscópica*, quando é levada em consideração a interação entre os robôs, ou *macroscópica*, quando o enxame como um todo é considerado em um alto nível de abstração.

No trabalho de Mermoud *et al.* [8] foram comparados os pontos fortes e fracos de metodologias baseadas em modelos macroscópicos e microscópicos. Em seu trabalho, tais abordagens para projeto de controladores foram chamadas de *bottom-up* e *top-down*. Os autores mostram que estratégias *top-down*, isto é, quando a interação entre robôs é produzida a partir do comportamento macroscópico do enxame, não são adequadas para todos os tipos de tarefas. As estratégias *buttom-up* mostram-se robustas, mas lidam com a dificuldade existente em se obter um comportamento global a partir de interações locais.

As técnicas supracitadas possuem como interesse comum a capacidade de geração de comportamentos coletivos em enxames de robôs de forma sistemática. Sem dúvida, tarefas de naturezas diferentes possuem particularidades que devem ser levadas em conta no projeto do *software* de controle. O desenvolvimento de procedimentos para a implementação de tarefas que sejam comuns a diferentes aplicações é de interesse para a robótica de enxame. Tais metodologias auxiliam na redução do esforço empregado na busca por implementações dedicadas.

1.3 Estratégia de Ondas de Mensagens

Esta tese propõe uma metodologia geral para desenvolvimento de tarefas em enxames de robôs. Esta abordagem tem como premissa a capacidade do enxame operar como um sistema de computação distribuída. Os robôs são elementos de processamento que realizam alguma computação localmente. Ao mesmo tempo, estes robôs são capazes de interagir com outros robôs próximos por meio do envio e recebimento de mensagens.

A partir destas duas características de computação e comunicação, idealizou-se o chamado *algoritmo distribuído de ondas de mensagens*, aplicado à robótica de enxame. Este algoritmo foi concebido para ser executado localmente pelos robôs. A inspiração para este mecanismo vem dos algoritmos de propagação de informação em redes conectadas [9]. Os robôs reagem ao recebimento de uma mensagem enviando

outras mensagens para robôs vizinhos. Esta reação em cadeia resulta em uma onda de mensagens propagada a partir de um inicializador, isto é, um robô que inicia a propagação de forma deliberada. O funcionamento do algoritmo de ondas de mensagens é detalhado no Capítulo 3.

A estratégia de ondas de mensagens é de uso geral, isto é, não foi concebida especificamente para uma tarefa em particular. Esta tese mostra como o conceito de propagação de mensagens pode ser empregado como fundamento para tarefas coletivas de naturezas distintas. Isto contrasta com a ideia de implementações dedicadas introduzida na Seção 1.1. De fato, esta estratégia lida com duas ações básicas: o envio de mensagens de propagação a partir de um robô inicializador e o subsequente envio de mensagens de realimentação no sentido oposto. Estas duas ações auxiliam no projeto de controladores de robôs para realizar tarefas onde há a necessidade de divulgação de algum tipo de informação. Também é possível realizar a sincronização de ações por meio da onda de mensagens, com uma segunda tarefa só sendo iniciada no enxame após o término da primeira.

A generalidade da estratégia proposta foi validada por meio da implementação de diferentes tipos de tarefas em um enxame de robôs simulado no ambiente *V-REP* [10]. Embora não seja uma implementação no mundo real, este ambiente é capaz de simular características físicas como gravidade, atrito, inércia e colisões. Ao mesmo tempo, o *V-REP* possibilita a alteração e personalização de modelos de robôs. Esta característica mostrou-se fundamental para este trabalho, pois diferentes recursos são necessários para diferentes tarefas. Foi escolhido para uso nos experimentos desta pesquisa um modelo simulado do robô *Kilobot* [11]. Este robô possui características de comunicação mínimas à realização da propagação de mensagens. Vídeos ilustrativos[1] com o registro de simulações estão disponíveis para visualização em [12].

Nesta tese, são discutidas e analisadas a idealização, implementação e avaliação dos comportamentos coletivos de recrutamento, alinhamento, navegação, agregação e dispersão. A concepção de cada uma destas tarefas é descrita no Capítulo 4. Todos estes casos foram baseados na estratégia de ondas de mensagens. Neste contexto, estes comportamentos podem ser caracterizados como tarefas básicas, subtarefas ou tarefas complexas. Uma tarefa complexa é um comportamento composto por outras tarefas, chamadas subtarefas. Estas subtarefas são executadas em sequência para formar uma tarefa complexa. Mesmo uma subtarefa pode ser composta por outras subtarefas. Uma tarefa básica é uma operação que não pode ser subdividida. Neste trabalho, o recrutamento de robôs é uma tarefa básica que é empregada como subtarefa na navegação, que por sua vez, é uma tarefa complexa.

[1] `https://www.youtube.com/channel/UCjMHBBznDeXfL6kB-UcQNFg`

1.4 Contribuições

As principais contribuições desta tese são listadas a seguir:

- A realização de um extenso levantamento do estado da arte em robótica de enxame. Foram analisadas as diferenças presentes nas principais plataformas de desenvolvimento existentes na atualidade, incluindo modelos de robôs físicos e ambientes de simulação. Também foram identificadas as tarefas coletivas mais estudadas em pesquisas nesta área. Tal investigação proporcionou a publicação de um artigo de revisão em periódico [13].

- A idealização e elaboração do algoritmo distribuído de ondas de mensagens executado em um enxame de robôs móveis. Este algoritmo forma a base para tarefas onde há a necessidade de envio de informações através de robôs vizinhos. A propagação da onda gera uma relação entre robôs vizinhos. O sistema de comunicação sem fio permite que um robôs recebam mensagens de quaisquer outros robôs na vizinhança. A relação entre robôs filhos e robôs pais pode ser empregada para restringir a comunicação. As mensagens recebidas por um robô que não foram enviadas por seu robô pai ou por robôs filhos são descartadas. Os robôs que recebem e propagam a onda de mensagens constituem um *grupo de robôs*. Foi feita uma representação por *máquina de estados* do algoritmo local, isto é, a operação realizada localmente pelos robôs. A transição de estados é controlada pelo recebimento de mensagens enviadas por robôs vizinhos.

- A implementação de uma tarefa de *recrutamento de robôs* seguindo a estratégia proposta. Quando as mensagens contém alguma informação, esta passa a ser conhecida por todos os robôs do grupo. Nesta situação, a execução do algoritmo realiza a tarefa de recrutamento de robôs. O recrutamento, por ser baseado na propagação de informação com realimentação, possui terminação. Esta característica é importante, pois todas as tarefas com tal estrutura possuirão um robô, o inicializador da propagação, com o conhecimento da terminação global da tarefa. Nesta situação, um próprio membro do enxame sabe que os demais terminaram a tarefa, não havendo a necessidade de um observador externo. O estudo do recrutamento proporcionou a publicação de um artigo de conferência [14] e um artigo de periódico [15].

- A implementação de uma tarefa de *alinhamento de robôs* seguindo a estratégia proposta. O alinhamento é realizado pelo envio do ângulo com o qual um robô observa um vizinho, considerando a existência de um sistema de distância e direção (*range and bearing*). O envio e recebimento dos ângulos ocorre com

base na relação entre robôs pais e filhos. Isto permite que o direcionamento global do enxame convirja para a direção do robôs inicializador. Esta abordagem garante que um direcionamento global do enxame seja atingido sem que haja a necessidade de um referencial externo.

- A implementação de uma subtarefa de *movimentação coordenada* usando um esquema de seguidor de líder *leader following* combinado com a relação entre robôs pais e filhos. Este esquema de movimentação permite que o enxame mantenha a formação original sem que haja a necessidade do conhecimento global desta formação. Cada robô conhece apenas a direção e a distância em relação a seu robô pai.

- A possibilidade de execução sequencial das subtarefas que usam a estratégia de ondas de mensagens. Nesta tese é proposto que tarefas complexas sejam implementadas como uma sequência de subtarefas. Uma tarefa de *navegação coletiva* é apresentada, sendo esta composta por subtarefas de recrutamento, alinhamento e movimentação. O estudo da navegação coletiva proporcionou a publicação de um artigo de conferência [16], um artigo de periódico [17] e um capítulo de livro [18].

- A elaboração de um algoritmo de controle que possibilita a execução tanto da agregação quanto da dispersão dos robôs. Esta tarefa foi denominada *redimensionamento do enxame*. O comportamento de atração ou repulsão entre robôs é definido pelo ajuste de parâmetros do algoritmo. O redimensionamento é realizado com o uso de regras de *flocking*. A movimentação de cada robô é definida pelo posicionamento dos robôs vizinhos. A estratégia de onda de mensagens opera em conjunto com as regras de *flocking* para definir quando os robôs encerrarão sua movimentação.

1.5 Organização

Os demais capítulos desta tese têm a organização descrita a seguir. No Capítulo 2 são apresentadas informações gerais sobre a robótica de enxame. Isto inclui suas inspirações na natureza e a comparação outros sistemas artificiais compostos por múltiplos agentes, as principais plataformas de desenvolvimento e as tarefas coletivas mais estudadas. No Capítulo 3 são introduzidos conceitos básicos de sistemas distribuídos, direcionados para o contexto da robótica de enxame. É apresentada uma descrição das características gerais que o sistema de comunicação no enxame necessita para a possibilitar a realização de algoritmos distribuídos, mais precisamente, a propagação de informação. No Capítulo 4 são apresentadas tarefas imple-

mentadas seguindo a estratégia de onda de mensagens. Os algoritmos criados para as subtarefas que constituem a navegação coletiva são discutidos, compreendendo o recrutamento, o alinhamento e a movimentação dos robôs. Neste capítulo ainda é discutido o redimensionamento do enxame, com o qual é possível realizar as tarefas de agregação e dispersão. O Capítulo 5 apresenta os experimentos de simulação realizados para validar as tarefas propostas. O Capítulo 6 conclui esta tese, ressaltando as principais contribuições deste trabalho, bem como as dificuldades observadas em seu desenvolvimento. Por fim, são apresentados os possíveis direcionamentos de trabalhos futuros.

Capítulo 2

Robótica de Enxame

Os sistemas de múltiplos robôs (*Multi-Robot Systems*, MRS) surgiram como uma extensão do estudo da robótica. Em um MRS, os robôs devem ser capazes de concluir uma determinada tarefa mais rápido que um único robô isolado [19], atuando em diferentes locais ao mesmo tempo. Este tipo de sistema possui um paralelismo intrínseco, podendo ainda se valer da tolerância a falhas devido à redundância de robôs.

O termo *enxame* foi introduzido no contexto da robótica por Beni [1] em seu estudo de *sistemas robóticos celulares*. Segundo Beni, um enxame robótico deve possuir certas características, tais como controle descentralizado, assincronismo e ser composto por indivíduos simples e quase idênticos. Este conceito de enxame mostrou-se robusto o suficiente a ponto de explicar comportamentos em sociedades de insetos no campo da biologia [20, 21]. A ideia geral de enxame também possibilitou o desenvolvimento de várias meta-heurísticas bioinspiradas, como a Otimização por Colônia de Formigas [22] e a Otimização por Enxame de Partículas [23], entre outras [24].

Uma definição de Robótica de Enxame amplamente aceita na literatura foi instanciada por Şahin [25]:

Definição 1 *Robótica de Enxame é o estudo de como agentes físicos simples podem ser projetados de forma que um determinado comportamento coletivo emerja da interação local entre agentes e entre os agentes e o ambiente.*

Três características desejadas na robótica de enxame são a *robustez*, a *flexibilidade* e a *escalabilidade*. Um sistema de enxame é dito robusto se este for capaz de continuar a operar corretamente mesmo que um ou mais membros do enxame deixem de funcionar devido a problemas diversos, como falhas internas ou mudanças no ambiente. A robustez pode ser alcançada pela redundância de indivíduos somada à ausência de líderes. A escalabilidade está relacionada com a capacidade do sistema funcionar com diferentes quantidades de membros. Assim, um indivíduo é

capaz de operar em enxames com poucos ou muitos membros, sem que haja uma variação significativa no seu desempenho. O enxame é considerado flexível quando é possível implementar mudanças em seu comportamento. Em um sistema flexível, os parâmetros internos dos indivíduos podem ser ajustados na ocorrência de mudanças no ambiente ou seguindo regras preestabelecidas pelo projetista. Estes três conceitos estão relacionados com uma mesma ideia, que determina que um sistema de enxame deve ser capaz de realizar uma determinada tarefa sob diferentes condições. Embora sejam encontradas em sistemas de enxame naturais, tais características são difíceis de serem replicadas em enxames artificiais, podendo até mesmo serem impossíveis em determinadas plataformas robóticas devido a limitações de *hardware*. Winfield *et al.* [26] argumentam a real possibilidade de enxames de robôs alcançarem estas três características. Os autores concluem que robustez e escalabilidade podem ser obtidas com o estudo de tolerância a falhas.

De fato, embora os aspectos gerais sobre a robótica de enxame sejam bem definidos, não há um consenso na literatura sobre quais características específicas a diferenciam de outros estudos envolvendo sistemas de múltiplos robôs. Por exemplo, Şahin [25] descreve cinco características da robótica de enxame:

1. Um enxame é composto por *robôs autônomos* capazes de se movimentar em um determinado ambiente e interagir com outros objetos.

2. O controle deve permitir a existência de um grande número de robôs.

3. O enxame é homogêneo ou é composto por grupos homogêneos de robôs.

4. Robôs devem interagir e cooperar para a realização de uma determinada tarefa.

5. As capacidades de comunicação e detecção são locais, não havendo possibilidade de compartilhamento global de informações.

O principal objetivo da robótica de enxame, assim como qualquer sistema robótico, é realizar tarefas que auxiliem o trabalho humano. Os robôs podem se mover em regiões perigosas, usar diferentes tipos de sensores para observar o ambiente e/ou operar por longos períodos de tempo. Robôs ainda podem ser considerados mais eficientes que humanos em determinadas tarefas devido à sua capacidade computacional e à precisão de seus sistemas. Mesmo que a robótica de enxame ainda não seja algo comum em nosso cotidiano, diversas pesquisas acadêmicas, como as descritas na Seção 2.4, indicam o interesse nessa direção [3, 4]. Dois trabalhos marcantes nesta área são o *enxame de mil robôs* apresentado por Rubenstein et al. [27] e o grupo de robôs construtores inspirados em insetos desenvolvido por Werfel et al. [28]. Em ambos os casos, os robôs realizam uma tarefa de montagem. No primeiro trabalho, os robôs se organizam espacialmente para que o enxame apresente,

no nível global, formas predefinidas pelo usuário. No segundo trabalho, os robôs se organizam para construírem estruturas complexas a partir de blocos simples. Estes dois estudos mostram um enxame de robôs realizando uma tarefa útil em resposta a uma determinada configuração definida pelo usuário. Contudo, estes trabalhos mostram apenas alguns aspectos dos enxames relacionados com o controle dos robôs, não abordando outros desafios da área, como a robustez em face a tarefas dinâmicas.

O projeto completo de um enxame de robôs pode ser dividido em vários níveis, desde o estudo de materiais para a implementação física de robôs, até a descrição abstrata e matemática de comportamentos coletivos. Projetar controladores distribuídos é importante, pois é o que define como cada robô deve reagir ao ambiente para executar uma determinada tarefa. Contudo, a implementação desses controladores, na maioria dos casos, é um processo complexo. Cada robô deve processar as leituras de sensores e receber mensagens de robôs vizinhos para, em seguida, decidir como mover-se e interagir com outros robôs. Abordagens baseadas em computação inteligente são usadas para facilitar este projeto [29], alcançando resultados aceitáveis em situações em que a modelagem analítica é difícil, ou mesmo impossível. Um exemplo é o uso de *Computação Evolucionária* para adaptar iterativamente as configurações do controlador do robô [5]. Algumas abordagens também fazem uso da própria arquitetura de enxame para a execução de algoritmos bioinspirados, como por exemplo, a agregação de robôs baseada no comportamento de abelhas [30] e o particionamento de tarefas com base na otimização do enxame de partículas [31].

Esta tese propõe uma metodologia para desenvolvimento de tarefas robóticas. Como será exposto nos capítulos seguintes, a estratégia de ondas de mensagens é usada como base para a criação de tarefas em enxames operando como sistemas distribuídos de computação, isto é, com cada robô realizando algum processamento e um sistema de comunicação por troca mensagens intermediando a interação entre os robôs. Para a realização deste trabalho, é fundamental o estudo abrangente de diversos aspectos da robótica de enxame. Neste capítulo são investigadas as características das plataformas existentes para pesquisa de enxames, analisando quais são as mais adequadas para o desenvolvimento deste trabalho. Também é feito um levantamento das tarefas coletivas mais estudadas em pesquisas de de robótica de enxame. Algumas destas tarefas serão empregadas como prova de conceito para a estratégia proposta.

A Seção 2.1 apresenta o conceito geral de *inteligência de enxame*, sua ocorrência na natureza e seu uso em técnicas computacionais. Na Seção 2.2 é apresentada uma descrição das principais plataformas de desenvolvimento para robótica de enxame. A forma com que os estudos de robótica de enxame são comumente classificados na literatura é apresentada na Seção 2.3. Tarefas coletivas básicas realizadas por enxames de robôs são descritas na Seção 2.4

2.1 Inteligência de Enxame

A chamada *Inteligência de Enxame* é um termo geral usado para classificar sistemas compostos por múltiplos agentes capazes de solucionar coletivamente algum tipo de problema [2]. Este conceito descreve o comportamento de insetos sociais, que por sua vez, são inspiração para sistemas artificiais.

2.1.1 Enxames Naturais

Na natureza, diversos seres apresentam comportamentos sociais. Tais indivíduos não apenas constituem uma população, mas também a evolução de mecanismos de interação permitiu a emergência de grupos auto-organizados. Para certas espécies, a interação entre os membros resultou em uma maior capacidade de exploração de recursos, e consequentemente, em uma maior eficiência na manutenção das populações. O aperfeiçoamento destes mecanismos por meio da evolução natural possibilitou o surgimento de comportamentos coletivos sofisticados, tais como:

- *Colônias de Bactérias*: Sendo individualmente organismos unicelulares, bactérias são capazes de se organizar em colônias multicelulares [32]. A divisão de trabalho e especialização de determinadas células permitem uma maior proteção da colônia contra agentes externos, otimizando a sobrevivência da população.

- *Sociedades de Insetos*: Formigas, cupins, abelhas e vespas habitam colônias compostas por milhares de indivíduos. Em uma mesma espécie, grupos de insetos se especializam em tarefas específicas, tais como reprodução, construção ou defesa, entre outras. A auto-organização nestas colônias existe devido à interação entre os insetos, com o uso de feromônios, modificações do ambiente, ou mesmo por contato físico [33].

- *Cardumes*: Peixes são capazes de nadar em grupos coordenados, com movimentos rápidos e evitando colisões. Tal organização possibilita que o cardume encontre alimento com maior facilidade e ao mesmo tempo aumente a proteção contra predadores [34].

2.1.2 Algoritmos de Enxame

Embora a inteligência do enxame tenha sido originalmente descrita por Beni [35] como uma característica de sistemas robóticos, este conceito também possibilitou o surgimento de vários métodos computacionais bem-sucedidos. Algoritmos bioinspirados utilizam múltiplos agentes para propor soluções numéricas de problemas matemáticos complexos. A interação entre os agentes segue modelos de comunicação

inspirados em grupos de animais e sociedades de insetos [2]. Algumas das meta-heurísticas de otimização mais difundidas são brevemente descritas a seguir.

A *Computação Evolucionária* [36] é um campo de estudo caracterizado por algoritmos que são inspirados em mecanismos de evolução biológica, tais como reprodução, mutação, recombinação e seleção. Em termos gerais, a computação evolutiva lida com a evolução de gerações sucessivas de uma população de indivíduos. A qualidade de cada indivíduo está relacionada com a probabilidade deste se reproduzir, propagando assim seus genes para a próxima geração de indivíduos. Este processo, muitas vezes usado para descrever a evolução biológica, é o fundamento dos algoritmos evolutivos. Um exemplo bem-sucedido são os Algoritmos Genéticos (*Genetic Algorithms*, GAs) , propostos por Holland [37]. Esta meta-heurística pode ser usada para gerar soluções úteis em problemas de otimização e busca. Em um GA, a população é composta por soluções candidatas do problema a ser resolvido. Cada solução tem um conjunto de propriedades organizadas em uma codificação binária, chamada cromossomos ou genótipo. A aptidão ou qualidade das soluções é definida por uma função objetiva, que é, em geral, dependente do problema. A execução do GA é iterativa: as soluções com maior aptidão possuem maior probabilidade de passar seus genes para as seguintes gerações. Operações genéticas, tais como cruzamento e mutação, permitem uma maior diversidade de novas soluções na geração seguinte.

A Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO) [23] é uma meta-heurística inicialmente usada na resolução de problemas contínuos. Esta técnica é inspirada pelo voo coordenado de pássaros. O objetivo é encontrar o valor ótimo de uma função em um problema de otimização mono objetivo. Para isso, o algoritmo usa múltiplos agentes, chamados de partículas. Cada partícula desloca-se dentro do espaço de busca da função objetivo. A posição de cada partícula está associada a uma solução do problema em questão. As partículas atualizam iterativamente sua velocidade e direção usando soluções encontradas localmente (componente cognitivo da velocidade da partícula) e globalmente (componente social da velocidade da partícula). A utilização da informação local e global faz com que as posições das partículas tendam a regiões do espaço de busca associadas com valores mais próximos de ótimos locais ou global da função objectivo. Em cada iteração, cada partícula deve compartilhar informações sobre a solução obtida com outras partículas do enxame. Assim, o PSO requer uma topologia de comunicação eficiente entre as partículas.

Outra meta-heurística amplamente utilizada é a Otimização por Colônia de Formigas (*Ant Colony Optimization*, ACO) [22]. Este método é baseado na busca de caminhos feitos por formigas. Cada agente, conhecido como formiga artificial, percorre o espaço de pesquisa associado ao problema cuja solução deve ser encontrada.

Ao contrário do PSO, que foi inicialmente utilizado em espaços de busca é contínuos, o ACO foi inicialmente desenvolvido para resolver problemas de otimização discreta, onde o espaço de pesquisa é geralmente descrito por um grafo. O método também é iterativo. Assim, em cada ciclo, as formigas artificiais encontram caminhos, que são soluções potenciais do problema, e usam o conhecimento para dirigir suas iterações futuras, bem como as soluções de outras formigas. Esta meta-heurística difere de outros métodos pela maneira indireta na qual os agentes se comunicam. Formigas artificiais usam um feromônio artificial que é depositado ao longo de caminhos visitados no espaço de busca. Quanto maior for a concentração de feromônio num determinado percurso, melhor será a qualidade da solução correspondente. Há também a noção de evaporação de feromônio, que é aplicada em cada iteração. Assim, ao longo das iterações, o feromônio associado a más soluções torna-se cada vez menor, enquanto soluções próximas ao melhor caminho terão seu feromônio sempre reforçado por formigas que optam por usá-lo devido à alta concentração correspondente de feromônio.

Motivados pelos resultados promissores do PSO e ACO, muitos outros algoritmos inspirados na natureza foram idealizados [38]. Clerc propôs um PSO livre de parâmetros, chamado *Tribes* [39]. Karaboga desenvolveu o algoritmo de Colônia de Abelhas Artificial (*Artificial Bee Colony*, ABC) [40] para otimizar funções numéricas multivariáveis e multimodais. O Algoritmo de Vaga-lumes (*Firefly Algorithm*) [41] é inspirado no comportamento social dos vaga-lumes e no fenômeno da comunicação bioluminescente. O *Cuckoo Search* [42] é outra técnica, baseada no comportamento reprodutivo de certas espécies de cucos, como o parasitismo da ninhada. O Algoritmo de Otimização por Pesquisa de Retrocesso (*Backtracking Search Optimization Algorithm*, BSA) [43] é um algoritmo evolucionário híbrido que explora alguns princípios de enxame para resolver problemas de otimização com valores reais. Wolpert e Macready [44] apontam que a diversidade existente de métodos de otimização está relacionada com as características particulares de cada problema a ser solucionado.

As técnicas computacionais de inteligência de enxame discutidas nesta seção têm características comuns, tais como o (*i*) uso de múltiplos agentes (processamento distribuído), que avaliam a qualidade de certas magnitudes, medidas ou calculadas; e (*ii*) a comparação de tais magnitudes obtidas por diferentes agentes, utilizando comunicação local ou global.

2.2 Plataformas de Desenvolvimento

O projeto de sistemas robóticos pode ser dividido em dois componentes: *hardware* e *software*. O *projeto de hardware* é a concepção e implementação física da estrutura

do robô, bem como a especificação de placas de circuito impresso e componentes, como processadores, sensores e atuadores. O *projeto de software* é representado pelos algoritmos executados individualmente pelos robôs. Estes componentes do sistema devem ser projetados de tal forma que o comportamento desejado do enxame seja decorrente do comportamento individual dos robôs, da interação entre si e com o ambiente.

A avaliação dos métodos ou comportamentos descritos por enxames de robôs pode ser realizada de várias maneiras. Muitos dos trabalhos citados neste capítulo são focados no desenvolvimento do *software* distribuído que controla o funcionamento de robôs reais ou modelos simulados. Por outro lado, considerar também as capacidades de fabricação é uma abordagem diferente, mas igualmente viável.

2.2.1 Robôs Físicos

O uso de robôs reais é uma opção que se aproxima de aplicações do mundo real, embora tais experimentos sejam usualmente realizados em ambientes controlados. Vários modelos de robôs são empregados em trabalhos de pesquisa. Existem modelos projetados especificamente para alguma determinada pesquisa, e modelos comerciais, projetados para uso geral. A seguir são descritos alguns dos robôs de enxame mais conhecidos.

O *s-bot* é um robô móvel desenvolvido para o *Swarm-bots Project* de Dorigo *et al.* [45]. O grupo de *s-bots*, mostrado na Figura 2.1(a), é chamado *swarm-bot*. A principal característica do *s-bot* é a capacidade de se conectar fisicamente a outros membros do enxame. Esse recurso permite que o enxame execute tarefas de auto-organização e automontagem. Os robôs também possuem capacidades de detecção, processamento e comunicação, necessárias para executar diferentes tipos de tarefas coletivas, tais como navegação e transporte. A extensão deste trabalho é o *Swarmanoid Project* [46]. Três robôs diferentes, mostrados na Figura 2.1(b), são usados para formar um enxame heterogêneo. Os robôs móveis similares ao *s-bot* são chamados *foot-bots*, e podem andar no solo e explorar o ambiente. O *hand-bot* é um robô que move-se escalando estruturas verticais. Ele também pode manipular objetos usando garras. O movimento horizontal do *hand-bot* é obtido em associação com os *foot-bots*. O terceiro robô, *eye-bot*, é um robô voador que explora o ambiente com o uso de câmeras, fornecendo esta informação aos robôs no solo.

O robô móvel *SwarmBot*, mostrado na Figura 2.1(c), é o membro individual do *iRobot Swarm Project* desenvolvido por McLurkin [47]. Tal projeto visa desenvolver algoritmos distribuídos para controlar eficientemente um enxame de centenas de robôs. Este enxame foi bem-sucedido em executar a dispersão do robôs em interiores, usando a comunicação gradiente vizinha. Outro sistema de múltiplos robô projetado

por McLurkin é o *r-One* [48]. É um projeto de robô de baixo custo para pesquisas de enxame em grande escala e propósitos educacionais para alunos mais jovens. O *r-One*, ilustrado na Figura 2.1(d), possui vários dispositivos, incluindo um giroscópio, um acelerômetro, dois codificadores de roda (*encoders*), um rádio de controle global (*radio frequency*, RF), um farol infravermelho (*infrared*, IR) para a localização e utiliza também sensores IR para comunicação entre robôs. Sua operação pode ser expandida para tarefas de transporte usando um dispositivo manipulador [49].

O *Khepera*, ilustrado na Figura 2.1(e), é um robô móvel em miniatura desenvolvido por Mondada *et al.* [50] para uso em pesquisa e educação. Três versões atualizadas foram lançadas pelo K-Team [51–53]. Sua versão mais recente, o *Khepera IV*, mostrada na Figura 2.1(f), inclui doze sensores IR, cinco transceptores ultra-sônicos, uma unidade de medida inercial com um acelerômetro e um giroscópio, uma câmera frontal e dois codificadores de roda. Também é compatível com os módulos de hardware externos feitos para as versões anteriores do Khepera, como um manipulador de pinça e um localizador laser. Sendo um produto comercial, o *Khepera* é um modelo de robô em constante evolução, com melhorias de desempenho em cada nova versão e inclusão de novos recursos, e uma maior compatibilidade com os recentes sistemas operacionais recentes, protocolos de comunicação e linguagens de programação.

O robô *Kilobot* [11] é um robô móvel desenvolvido para experimentação de comportamentos de enxame e que possui tamanho reduzido em relação a outros modelos. Sua movimentação ocorre por meio de vibrações produzidas por dois motores de passo. O *Kilobot* possui um sistema de comunicação IR capaz de realizar o envio e recebimento de mensagens. Este mesmo sistema faz a medição da distância em relação a outros robôs com base na intensidade do sinal da mensagem recebida. Esta característica de comunicação fez do *Kilobot* uma plataforma adequada para o desenvolvimento da estratégia proposta neste trabalho, conforme será exposto nos capítulos seguintes.

Outras plataformas de múltiplos robôs comerciais ou de código aberto estão disponíveis, como o *E-puck* [54], o *WolfBot* [55] e o *Pheeno* [56]. Devido ao tamanho miniaturizado desses robôs, seu custo de produção é geralmente menor do que o de robôs complexos. Esta característica pode reduzir o custo de manter um conjunto de vários robôs durante o estágio de desenvolvimento do projeto.

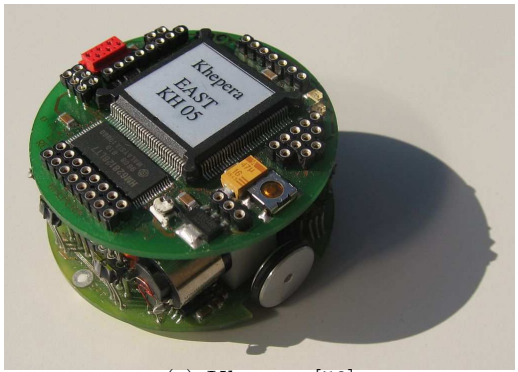
O uso de robôs simples é impulsionado pela sua adequação para executar uma série de tarefas. A alteração de alguns aspectos do comportamento desejado, como a inclusão de novas funcionalidades, pode levar a requisitos de *hardware* que não existem no início do projeto. Em alguns casos tais limitações podem ser superadas com o auxílio de dispositivos externos. Por exemplo, o *e-puck* é um modelo de robô que suporta a conexão de placas externas para comunicação por IR. Para evitar possíveis limitações existentes em robôs disponíveis, alguns projetos incluem

(a) S-bot [45]

(b) Swarmanoid robots [46]

(c) iRobot SwarmBot [47]

(d) r-One [48]



(e) Khepera [50]

(f) Khepera IV [51]

Figura 2.1: Robôs móveis usados em pesquisas de robótica de enxame.

o desenvolvimento de seus próprios modelos de robôs, como é o caso do *Swarm-bots Project*. Por outro lado, se os pesquisadores já possuem um modelo específico de robô e desejam usá-lo, então as limitações do modelo devem ser consideradas. Para os pesquisadores que desejam implementar um comportamento específico em um modelo comercial, pode ser necessário verificar extensivamente quais recursos de hardware esse comportamento pode exigir. Para ajudar nesta etapa intermediária do projeto, os simuladores são normalmente empregados para experimentar diferentes combinações de *hardware*. Isto é discutido na próxima seção.

2.2.2 Simuladores

O uso de simuladores é outra forma de realizar experimentos em robótica de enxame. Este é um passo intermediário entre a verificação computacional em um alto nível de abstração, tais como o uso de modelos matemáticos e a validação usando robôs físicos. Os simuladores podem se diferenciar de várias maneiras, desde softwares multiagentes para aplicações acadêmicas específicas, até simuladores comerciais com ambientes virtuais sofisticados. Nesta seção serão enumerados alguns dos simuladores de múltiplos robôs mais empregados em estudos de robótica de enxame.

O *V-REP* (*Virtual Robot Experimentation Platform*) [10] é um simulador comercial tridimensional com licença gratuita para uso educacional. O *V-REP* é baseado em arquitetura de controle distribuído. Cada objeto ou modelo pode ser controlado individualmente através de um *script* dedicado. O simulador também suporta controladores escritos em uma variedade de linguagens de programação. Isso torna o *V-REP* versátil e muito útil para aplicações de múltiplos robô. O *V-REP* possui uma biblioteca contendo inúmeros modelos de robôs. Este simulador será empregado na validação da estratégia proposta neste trabalho por possuir modelos de robôs de enxame. Mais precisamente, o *V-REP* será empregado na implementação de tarefas em enxames simulados de *Kilobots*.

O *Player Project* [57] é um simulador de código aberto para robôs, sensores e atuadores. Ele inclui uma ampla biblioteca de robôs e outros dispositivos relacionados. Experimentos com vários robôs podem ser realizados no *Stage*, que é um ambiente bidimensional que está vinculado ao *Player* e pode operar com até 1000 robôs. Uma extensão deste projeto é o simulador tridimensional *Gazebo* [58].

O *Enki* [59] é um simulador que opera em um ambiente bidimensional (2D) e que faz uso de modelos inspirados em robôs móveis comerciais, como o *Khepera* e o *s-bot*. Tal simulador é capar de realizar a detecção de colisões entre robôs, mas possui suporte limitado para a simulação física. A principal característica de *Enki* é a velocidade de simulação, que pode simular certos comportamentos do enxame muito mais rápido do que com robôs reais.

O *Webots* [60] é um simulador comercial com ambiente em três dimensões. Permite experimentos com um único robô ou com múltiplos robôs interagindo. Possui suporte para a física, como gravidade e inércia, e é capaz de detectar colisões entre os objetos simulados. Fornece uma extensa biblioteca com modelos de robôs comerciais, e permite ainda a simulação de robôs personalizados com o acréscimo de sensores e atuadores também presentes na biblioteca.

O *ARGoS* [61] é um simulador de robótica com suporte a física e de código aberto. Este simulador pode executar experimentos complexos envolvendo enxames de robôs em grande escala. A arquitetura do *ARGoS* é otimizada para o uso em processadores com múltiplos núcleos. Além disso, é altamente modular, permitindo fácil adição de recursos personalizados e apropriada alocação de recursos computacionais. Os resultados experimentais apresentados pelos desenvolvedores de robôs [61] demonstram que o tempo de execução da simulação aumenta linearmente com o número de robôs.

O *USARSim* (*Urban Search and Rescue Simulation*) [62] é outro simulador robótico de código aberto de alta-fidelidade que pode ser usado tanto para pesquisa quanto para educação. Este simulador é o ambiente usado na competição de robôs virtuais Robocup [63].

Outro simulador utilizado para várias experiências em sistemas de múltiplos robôs é *Roborobo* [64]. Este simulador se baseia numa configuração de *hardware* básica semelhante à de robôs como o *E-puck* e o *Khepera*. O *Roborobo* pode ser considerado um ambiente intermediário entre as ferramentas de simulação mais realistas, porém lentas, e as simulações abstratas baseadas em interações entre múltiplos agentes. Este simulador tem sido amplamente utilizado em vários contextos, mas principalmente voltado para robótica evolutiva e a robótica de enxame.

2.2.3 Discussão sobre as Plataformas

As várias plataformas usadas na pesquisa de robótica de enxame possuem tanto semelhanças quanto diferenças . Algumas características impedem a implementação de uma variedade de tarefas. Na Tabela 2.1 são mostrados alguns recursos de robôs amplamente utilizados em estudos de robótica de enxame.

O *s-bot*, por exemplo, é o robô com o maior conjunto de sensores, variando de detectores IR de distância até incomuns sensores de umidade. Este robô é específico do Projeto *Swarm-bots* e foi intencionalmente projetado com um hardware extenso, necessário para a execução de diversas tarefas de enxame. No outro extremo, o robô *Kilobot* tem a especificação de *hardware* mais simples: um único sensor IR é usado para detectar a distância dos demais robôs. Esta limitação restringe as tarefas que um enxame de *Kilobots* pode realizar, mas também é interessante por permitir

Tabela 2.1: Capacidade de detecção e movimento de diferentes modelos de robôs.

Robô	Sensores	Movimento e manipulação	Comunicação
s-bot	IR, luz ambiente, acelerômetro, *encoders* de rodas, câmera omnidirecional, microfone, umidade, temperatura	rodas, esteira, garras	IR, *Wi-Fi*
r-One	IR, luz ambiente, acelerômetro, *encoders* de rodas, giroscópio, colisão	rodas, garras (opcional)	IR, RF
Khepera IV	IR, luz ambiente, acelerômetro, *encoders* de rodas, giroscópio, câmera, microfone, ultrassom	rodas, garras (opcional)	IR, *Wi-Fi*, *bluetooth*
e-puck	IR, acelerômetro, câmera, microfones	rodas	IR, *bluetooth*, serial
Wolfbot	IR, luz ambiente, acelerômetro, magnetômetro, câmera, microfone	rodas	*Wi-Fi*, *zigbee*
Pheeno	IR, acelerômetro, *encoders* de rodas, magnetômetro, câmera	rodas, garras (opcional)	*Wi-Fi*, *blueto-oth*, serial
Kilobot	IR, luz ambiente	motores de vibração	IR

a exploração do trabalho coletivo usando dispositivos simples. Os sensores de IR estão presentes em todos os robôs listados na Tabela 2.1. Estes são necessários na detecção de obstáculos e durante a comunicação entre robôs. As capacidades de atuação variam de um robô para outro. A maioria dos robôs usam rodas para se mover, exceto para o *Kilobot*, que usa vibração. O *s-bot* também conta com um sistema de esteiras, além das rodas normais, para aumentar o torque. O *Wolfbot* possui um sistema de movimento omnidirecional único, composto de três rodas. Isto permite uma maior mobilidade em comparação com outras plataformas, que são normalmente baseadas em sistemas não-holonômicos (similar a carros) [65].

Apesar dos avanços nas capacidades de processamento e detecção, ainda há limitações na comunicação. Os sensores IR presentes na maioria dos modelos podem ser usados para comunicação entre robôs, mas estes são muito limitados em termos de quantidade de informação que podem ser enviada e recebida. Esta restrição é análoga à comunicação em enxames reais, em que os membros do enxame usam apenas comunicação local. Os robôs listados na Tabela 2.1 também apresentam diferentes versões de comunicação de média ou longo alcance. Tal capacidade é direcionada principalmente à comunicação com algum sistema central de monitoramento, usado por exemplo, para receber a captura de vídeo das câmeras dos robôs. Uma exceção a isso é o *Wolfbot*, que está equipado com um módulo *zigbee*, que pode

estabelecer uma rede de malha com outros *Wolfbots*, permitindo a transmissão de mensagens através do enxame. Para futuras plataformas, a inclusão de infraestruturas de comunicação robustas, tais como *Wi-Fi*, parece ser essencial. Isto permite a implementação de uma variedade de tarefas dependentes do compartilhamento de informações.

Como no caso de robôs reais, as plataformas de simulação também são diversas em termos de recursos, como mostrado na Tabela 2.2. A principal característica que difere entre os simuladores é seu nível de abstração. Alguns simulam ambientes tridimensionais onde fenômenos físicos são considerados, tais como atrito e gravidade. Os outros executam simulações bidimensionais rápidas, considerando apenas a interações entre os robôs.

Tabela 2.2: Características das plataformas de simulação.

Simulador	Ambiente	Física	Robôs	Licença
Gazebo	3D	Sim	Biblioteca e robôs personalizados	Código aberto
Enki	2D	Não	Biblioteca e robôs personalizados	Código aberto
Webots	3D	Sim	Biblioteca e robôs personalizados	Comercial
V-REP	3D	Sim	Biblioteca e robôs personalizados	Educacional
ARGoS	3D	Sim	Foot-bot e robôs personalizados	Código aberto
USARSIM	3D	Sim	Biblioteca e robôs personalizados	Código aberto
Roborobo	2D	Não	Baseados no e-puck e Khepera	Código aberto

2.2.4 Robôs de Uso Dedicado

Os estudos em robótica de enxame podem encontrar sérias limitações devido aos recursos de *hardware* presentes na plataforma adotada. A principal dificuldade diz respeito à quais tipos de tarefas podem ser executadas com um determinado conjunto de sensores e atuadores. Uma opção à aquisição e ao uso de robôs complexos e com versatilidade de recursos é o desenvolvimento de robôs dedicados. Nesta abordagem, é realizado o desenvolvimento não apenas dos controladores em *software*, mas também de todo o projeto físico de um robô, o que pode tornar o enxame altamente especializado em uma tarefa específica. Segundo Bezzo *et al.* [66], duas filosofias de projeto coexistem na atualidade. A primeira diz respeito à criação de modelos de uso geral, com o foco na expansão da capacidade individual de cada robô. Na segunda, há o desenvolvimento de ferramentas que auxiliam a criação de modelos de robôs dedicados ao usuário para a realização de tarefas específicas.

A popularização da impressão tridimensional (3D) permite a criação de partes rígidas de forma arbitrária [67]. No projeto de circuitos eletrônicos, geralmente são consideradas duas opções: a adoção de uma plataforma modular e reutilizável, ou a geração de placas de circuito impresso dedicadas (*printed circuit boards*, PCB). Por

exemplo, Yu *et al.* [68] propuseram um enxame de robôs móveis simples chamado *microMVP* baseado em uma plataforma de prototipagem para microcontroladores e com um corpo implementado por impressão 3D. Pan *et al.* [69] propuseram um robô subaquático com corpo esférico, também implementado usando a tecnologia de impressão 3D. Em seu trabalho, os autores usaram um sistema embutido (*System-on-Chip*, SoC) para incorporar a maioria dos circuitos em um único FPGA (*Field Programmable Gate Array*). Esta é uma abordagem interessante, pois o robô tem uma forma muito específica, mas o controlador eletrônico é baseado em um dispositivo reconfigurável. Em ambos os casos, o projeto do *software* é posterior à implementação física. Primeiro, o robô dedicado é construído a partir de peças impressas em 3D, PCBs, atuadores, sensores, entre outros componentes. Só então o *software* é especificado.

A integração e cogeração dos sistemas mecânico, eletrônico e de *software* é um possível futuro para a fabricação de robôs dedicados. Alguns estudos foram desenvolvidos nessa direção. O ROSLab [70] é um exemplo de ambiente de programação modular para aplicações robóticas baseado em uma biblioteca de recursos, como atuadores e sensores. Em estudos posteriores, o ambiente foi expandido para especificar também o projeto mecânico [71] e os circuitos para o robô desejado [72]. Tais abordagens são específicas para robôs individuais. Contudo, é esperado que futuros sistemas considerem características de enxame, como comunicação e coordenação, para a geração de modelos dedicados a tarefas de enxame.

2.3 Taxonomias

O estudo da robótica de enxame é amplo, sendo uma extensão das pesquisas de sistemas de múltiplos robôs. Diversos autores classificam os trabalhos nessa área considerando algumas características comuns, mas não há um consenso sobre essa classificação. De fato, por ser uma área de estudos em constante evolução, diferentes taxonomias são apresentadas de tempos em tempos. A seguir são discutidas cinco classificações sobre o estudo da robótica de enxame.

Um dos primeiros trabalhos em que uma taxonomia para sistema de múltiplos robôs foi proposta foi feito por Dudek *et al.* [73]. Sua classificação se concentrou mais nos robôs e nas características da arquitetura distribuída de enxame do que nas tarefas que os robôs realizam, ou nas etapas relacionadas ao projeto do enxame. Dudek classificou os enxames considerando os seguintes aspectos:

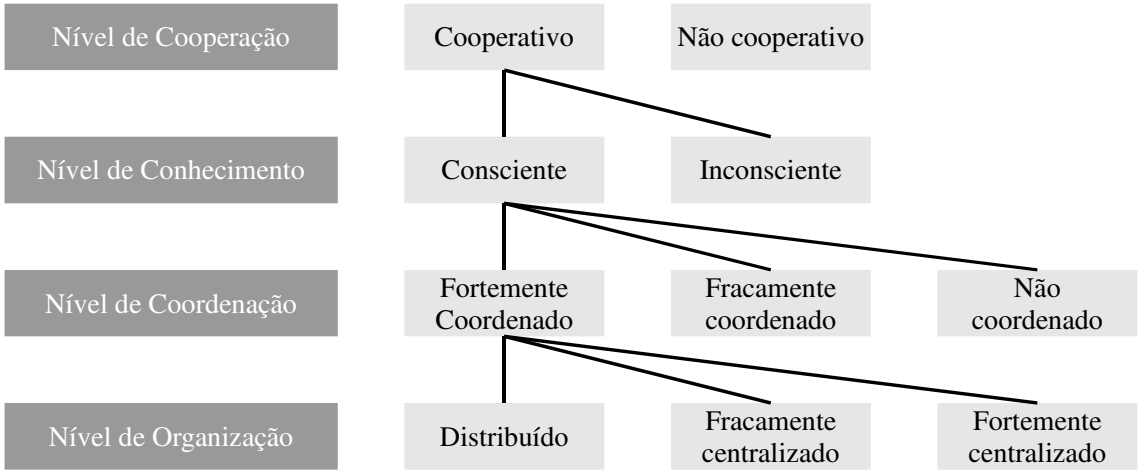
- *Tamanho do enxame*: o número de robôs define as características do enxame.
- *Faixa de comunicação*: um robô pode se comunicar com: (*i*) todos os outros robôs no enxame, que é definido como comunicação em *broadcast*; (*ii*) somente

com robôs dentro de uma distância específica, que é definido como comunicação local; (*iii*) ou os robôs não pode se comunicar diretamente, conseguindo apenas detectar a presença ou o comportamento de robôs vizinhos, o que é definido como comunicação indireta.

- *Topologia de comunicação*: os robôs podem se comunicar uns com os outros por meio de *broadcast*, usando endereços, ou através de algum tipo de hierarquia.
- *Largura de banda*: o custo de comunicação em relação ao movimento do robô.
- *Reconfiguração espacial*: a topologia de comunicação pode ser fixa, reconfigurável ou dinâmica.
- *Processamento de unidades*: a forma como os robôs processam as informações.
- *Composição*: o enxame pode ser homogêneo, quando todos os robôs são do mesmo tipo, ou heterogêneo, composto por robôs de diferentes tipos.

O conceito de *comportamento cooperativo* foi introduzido nos trabalhos de Cao *et al.* [74], com um grupo de indivíduos trabalhando juntos para alcançar algum objetivo comum. Isto contrasta com o conceito de comportamento coletivo presente em sistemas de múltiplos robôs, onde vários indivíduos se dedicam a fazer a mesma tarefa, acelerando assim o processo. Os autores classificam os estudos em robótica de enxame em cinco eixos de pesquisa: Arquitetura de Grupo, Conflito de Recursos, Origem da Cooperação, Aprendizagem e Problemas Geométricos. A *Arquitetura de Grupo* trata da infraestrutura do enxame. O *Conflito de Recursos* é o eixo de pesquisa relacionado à solução de problemas que surgem da presença de múltiplos robôs no mesmo ambiente. A *Origem da Cooperação* estuda como o comportamento cooperativo é realmente motivado e alcançado. O *Aprendizado* baseia-se na capacidade dos robôs alterarem e otimizarem seus próprios parâmetros de controle. O *Problemas Geométricos* explora aplicações e comportamentos de enxame robótico que dependem da posição do robô, da distância e de outras questões relacionadas à distribuição espacial dos robôs.

Em Iocchi *et al.* [75], uma outra taxonomia é proposta, baseada em quatro níveis, representando diferentes aspectos de robôs. Estes níveis são apresentados no diagrama da Figura 2.2. O primeiro aspecto é o *Nível de Cooperação*, seguido pelo *Nível de Conhecimento*, em que cada robô está ciente ou não da presença de outros membros do enxame. Um sistema onde os robôs conhecem sua vizinhança pode ainda ser classificado considerando um *Nível de Coordenação*, que é dividido em *Fortemente Coordenado*, *Fracamente Coordenado* e *Não Coordenado*. Para os enxames fortemente coordenados, isto é, quando um robô depende de outros robôs



i

Figura 2.2: Níveis na taxonomia proposta por Iocchi [75].

para definir o que será feito, existe um *Nível de Organização*, dividido em *Fortemente Centralizado*, *Fracamente Centralizado* e *Distribuído*. Todos os sistemas nesta taxonomia também podem ser caracterizados como *Deliberativo* ou *Reativo*.

Em Bayındır e Şahin [76], a taxonomia é dividida em cinco eixos principais: Modelagem, Projeto de Comportamento, Comunicação, Estudos Analíticos e Problemas, conforme apresentado na Figura 2.3. A *Modelagem* é o estudo de como o enxame está organizado e se o modelo de enxame é macroscópico ou microscópico. O *Projeto de Comportamentos* está relacionado com a capacidade de aprendizagem dos robôs. A *Comunicação* define se os robôs podem interagir diretamente, indiretamente ou por detecção. O *Estudo Analítico* envolve os estudos matemáticos e estatísticos do enxame. O eixo de pesquisa mais característico desta taxonomia é chamado de *Problemas*, que inclui as tarefas coletivas que um enxame de robôs irá efetivamente realizar. Em Bayındır [3], há uma revisão dos diferentes tipos de tarefas coletivas que podem ser executadas por enxames robóticos.

Uma taxonomia mais concisa foi proposta por Brambilla *et al.* [4]. De acordo com esta taxonomia, os trabalhos sobre a robótica em enxame podem ser divididos em duas classes: *Métodos* e *Comportamentos Coletivos*, como mostrado na Figura 2.4. O primeiro inclui métodos de projeto e estudos analíticos, enquanto o último está relacionado com problemas e tarefas realizadas pelos membros do enxame. Os autores enumeram os principais comportamentos coletivos e os classificam em três categorias principais: *Organização Espacial*, *Comportamentos de Navegação* e *Tomada de Decisão Coletiva*.

A análise dos trabalhos de revisão em robótica de enxame é uma forma interessante de entender não só como esse campo evoluiu ao longo dos anos, mas também como ele é interpretado por diferentes pesquisadores. Em trabalhos anteriores, houve

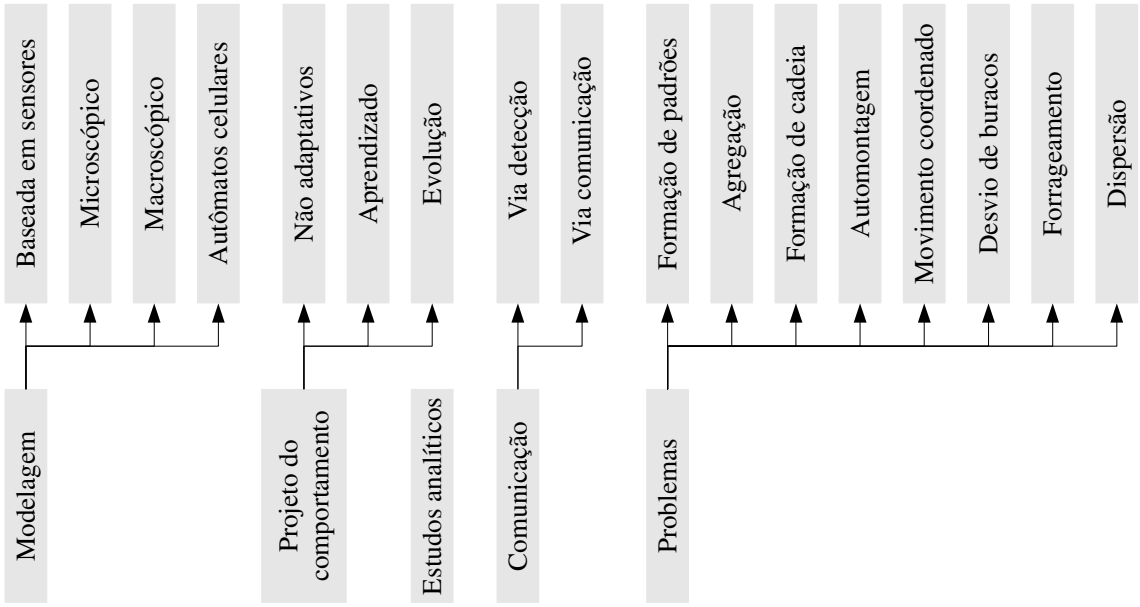


Figura 2.3: Taxonomia proposta por Bayındır e Şahin [76].

a necessidade de classificar os sistemas com base em algumas características globais, como a organização de arquitetura do enxame, como feito por Dudek *et al.* [77] e Cao *et al.* [78]. Outros autores apresentam um diferente ponto de vista, como no trabalho de Iocchi *et al.* [75], em que as características são dependentes uma da outra. Mesmo assim, essas taxonomias estão mais focadas em como os sistemas são organizados para executar uma determinada tarefa, mas não em quais são os diferentes comportamentos que os robôs executarão. É possível notar que taxonomias mais recentes classificam os estudos em robótica de enxame como trabalhos centrados em tarefas e comportamentos, onde a tarefa em si é avaliada, e trabalhos centrados no robô e no enxame, onde o foco é o *hardware* e as metodologias de projeto.

2.4 Tarefas Coletivas

Vários problemas e tarefas são estudados na pesquisa de robótica de enxame. Comportamentos básicos são geralmente usados para compor tarefas complexas que são mais próximas de problemas do mundo real. Tais comportamentos complexos podem ser usados para avaliar o desempenho de diferentes métodos propostos.

Definição 2 *Uma tarefa é uma operação realizada pelo enxame e que descreve algum comportamento coletivo. Uma tarefa possui significância em escala macroscópica, quando o enxame como um todo é considerado.*

Seguindo a divisão descrita por Brambilla *et al.* [4], as tarefas de enxame tratam, em geral, da (*i*) organização espacial entre robôs, ou entre os robôs e o ambiente, ou ambos; do (*ii*) movimento dos robôs no ambiente, incluindo a busca, identificação

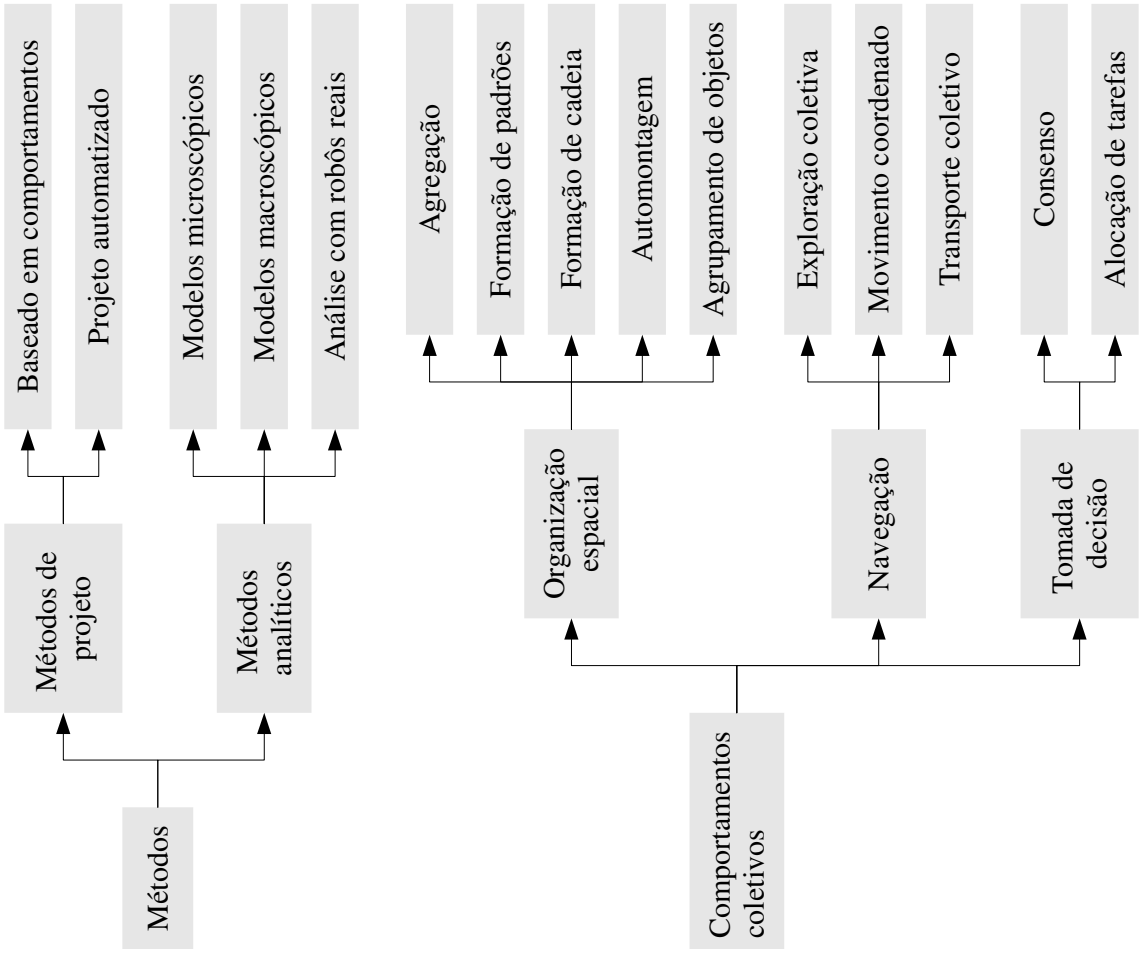


Figura 2.4: Taxonomia proposta por Brambilla *et al.* [4].

e movimento de objetos; e do processo de (*iii*) tomada de decisão coletiva, também chamado de consenso. As tarefas que se enquadram nestas três classes são descritas nesta seção.

2.4.1 Organização espacial

As tarefas que dependem da organização espacial do enxame são caracterizadas pelo movimento dos robôs, geralmente aumentando ou reduzindo a distância entre si, resultando assim no surgimento de uma propriedade particular. Algumas das tarefas de organização espacial mais conhecidas são a *agregação*, a *dispersão*, a *cobertura* e a *formação de padrões*. Todas essas tarefas necessitam que os robôs sejam cientes da existência do enxame. Para tal, cada robô deve possuir dispositivos que permitam a identificação dos demais, como por exemplo, sensores de distância IR.

Agregação

A *Agregação* é uma tarefa onde os membros do enxame, inicialmente dispersos no ambiente, devem se aproximar uns dos outros em uma mesma região. Este é um

comportamento coletivo simples presente em muitas sociedades de insetos [33]. No entanto, no contexto da robótica esta tarefa não possui solução simples, pois os robôs devem ajustar sua distância em relação aos robôs vizinhos, ao mesmo tempo que evitam a criação de grupos separados. A agregação é útil em situações onde um número significativo de robôs sé necessário em uma mesma localidade, como no caso do transporte colaborativo [79] e na formação de padrões [80].

Vários estudos apresentam diferentes abordagens para a agregação de robôs. Alguns métodos são inspirados em estratégias de agregação descritas por grupos de insetos, como abelhas [30] e baratas [81]. As abelhas apresentam um comportamento de agregação termo-tátil, permanecerem imóveis em locais de temperatura mais elevada. Em um campo com um gradiente de temperatura, cada abelha interrompe seu movimento em conjunto com outras abelhas, mas começa a se mover novamente após algum tempo que é proporcional à temperatura local. As baratas usam um comportamento similar, mas baseiam sua decisão na intensidade da luz local, agregando-se em lugares escuros. As baratas movem-se no ambiente, e param em um determinado ponto dependendo luminosidade e do número de indivíduos já existentes nessa região. Em ambos os casos, há uma informação externa que, junto com a comunicação entre os indivíduos, coordena o comportamento do enxame. Por exemplo, a estratégia desenvolvida por Schmickl e Hamann, chamada algoritmo BEECLUST [30], utiliza luz ambiente, em vez de temperatura, para permitir a agregação de robôs. No trabalho de Garnier *et al.* [81], tanto experimentos de robôs simulados quanto com robôs físicos usam lugares claros e escuros para controlar a probabilidade de agregação.

Dispersão

Um comportamento que é oposto à agregação é a *Dispersão* de robôs. Neste caso, os robôs devem se afastar uns dos outros, ocupando uma área maior que a inicial de forma cooperativa. Em outras palavras, a dispersão do enxame é normalmente necessária quando a área explorada deve ser expandida sem perda de conectividade. Existem muitas abordagens para este problema.

Em McLurkin e Smith [47], a dispersão é realizada por *SwarmBots* usando apenas comunicação inter-robô. A rede de robôs forma um gradiente, e o fluxo de mensagens guia o movimento dos robôs. Uma abordagem diferente usada nos estudos de Ludwig e Gini [82], Ugur *et al.* [83] e Necsulescu e Schilling [84] consideram a intensidade do sinal sem fio recebido para estimar a distância entre os robôs.

Um caso particular de dispersão é o problema de *cobertura de área*. Nesta tarefa, os robôs devem se espalhar, abrangendo igualmente uma determinada área sem um conhecimento prévio sobre o ambiente. É útil em muitas aplicações do mundo real, como o mapeamento de regiões desconhecidas e o uso do enxame na implementação

uma rede de sensores. Em muitos trabalhos [85–87], a cobertura do enxame usa a comunicação de insetos sociais como inspiração para regular a distância mútua entre os robôs. Nos trabalhos de Howard *et al.* [88] e Reif e Wang [89], a distância inter-robô é controlada por forças virtuais e campos potenciais.

Formação de padrões

A *Formação de Padrões* é um problema de auto-organização, onde os robôs devem se reorganizar espacialmente para alcançar algum padrão global. Através de interações locais, os robôs ajustam suas posições, formando uma estrutura regular e repetitiva. As formações auto-organizadas podem ser encontradas na natureza em diferentes cenários, tais como ninhos de abelhas [90], colônias bacterianas [91] e estrutura cristalina de sólidos [92].

O *Physicomimetics Framework* foi proposto por Spears *et al.* [80] como um controle de robô descentralizado inspirado em fenômenos físicos. Nesta estratégia, os robôs podem detectar a posição relativa de seus vizinhos, e reagir às forças atrativas e repulsivas, resultando em formações de uma rede. Em Flocchini *et al.* [93], os robôs podem descrever qualquer padrão arbitrário usando duas bússolas independentes como referência. Neste caso, há uma informação global a qual os membros do enxame possuem acesso.

Automontagem

Um exemplo extremo de auto-organização é a automontagem, onde os robôs não só permanecem próximos uns dos outros, mas também são capazes de se conectar, formando um único organismo. Esse comportamento é inspirado pela organização simbiótica encontrada na natureza, na qual espécies independentes evoluíram conectadas entre si, tendo assim uma melhor chance de sobrevivência.

O *Symbrion Project* [94] investiu um grande esforço no projeto de tarefas relacionadas com a automontagem. Seu nome denota a expressão "Organismos de Robôs Evolutivos Simbióticos". Em tal abordagem, robôs modulares operam como um enxame típico, movendo-se e explorando o ambiente. Os robôs também são capazes de se automontar para formar um organismo tridimensional. Para implementar este conceito, foi necessário desenvolver três tipos de robôs modulares, cada um com uma funcionalidade diferente, formando assim um enxame heterogêneo [95].

2.4.2 Movimentação coletiva

Os enxames de robôs são compostos por dispositivos móveis, como robôs com rodas, aéreos ou subaquáticos, que podem se deslocar em um determinado ambiente. Diferentes tarefas baseadas no movimento do enxame podem ser consideradas a partir

dessa característica básica, tais como o movimento de um único ou múltiplos robôs, a prevenção de colisões, a identificação de outros robôs ou objetos, entre outras.

Embora também haja movimento dos robôs em tarefas de organização espacial, estas possuem uma característica principal que as diferem das tarefas de movimentação coletiva. Nas tarefas de auto-organização, cada robô se move dentro do enxame visando algum arranjo. No entanto, não há deslocamento real do enxame. Por outro lado, em tarefas de movimento coletivo, é considerado que o enxame inteiro é capaz de ir de uma região para outra. Nesta seção são descritas tarefas baseadas no movimento de enxames, incluindo aquelas relacionadas à *exploração*, ao *forrageamento*, à *navegação coletiva* e ao *transporte coletivo*.

Exploração

A exploração de ambientes desconhecidos é um problema fundamental na robótica, sendo útil em muitas aplicações do mundo real em que o ambiente é perigoso para a vida humana. No caso de um único robô, este deve vagar no ambiente, registrando as informações sobre os obstáculos encontrados. Um exemplo conhecido de exploração é a tarefa de Mapeamento e Exploração Simultâneas, ou SLAM (*Simultaneous Localization and Mapping*) [96]. No SLAM, o robô deve construir um mapa da região ao redor de si com base na leitura de seus sensores, e ao mesmo tempo, deve ser capaz de se localizar dentro deste mesmo mapa.

A *Exploração Colaborativa* é realizada não por um único robô, mas por um enxame [97, 98]. Esta situação possui vantagens sobre o caso de um único robô, pois múltiplos robôs podem executar a tarefa mais rapidamente devido ao paralelismo. Além disso, a redundância de informações pode compensar a incerteza do sensor, tornando o sistema mais tolerante a eventuais falhas. Contudo, a exploração colaborativa também apresenta dificuldades, como a necessidade de otimização dos caminhos de forma que cada robô se movimente por diferentes regiões do ambiente. É necessário ainda realizar a fusão dos dados amostrados por diversos robôs, e posteriormente, disponibilizar o mapa completo para os demais.

Forrageamento

Um exemplo de tarefa mais complexa é o *Forrageamento*, onde os robôs devem encontrar e recuperar algum objeto no ambiente. Isto é inspirado diretamente no comportamento de busca por alimentos em colônias de formigas e outros insetos sociais. Sabe-se que tais insetos podem explorar eficientemente o ambiente, sendo capazes de coletar alimentos usando os caminhos mais curtos disponíveis no ambiente. As tarefas de forrageamento são normalmente identificadas como o principal teste para a robótica cooperativa [78].

De fato, uma tarefa de forrageamento completa pode ser descrita como uma sequencia de tarefas básicas, ou subtarefas: (*i*) Primeiro, os robôs vagam no ambiente, evitando colisões com os demais robôs e outros objetos que não são de interesse; (*ii*) Estes robôs devem também ser capazes de diferenciar os objetos de interesse encontrados no ambiente daqueles que não são de interesse; (*iii*) Quando os robôs encontrarem um objeto de interesse, este deve ser levado a um lugar específico, de forma analógica aos insetos que levam o alimento até seus ninhos. Diferentes implementações do forrageamento com robôs podem ser encontradas em Russel *et al.* [99], Liemhetcharat *et al.* [100] e Pitonakova *et al.* [101].

Uma extensa descrição da tarefa de forrageamento pode ser encontrada no trabalho de Winfield [102], que também define esta tarefa como um problema de referência para sistemas de robôs únicos e múltiplos. O forrageamento, seja em sistemas naturais ou artificiais, pode ser modelado como um processo repetitivo de quatro etapas: busca, captura, retorno e depósito. No entanto, o autor também aponta que, embora os princípios de forrageamento robô sejam bem compreendidos, a engenharia de seu comportamento coletivo emergente continua a ser um desafio.

Navegação coletiva

O movimento coordenado de robôs, que também é conhecido como *Navegação Coletiva*, pode ser descrito pelo movimento de um grupo de indivíduos para uma mesma direção comum, de forma agregada. Para isso, a posição dos indivíduos pode seguir uma formação particular ou ser reorganizada dinamicamente. Além disso, a direção do grupo pode ser apontada por um alvo global ou emergir da interação entre os membros do enxame.

Em Balch e Arkin [103], as diferentes abordagens do movimento coordenado são classificadas pela referência usada pelos robôs para manterem seu movimento: referenciado por uma unidade central, por um robô líder ou por um robô vizinho. Em Navarro e Matía [104], esta classificação é estendida, considerando também uma referência de múltiplos vizinhos.

O primeiro sistema artificial usando navegação coletiva foi desenvolvido na área de computação gráfica por Reynolds [105]. As animações de grupos de agentes passaram a ser conhecidas como *flocking*, em alusão ao voo coordenado de bandos de pássaros. Três regras foram usadas para descrever o movimento do grupo de agentes: o desvio de obstáculos, também conhecida como regra de separação; correspondência de velocidade, também conhecida como regra de alinhamento; e a centralização do grupo, também conhecido como regra de coesão. Em Turgut *et al.* [106], as três regras de *flocking* para movimento coordenado foram implementadas em um grupo de robôs reais. Nesse enxame, os robôs podem gerenciar colisão e velocidade com o uso de sensores de proximidade infravermelho, enquanto o alinhamento é conseguido

através de um sensor tipo bússola. O movimento coletivo foi realizado no trabalho de Erfianto e Trilaksono [107] por um enxame de robôs capazes de se comunicar por meio de mensagens ponto a ponto. Os autores analisaram o impacto da conectividade entre os robôs sobre o movimento do enxame. Em [108], o movimento conjunto de robôs é obtido pela leitura de distâncias por meio de câmera omnidirecional.

Transporte coletivo

Muitas espécies de insetos sociais, particularmente formigas, possuem a capacidade de transportar coletivamente objetos de um local para o seu ninho [33]. Esses objetos podem ser várias vezes o tamanho e o peso de uma única formiga, o que torna esta tarefa impossível para um único indivíduo. Mesmo assim, o grupo de insetos consegue executar esta tarefa de forma eficiente. O *Transporte Coletivo* é uma tarefa de grande interesse na robótica de enxame, onde um grupo de robôs fisicamente limitados interagem uns com os outros para mover um objeto para uma determinada região. Este objeto pode ser de grande dimensão em relação ao tamanho de um único robô. Kube e Bonabeau [79] descreveram e implementaram em um enxame de robôs móveis um modelo de transporte coletivo baseado no comportamento de formigas. Nesta trabalho, os robôs conseguiram coordenação sem uso de comunicação direta.

A *manipulação planar distribuída* [109, 110] é um caso particular de transporte coletivo, onde os robôs cercam um objeto de forma conhecida e o movem ao longo de uma trajetória global predefinida. Cada robô tem sua direção e força aplicadas ao objeto calculada com base na posição e orientação do objeto e dos demais robôs. Essa dependência de informações globais é um inconveniente, pois os robôs podem necessitar muito tempo de processamento para estimar a localização do objeto. Outros métodos foram projetados evitando o uso do conhecimento global. Por exemplo, em Chen *et al.* [111], uma estratégia de transporte foi desenvolvida baseada na oclusão da meta ou posição de destino. No entanto, ainda há informação global, que é a posição da meta obtida por meio de câmeras omnidirecionais. Uma estratégia descentralizada foi investigada por Rubenstein *et al.* [112]. Usando um modelo baseado em física, esta estratégia provou ser bem-sucedida no transporte de um objeto complexo para o seu destino. Neste trabalho, os robôs conhecem a direção do alvo, mas não têm informações sobre o peso e a forma do objeto. Além disso, o robô não conhece sua própria posição, nem a quantidade de robôs do enxame.

2.4.3 Tomada de decisão

A tomada de decisão coletiva é um problema onde os indivíduos devem fazer uma escolha, dado um conjunto de opções. A escolha pode ser influenciada pelos outros robôs. Esta dinâmica pode convergir para uma opinião global ou dar origem ao

surgimento de grupos de indivíduos de mesma opinião. Este tipo de tarefa é essencial na maioria dos sistemas baseados em múltiplos agentes. No contexto da robótica de enxame, o estudo deste tipo de tarefa é necessário devido à natureza distribuída do enxame, e devido à ausência de informações compartilhadas globais que possam ser usadas para influenciar em decisões locais.

Consenso

A convergência para uma escolha comum entre diferentes alternativas é chamada *Consenso*. Muitos estudos investigam esta tarefa no caso de sistemas de múltiplos robôs. Em Valentini *et al.* [113], cada robô iterativamente envia sua decisão para robôs vizinhos por um período de tempo proporcional à qualidade da opção. A tomada de decisão é controlada por uma máquina probabilística de estados finitos aumentada pela regra da maioria. Posteriormente [114], os autores investigaram o impacto da densidade espacial de robôs no processo de tomada de decisão em um enxame de cem *Kilobots*.

Alocação de tarefas

A *Alocação de tarefas* é um processo de decisão onde os robôs selecionam, a partir de uma lista de opções, a tarefa que cada um executará. Este problema também é conhecido como divisão de trabalho. O particionamento do enxame deve ser auto-organizado, em que o número de robôs que deve executar cada tarefa é definido por uma configuração global. Muitos esquemas de alocação de tarefas baseiam-se em *métodos de limiar* [115], quando um determinado valor observado não deve exceder um limiar predefinido, ou em *métodos probabilísticos* [116].

Em Mathias *et al.* [31], um algoritmo dinâmico de alocação de tarefas baseado em consenso por limiar foi proposto e implementado em um enxame de até 48 robôs Elisa-3 [117]. O esquema proposto foi bem-sucedido para atingir uma proporção de robôs predefinida. Em Nedjah *et al.* [118], outro esquema dinâmico de atribuição de tarefas é proposto baseado em um processo de otimização usando o algoritmo PSO. Neste esquema, cada robô do enxame é responsável pelos cálculos de uma partícula. O número de dimensão do espaço de pesquisa coincide com o número de tarefas. Quando o PSO converge, todos os robôs têm a mesma posição, isto é, a atribuição de tarefas que deve ser usada pelo enxame. Também neste caso, o algoritmo proposto foi implementado e validado usando o enxame de robôs Elisa-3. Por meio de interações locais e globais, essa abordagem distribuída foi eficiente para atingir uma proporção predefinida de robôs executando tarefas diferentes.

Localização

Muitas aplicações requerem que os robôs conheçam sua posição, que pode ser uma posição absoluta ou relativa. O problema de *Localização* consiste em inferir a posição de um conjunto de dispositivos quando nenhuma referência externa, como Sistema de Posicionamento Global (*Global Positioning System*, GPS), está disponível. Muitos dos métodos de localização dependem da capacidade de um robô medir sua distância para outros robôs de referência, também conhecidos como âncoras, cujas posições são conhecidas.

Um método de localização que utiliza correspondência de agrupamentos foi apresentado por Rashid *et al.* [119]. Em seu trabalho, primeiramente os robôs usam um sensor IR para avaliar as coordenadas de vários robôs, obtendo assim uma primeira topologia de rede com posições absolutas. Posteriormente, cada robô mede as distâncias e os ângulos para outros robôs dentro de uma vizinhança local, obtendo uma segunda topologia de rede, com posições relativas. Finalmente, ambas as topologias de rede são mescladas de modo que as posições absolutas são computadas para um número de robôs que é igual ou superior ao dos nós incluídos na primeira topologia de rede. Isto é assim porque a primeira varredura pode ter perdido robôs que estavam fora do alcance do sensor.

Em Sá *et al.* [120], um método de localização baseado em múltiplos saltos é proposto. A posição de cada nó é calculada usando a informação de posicionamento dos robôs de referência e a distância informada pelos robôs vizinhos. O método inclui ainda uma avaliação de confiança associada com a contribuição de cada robô vizinho no cálculo das posições. Em um estudo posterior, os autores propuseram um novo método de localização baseado em Min-Max e PSO [121].

2.5 Considerações Finais do Capítulo

Este capítulo abordou o apanhado da literatura em robótica de enxame. Vários aspectos foram apresentados e descritos, como por exemplo, a forma com que os enxames são implementados e quais tipos de operações são executadas coletivamente. Diferentes arquiteturas de enxame, isto é, especificações de *hardware*, comunicação e movimentação, possibilitam a realização de diferentes tipos de tarefas. A investigação apresentada neste capítulo possibilitou a escrita de um artigo de revisão da lteratura de robótica de enxame [13].

Dentre as plataformas estudadas, o simulador *V-REP* apresentou-se como a opção mais adequada para utilização neste trabalho, possibilitando a edição e personalização dos modelos existentes de robôs. Um modelo simulado do robôs móvel *Kilobot* foi escolhido como base para a implementação da estratégia proposta, con-

forme será apresentado nos capítulos seguintes.

Muitos dos estudos discutidos neste capítulo apresentam um grande esforço na implementação de tarefas altamente otimizadas para uma determinada arquitetura. Como resultado, observa-se várias abordagens distintas para as mesmas tarefas, e o mais importante, estratégias que não se aplicam a diferentes tarefas.

No capítulo a seguir são introduzidos os conceitos de sistemas distribuídos, usados como fundamento para a concepção da estratégia de ondas de mensagens. Considerando o enxame como um sistema distribuído de agentes conectados, as mensagens propagadas de robô em robô possibilitam a criação de uma estratégia para implementação de diferentes tipos de tarefas coletivas. Esta é uma contribuição marcante desta tese.

Capítulo 3

Algoritmos Distribuídos em Enxame de Robôs

As tarefas de enxame descritas no Capítulo 2 apresentam diferentes tipos de implementação. Muitas das vezes, isto é consequência da adoção de diferentes plataformas, cada uma contendo recursos distintos de *hardware*. Neste sentido, é de interesse o desenvolvimento de metodologias comuns a diferentes arquiteturas e que possam ser usada para implementação de tarefas diversas.

Neste trabalho, a propagação de mensagens entre robôs é reconhecida como um fundamento geral para a execução de comportamentos básicos no enxame. Cada robô é caracterizado como o elemento de um sistema distribuído, realizando algum processamento local e se comunicando com outros robôs em uma vizinhança finita. Uma contribuição desta pesquisa é a computação realizada de forma distribuída pelo enxame, inspirada no funcionamento do chamados *Algoritmos de Onda*. Esta classe de algoritmos é empregada como base para a implementação das tarefas em enxames compostos por robôs capazes interagir por meio de mensagens. Observa-se na literatura estudos em que o *software* de controle de cada robô é altamente otimizado em relação a suas característica de *hardware* e à tarefa que será realizada. Em tal abordagem, modificações no modelo de robô ou no tipo de tarefa necessitam de um projeto de controlador inteiramente novo. No presente trabalho, o principal objetivo não é a implementação de tarefas da forma mais otimizada possível, mas sim mostrar como uma mesma metodologia pode ser empregada na realização de tarefas em geral. Tarefas seguindo esta estratégia serão apresentadas no Capítulo 4.

Este capítulo segue a organização descrita a seguir. Alguns conceitos básicos relacionados com sistemas distribuídos são introduzidos na Seção 3.1, sendo relacionados com a robótica de enxame em geral. Na Seção 3.2, as características dos robôs são discutidas, com um foco maior em sua de comunicação. Por fim, a Seção 3.3 apresenta os algoritmos de propagação de informação que são a base da estratégia geral de ondas de mensagens proposta.

3.1 Sistema Robótico Distribuído

O enxame de robôs é um sistema computacional composto por diversos indivíduos. Cada robô possui seu próprio controle interno, e é capaz de interagir com o ambiente externo. Como descrito na Seção 2.3, a comunicação entre os robôs pode ser realizada de forma *direta* ou *indireta*. Embora neste trabalho os robôs façam uso de ambas, nesta seção será tratada apenas a comunicação direta, mais precisamente, o envio e recebimento de mensagens. Desta forma, o enxame pode ser descrito como um caso particular de sistema distribuído, onde os robôs são os elementos de processamento, ou *processos*. A infraestrutura de comunicação em um enxame constitui uma rede, onde a topologia é definida pela distância entre robôs e o alcance do sistema de comunicação.

Definição 3 *O enxame de robôs é um sistema de processamento distribuído constituído por um conjunto de processos executados localmente em cada robô e um sub-sistema de comunicação.*

Os sistemas que sofrem mudanças de forma discreta são usualmente chamados de *sistemas de transição* [122]. Esta nomenclatura pode ser empregada tanto para um sistema distribuído como um todo, quanto para processos individuais. Um *estado* define a condição de operação de um determinado processo. O conjunto composto pelos estados de todos os processos do sistema distribuído em um determinado instante de tempo é chamado de *configuração* do sistema ou *estado global*. A alteração do estado caracteriza um *evento* em um processo, que por sua vez define uma *transição* no sistema.

Um algoritmo distribuído é o *software* executado de forma concorrente nos diferentes processos de um sistema distribuído. Este paradigma computacional está diretamente relacionado com a exploração paralela de recursos de *hardware* e com o gerenciamento de informações originadas de fontes distintas. Em um enxame de robôs, é possível afirmar que a realização coletiva de uma tarefa é equivalente à computação distribuída de um algoritmo. Em um enxame homogêneo, isto é, onde todos os robôs são iguais, o mesmo algoritmo é empregado localmente em cada robô. Possivelmente, diferentes robôs experimentam diferentes execuções do algoritmo. Isto se deve aos diferentes momentos referentes às tomadas de decisão de acordo com o recebimento de mensagens e sensoriamento do ambiente. Contudo, a programação que constitui o algoritmo local é idêntica para todos os robôs.

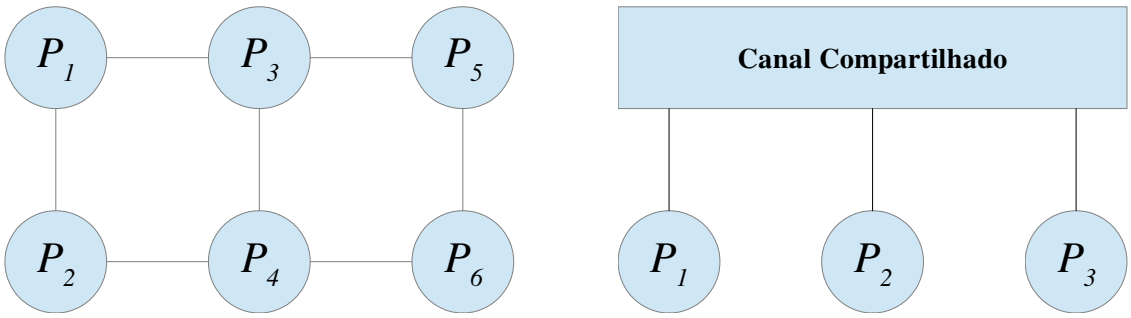
3.1.1 Comunicação por Passagem de Mensagens

A comunicação por mensagens em um sistema distribuído ocorre com a geração de dados em um processo de origem e com o consumo destes mesmos dados em um

processo de destino. Um *sistema de comunicação* interliga os diferentes processos e possibilita a transmissão das mensagens. Este sistema pode ser caracterizado de diversas formas, conforme a organização dos canais de comunicação. No estudo de sistemas distribuídos, os esquemas de comunicação são usualmente classificados como redes *ponto-a-ponto* e redes *broadcast* [123].

Em um sistema que faz uso de comunicação *ponto-a-ponto*, um canal de comunicação bidirecional dedicado permite a transmissão de mensagens entre dois processos. Conforme a especificação do sistema, cada processo pode possuir um ou mais canais que o conectam a outros processos. Da mesma forma, um processo pode se comunicar diretamente com diversos outros, mas não necessariamente com os demais. Um sistema ponto-a-ponto pode ser representado por um grafo conexo, onde os nós representam os processos, e os arcos representam canais de comunicação. A Figura 3.1(a) apresenta um exemplo de sistema ponto-a-ponto com seis processos. P_1 é capaz de enviar e receber mensagens diretamente de P_2 e P_3, mas não de P_4, P_5 e P_6. Para tal, uma eventual mensagem originada em P_1 e destinada a P_6 precisará ser retransmitida pelos processos intermediários.

Na comunicação por *broadcast*, um processo é capaz de enviar mensagens diretamente para todos os processos que constituem o sistema. Isto é alcançado por meio de um canal de comunicação compartilhado. Na representação da Figura 3.1(b), uma mensagem originada no processo P_1 pode ser recebida tanto por P_2 quanto por P_3. Contudo, por se tratar de uma estrutura compartilhada, existe o problema de colisões de mensagens durante o uso simultâneo de um canal por mais de um processo.



(a) Ponto-a-ponto (b) *Broadcast*

Figura 3.1: Exemplo de sistemas com diferentes esquemas de comunicação.

Em um enxame de robôs, a transmissão de mensagens está diretamente relacionada com os dispositivos empregados para comunicação. Na Seção 2.2 observou-se que modelos diferentes de robôs de enxame usam variados dispositivos de comunicação. Por exemplo, o robô *E-puck* suporta a transmissão de mensagens por meio de sensores IR. Estes sensores são altamente direcionados, e estão dispostos ao redor do corpo do robô. Neste caso, cada sensor pode operar como um canal

ponto-a-ponto independente, desde que os sensores em dois robôs distintos estejam alinhados e dentro de sua área de alcance. Os robôs com interface *bluetooth* podem usá-la para realizar uma comunicação *broadcast*: robôs conectados a um computador remoto podem enviar mensagens para um ou mais robôs no enxame usando o próprio computador como canal compartilhado.

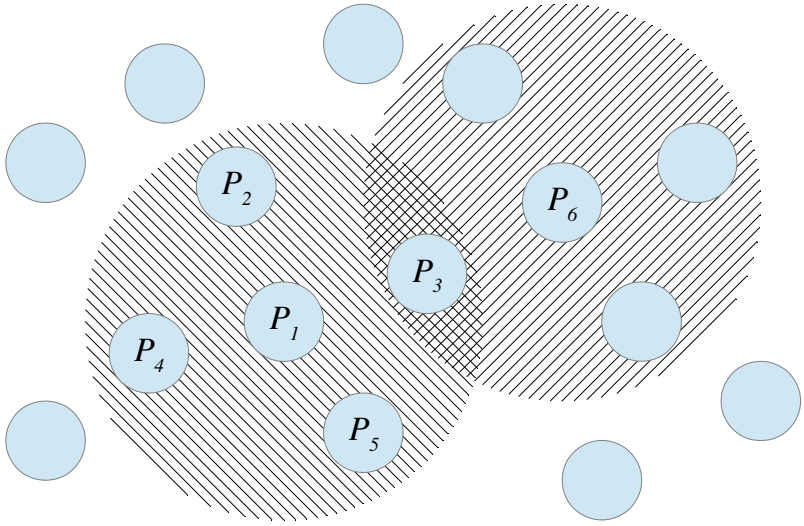


Figura 3.2: Exemplo de comunicação *multicast*: as áreas hachuradas representam a vizinhança dos processos P_1 e P_6.

Um esquema particular de comunicação que será empregado no decorrer deste trabalho é chamado de *multicast*. Neste esquema, cada processo é capaz de transmitir uma mesma mensagem para um conjunto específico de processos, que por sua vez são um subconjunto do conjunto de processos que compõem o sistema. Este mecanismo é exemplificado na Figura 3.2. Uma mensagem transmitida pelo processo P_1 pode ser recebida pelos processos P_2, P_3, P_4 e P_5, mas não por P_6. Já o processo P_3 pode receber mensagens tanto de P_1 quanto de P_6. A comunicação *multicast* retrata sistemas onde há limitação no alcance de transmissão de mensagens. Por exemplo, na Figura 3.2, a área hachurada ilustra o raio de alcance do sistema de comunicação. Esta área será chamada de *vizinhança* no decorrer deste trabalho. O *broadcast* pode ser dito como sendo um caso particular de *multicast*, onde todos os processos se encontram em uma mesma vizinhança.

O robô *Kilobot* [11] utiliza um mecanismo de transmissão de mensagens que pode ser caracterizado como um sistema *multicast*. Um dispositivo IR localizado no centro do corpo do robô é capaz de enviar dados para outros robôs dispostos em uma área finita em seu entorno, ou seja, em sua vizinhança.

3.1.2 Sistema Assíncrono

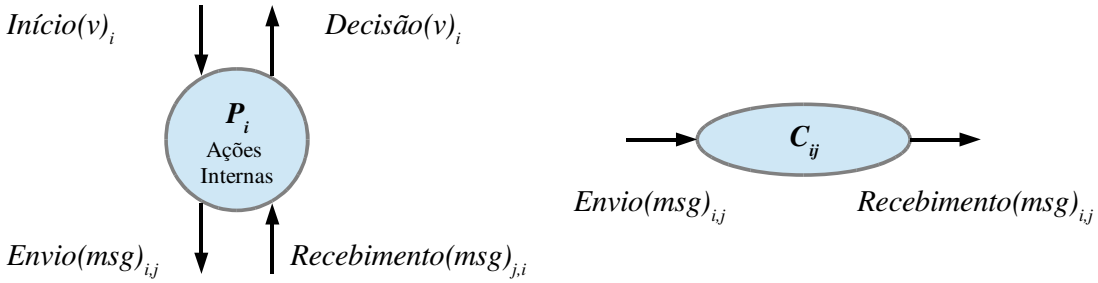
Um enxame de robôs constitui um *sistema assíncrono*. Em um sistema síncrono, as operações realizadas por diferentes processos são coordenadas por um mesmo relógio central. Em contrapartida, um sistema distribuído assíncrono é caracterizado pela ausência de um relógio global. A coordenação em um sistema assíncrono não depende de uma temporização específica, como ocorre em um sistema síncrono, mas sim pela ocorrência de eventos. Por exemplo, um evento pode ser caracterizado pelo recebimento de mensagens, ou por computações locais que acarretem em uma mudança de estado do processo. Na prática, todos os sistemas distribuídos reais são assíncronos, pois as mensagens entre dois processos sofrem um atraso finito, porém indeterminado [124].

Um processo em um sistema assíncrono pode ser descrito de forma geral como um autômato E/S (entrada e saída) [123]. Na Figura 3.3(a), o autômato E/S representa um processo P_i em um sistema distribuído assíncrono. Esta é uma representação gráfica abstrata similar a uma máquina de estados, cujas transições são chamadas *ações*. As ações podem ser do tipo entrada, saída, ou internas. Ações de entrada e saída são usadas como comunicação com o ambiente, enquanto que as ações internas são visíveis apenas pelo próprio autômato. O autômato não possui controle sobre ações de entrada, apenas sobre ações de saída e internas. Os elementos do autômato ilustrado na Figura 3.3(a) podem ser descritos como:

- P_i: Processo local do sistema distribuído assíncrono capaz de interagir com o ambiente a com outros processos.
- *Ações Internas*$(v)_i$: Eventos que mudam o estado interno de P_i e que são visíveis apenas pelo processo local P_i.
- *Início*$(v)_i$: Recebimento do valor v do ambiente externo indicando o início de operação do processo.
- *Decisão*$(v)_i$: Indicação para o ambiente externo de alguma decisão, caracterizada por v, realizada pelo processo P_i.
- *Envio*$(msg)_{i,j}$: O processo local P_i envia a mensagem msg_i para o processo remoto P_j.
- *Recebimento*$(msg)_{j,i}$ O processo local P_i recebe a mensagem msg enviada pelo processo remoto P_j.

Um autômato E/S também pode ser empregado para representar o canal de comunicação entre dois processos, como mostrado na Figura 3.3(b). O envio de

mensagens requer um canal, representado por $C_{i,j}$, que receba a mensagem produzida por P_i e a entregue a P_j. Neste trabalho, é assumido que todos os canais de comunicação são do tipo FIFO (*first in, first out*), ou seja, a primeira mensagem a ser recebida por $C_{i,j}$ produzida por P_i é a primeira mensagem entregue à P_j. Em outras palavras, é um canal de comunicação onde não há reordenação de mensagens.



(a) Processo (b) Canal de comunicação

Figura 3.3: Processo e canal de comunicação em um sistema assíncrono representados como autômatos E/S.

Em um sistema empregando comunicação *multicast*, a representação por um autômato E/S é levemente diferente. As ações que envolvem mensagens devem considerar a transmissão não apenas para um único destino, mas para um conjunto I de processos que podem interagir com P_i por meio de envio e recebimento de mensagens. Assim, têm-se:

- $Envio(msg)_{i,I}$: O processo local envia a mensagem msg para todos os processos contidos no conjunto I.
- $Recebimento(msg)_{I,i}$: O processo local P_i recebe a mensagem msg enviada pelo processo remoto P_j, onde $P_j \in I$.

É preciso ressaltar que as ações em um sistema modelado por autômatos E/S ocorrem com temporizações independentes para diferentes processos e canais, que por sua vez, é a principal característica do sistema assíncrono. As ações internas em um determinado processo possuem sua execução coordenada por um relógio local. Contudo, não é possível determinar o tempo necessário para um processo interagir com o ambiente durante as ações de entrada e saída.

3.2 Características dos Robôs

Um robôs de enxame possui várias características que permitem sua operação como um processo em um sistema distribuído. A execução de um algoritmo pelo enxame

de robôs, em um nível de abstração muito elevado, pode ser descrito pelo Algoritmo 1. Cada robô está constantemente repetindo a sequência de ações envolvendo a verificação do recebimento de mensagens e a reação a este recebimento. Em uma descrição mais detalhada de algum algoritmo específico, é necessário apontar quais mensagens podem ser enviadas e recebidas, quais ações são realizadas como reação a cada mensagem, qual a vizinhança de cada robô, quais estados os robôs podem assumir, entre outros.

Algoritmo 1 Computação local em um robô.

```
1: enquanto Em funcionamento faça
2:     Verifique o recebimento de mensagens;
3:     se Nova mensagem então
4:         Realize alguma tipo de ação;
5:     senão
6:         Realize outro tipo de ação;
7:     fim se
8: fim enquanto
```

3.2.1 Identificação Individual

Existem sistemas distribuídos em que o processo receptor não conhece o processo de origem das mensagens que recebe. Estes são conhecidos por *sistemas anônimos* e conseguem realizar computações graças ao conhecimento de quantos processos constituem o sistema. Nos enxames tratados neste trabalho, a estratégia é oposta: os robôs não precisam saber a dimensão total do enxame, mas devem ser capazes de se distinguir uns dos outros. Para isso, cada robô possui uma identificação individual, que é chamada no decorrer deste trabalho de *id*. O *id* geralmente é enviado como parte da mensagem, para que os demais robôs possam identificar sua origem.

3.2.2 Robô Inicializador

O enxame é um sistema assíncrono, e seu funcionamento depende da ocorrência de eventos. Contudo, se todos os robôs precisarem aguardar o recebimento de uma primeira mensagem para só então iniciar o envio de outras mensagens, então o sistema nunca iniciará seu funcionamento. É preciso diferenciar um dos robôs para que este inicie o envio de mensagens no enxame de forma espontânea. O robô que começa o envio de mensagens é chamado de *inicializador*, enquanto que os demais são chamados de *não-inicializadores*. Um robô pode ser definido como inicializador de forma explícita através de seu *id*, ou como reação ao ambiente. Um exemplo simples de inicialização local de cada robô é apresentado no Algoritmo 2. Neste exemplo, o valor do *id* define se um robô é inicializador ou não-inicializador.

Algoritmo 2 Definição do inicializador.

```
1: Verifica o próprio id;
2: se Id = 1 então
3:    Realiza ações como inicializador;
4: senão
5:    Realiza ações como não-inicializador;
6: fim se
```

3.2.3 Conhecimento da Vizinhança

Cada robô deve conhecer a identificação dos robôs que compõem sua vizinhança. Esta condição é essencial, pois em diversos casos um robô só deverá realizar uma ação após receber mensagens de todos os vizinhos. Assim, cada robô possui em sua memória interna a lista com os *ids* dos robôs vizinhos para os quais este pode enviar e receber mensagens. Esta lista de robôs pode ser construída de diversas formas, sendo três delas descritas a seguir.

1. A definição da vizinhança de um robô pode ser definida de forma explícita pelo projetista. Durante a gravação de sua ROM (*read only memory*, memória de somente leitura), cada robô pode receber junto com seu algoritmo local uma lista com os *ids* dos robôs que irão compor sua vizinhança. Embora esta seja uma solução simples, é necessário que durante a inicialização do enxame os robôs sejam posicionados espacialmente de forma a atender a vizinhança predefinida.

2. A vizinhança pode ser descoberta pelos próprios robôs, por meio de um algoritmo distribuído. Cada robô envia um tipo específico de mensagem contendo seu próprio *id*, ao mesmo tempo que aguarda mensagens deste mesmo tipo enviadas por robôs vizinhos. Nesta estratégia, resumida no Algoritmo 3, é considerado que todos os robôs iniciam sua operação simultaneamente. O tempo de espera para o recebimento de mensagens deve ser suficiente para que as vizinhanças sejam recíprocas, isto é, se um robô R_1 recebe a mensagem com a identificação de R_2, então o robô R_2 obrigatoriamente deve receber uma mensagem com a identificação de R_1.

3. Os robôs podem possuir dispositivos dedicados para a detecção da vizinhança. Por exemplo, o robô *s-bot* [45] possui em sua parte superior uma câmera omnidirecional com o qual é possível identificar a posição relativa de outros robôs ao seu redor. Assim, cada robô é capaz de identificar sua vizinhança com base na distância para os demais robôs.

As três abordagens para determinação da vizinhança listadas acima apresentam vantagens e desvantagens. O uso de uma tabela predefinida reduz o custo compu-

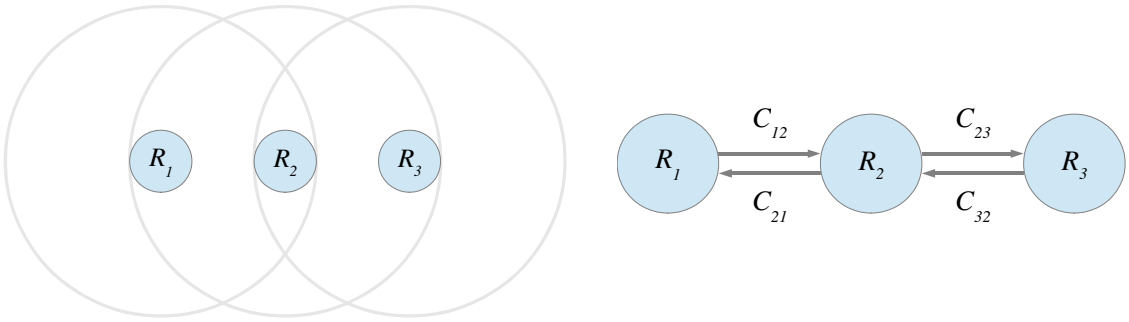
Algoritmo 3 Descobrimento da vizinhança.

```
1: Envie mensagem contendo id;
2: enquanto Contador < tempo_limite faça
3:    Aguarde o recebimento de mensagens;
4:    se Nova mensagem então
5:       Salve o id recebido na lista de vizinhos;
6:    fim se
7:    Incremente o Contador;
8: fim enquanto
```

tacional, mas necessita de uma disposição estática dos robôs. O uso de mensagens possibilita o descobrimento das vizinhanças independente da distribuição espacial dos robôs, mas requer uma inicialização simultânea dos robôs. Também neste caso é necessário um certo período para a transmissão e recebimento de mensagens. A terceira estratégia identifica os vizinhos sem uso de mensagens, mas necessita de um aparato de *hardware* dedicado para tal tarefa.

3.2.4 Representação da Topologia de Comunicação

A vizinhança de um robô é definida pela área de alcance do dispositivo de comunicação. Assumindo esta área igual para todos os membros do enxame, é possível considerar que a vizinhança de um robô é função de sua distância para os demais. Na Figura 3.4(a), três robôs são representados pelas circunferências internas R_1, R_2 e R_3. As circunferências externas representam a área de comunicação de cada robô. Desta forma, é possível afirmar que os robôs R_1 e R_3 são vizinhos de R_2, mas não são vizinhos um do outro. Com a vizinhança de cada robô definida, a comunicação no enxame é representada na Figura 3.4(b) por um grafo $G = (V, A)$, onde $V = \{R_1, R_2, R_3\}$ é o conjunto de nós associados com a computação em cada robô, e $A = \{C_{12}, C_{21}, C_{23}, C_{32}\}$ é o conjunto de canais de comunicação entre dois robôs vizinhos.



(a) Alcance de comunicação (b) Canais de comunicação

Figura 3.4: Representação da comunicação entre três robôs.

3.2.5 Gerenciamento de Mensagens

Uma mensagem em um enxame de robôs é definida como um sinal sem-fio produzido e transmitido por um robô e recebido e interpretado por um ou mais robôs. Neste trabalho, o enxame é composto por robôs *Kilobot* [11], conforme detalhado posteriormente no Capítulo 4 e 5. Desta forma, é necessário apresentar as principais características sobre a estrutura e uso das mensagens por esse modelo de robô.

Uma mensagem completa na plataforma *Kilobot* é composta por pacotes de 5 *bytes*, sendo apenas 3 *bytes* livres para alteração pelo usuário. Como esquematizado na Figura 3.5, o usuário possui acesso a 23 *bits*, enquanto que o *bit* menos significativo, bem como os demais *bytes*, é de acesso restrito ao robô. A mensagem, na maioria dos casos, é dividida em 3 campos, msg_1, msg_2 e msg_3. Os dois primeiros campos podem representar 256 valores distintos, enquanto que o terceiro campo é capaz de representar somente 128 valores. Devido à restrição na quantidade de informação, as mensagens são geralmente usadas para transmissão de dados simples, como o *id* dos robôs de origem e de destino e o tipo de mensagem.

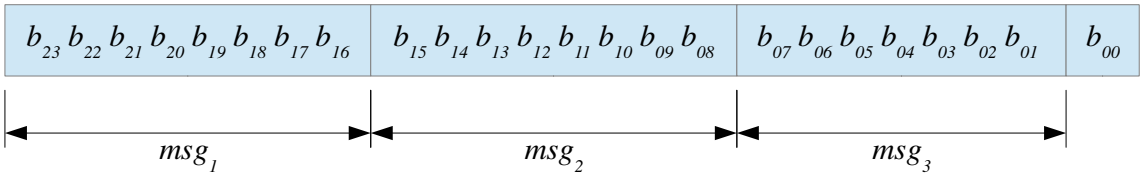
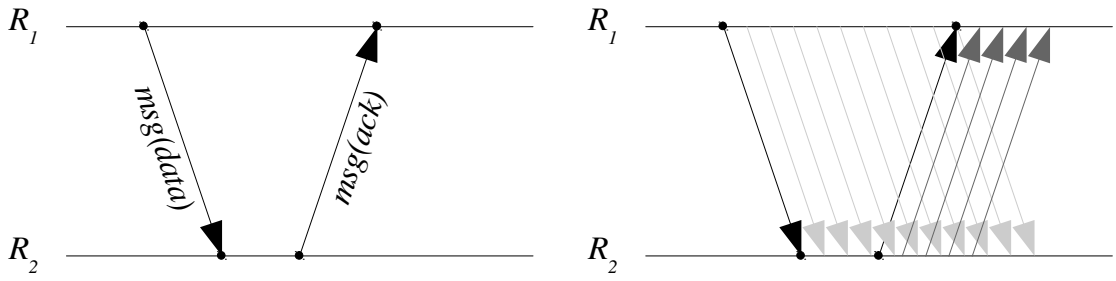


Figura 3.5: Representação dos três campos da mensagem no robô *Kilobot*.

Após o início do envio, o robô transmite a mesma mensagem repetidas vezes, para garantir que esta seja recebida pela vizinhança. A transmissão será encerrada quando o robô alterar seu estado, possivelmente em resposta à mensagens de confirmação enviadas pela vizinhança. A troca de mensagens entre dois robôs é ilustrada pela Figura 3.6(a). Nesta representação, também conhecida como *grafo de precedência* [124], *diagrama de envio e recebimento* [123] ou ainda, *diagrama de tempo-espaço* [122], as linhas horizontais representam o tempo local em cada um dos robôs R_1 e R_2. Os tempos locais progridem da esquerda para a direita. As setas representam a mensagem $msg(data)$ que contém alguma informação enviada por R_1 e recebida por R_2 e a mensagem de aceitação (*ack, acknowledgment*) enviada por R_2 e recebida por R_1. O intervalo de tempo entre o recebimento e o envio da aceitação, em R_2, ilustra que a resposta pode não ser imediata, dependendo também de algum processamento local em R_2.

Contudo, a Figura 3.6(a) ilustra apenas as mensagens efetivamente recebidas pelos robôs. Conforme descrito antes, cada robô envia a mesma mensagem repetidas vezes, conforme representado pelas setas claras na Figura 3.6(b). O robô R_2 recebe a primeira mensagem vinda de R_1, e ignora todas as demais repetições. Já o robô



(a) Mensagens recebidas (b) Todas as mensagens enviadas

Figura 3.6: Diagramas de envio e recebimento de mensagens.

R_1 interrompe o envio ao receber a primeira mensagem vinda de R_2, ignorando as demais. O recebimento de $msg(ack)$ por R_1 indica que este robô pode enviar uma nova mensagem de dados para R_2, reiniciando assim o mecanismo de aceitação.

O robô *Kilobot* é capaz ainda de lidar com o recebimento de mensagens vindas de diversos robôs distintos. O *hardware* responsável pela comunicação é composto por um par de dispositivos IR composto por um LED (*light emitting diode*, ou diodo emissor de luz) para transmissão e um fototransistor para recepção. O mesmo canal de comunicação IR é compartilhado por diversos robôs em uma mesma vizinhança, o que poderia eventualmente resultar em colisões entre mensagens quando mais de um robô transmitem simultaneamente. Segundo seus desenvolvedores, o *Kilobot* faz uso de métodos de transmissão que conseguem reduzir o efeito de perdas de mensagens [11]. Assim, é considerado que no esquema de comunicação empregado por este modelo de robô não ocorre o efeito de *starvation* ou inanição, quando um determinado robô não recebe uma ou mais mensagens pois outras mensagens possuem prioridade no uso do canal IR compartilhado.

3.2.6 Sequência de Estados

O algoritmo local em cada robô é estruturado na forma de uma *máquina de estados finitos* (FSM, *finite state machine*) . Sua execução é realizada de forma repetitiva e em cada iteração é dito que o algoritmo está em um estado específico. Uma FSM é descrita graficamente por um diagrama de estados.

A Figura 3.7 apresenta os principais elementos da representação isolada de um estado qualquer S_i. Estados em que a ocorrência um evento resulta na transição para S_i são chamados estados anteriores (setas escuras incidentes em S_i). Os estados para os quais o algoritmo pode sofrer transição quando em S_i são chamados estados posteriores (setas escuras originadas em S_i). Também é possível que existam ações de entrada e saída associadas com S_i, como o recebimento e o envio de mensagens (setas pontilhadas).

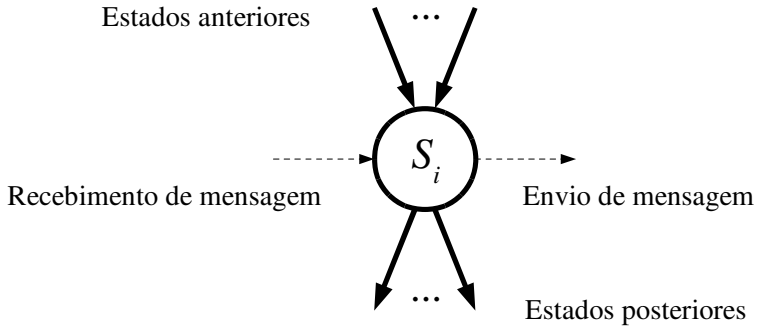


Figura 3.7: Estado qualquer S_i que compõe uma FSM.

O Algoritmo 4 é um exemplo de máquina de estados. Seu diagrama de estados é descrito na Figura 3.8. Este algoritmo realiza dois comportamentos alternados, representados pelos estados S_0 e S_1. Em toda FSM, um dos estados é definido como sendo o *estado inicial*, que é o estado em que o sistema se encontra quando entra em operação. Neste exemplo, S_0 é o estado inicial. As ações associadas com o estado S_0 são executadas apenas quando a condição A é atendida, e então ocorre a transição $S_0 \rightarrow S_1$. Caso A não ocorra, o algoritmo permanece em S_0 sem que as ações sejam realizadas. O análogo ocorrem em S_1: as ações associadas com o estado são realizadas apenas com a ocorrência da condição B, seguido da transição $S_1 \rightarrow S_0$.

Algoritmo 4 Exemplo de FSM composta por dois estados.

```
1:  ESTADO := S0; {Inicialização}
2:  enquanto verdadeiro faça
3:     se ESTADO = S0 então
4:        se Condição A então
5:           Realize ações do estado S0;
6:           ESTADO := S1; {Transição para o estado seguinte}
7:        fim se
8:     senão se ESTADO = S1 então
9:        se Condição B então
10:          Realize ações do estado S1;
11:          ESTADO := S0; {Transição para o estado seguinte}
12:       fim se
13:    fim se
14: fim enquanto
```

A representação por diagrama de estados, mesmo não apresentando os detalhes de seu equivalente em pseudo-código, é útil para descrever algoritmos caracterizados por uma sequência de transições. Dessa forma, esta representação será adotada para descrever os comportamentos que serão apresentados no Capítulo 4.

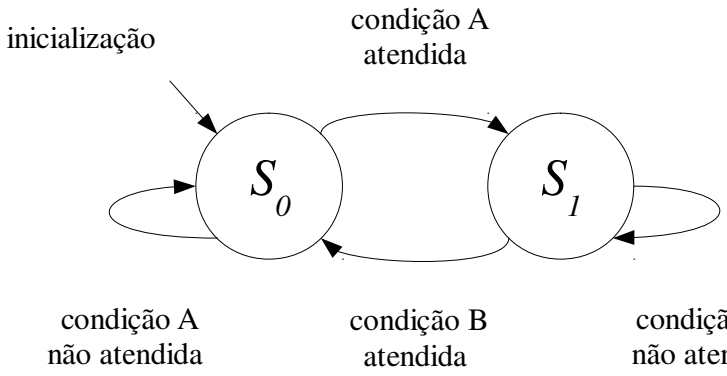


Figura 3.8: Diagrama de estados da FSM descrita pelo Algoritmo 4.

3.3 Algoritmos de Ondas de Mensagens

Os Algoritmos de Onda (*Wave Algorithms*) são uma classe de algoritmos que gerenciam a passagem de mensagens em sistemas distribuídos [122]. Tais algoritmos são usados como base para tarefas que necessitam de um esquema organizado de transmissão de mensagens. Em muitas situações, pode ser necessário que uma determinada informação seja conhecida por todos os processos em um sistema onde não há comunicação *broadcast*, como por exemplo, o enxame descrito na Seção 3.2. A informação a ser propagada pode ser um valor numérico a ser usado em alguma computação, ou um marcador (*token*) identificando que alguma ação local deve ser iniciada ou interrompida. Os algoritmos de onda lidam exatamente com este tipo de situação. Um inicializador começa o envio de mensagens espontaneamente, o que acarreta na ocorrência de eventos em todo o sistema. Este tipo de computação é denominada onda de mensagens.

3.3.1 Características Gerais

Para que sejam executados, os algoritmos de onda necessitam primeiramente de uma arquitetura composta por dispositivos que suportem o envio e recebimento de mensagens. O sistema robótico descrito na Seção 3.2 apresenta características necessárias para o uso de ondas de mensagens:

- A estrutura de comunicação do enxame constitui uma rede cuja topologia definida por um grafo $G = (V, A)$, onde V é o conjunto de robôs e A é o conjunto de canais de comunicação entre os robôs.
- A topologia não sofre alterações no decorrer da execução do algoritmo.
- A rede é conexa, isto é, todo robô é capaz de transmitir informações direta ou indiretamente para qualquer outro robô do enxame.

- Os robôs possuem algum conhecimento inicial, como a própria identificação ou a identificação dos robôs vizinhos.

A computação de uma onda possui terminação, isto é, a computação é finita. O envio de mensagens ocorre como reação ao recebimento de mensagens enviadas por robôs vizinhos. É considerado que existe um evento específico, chamado *evento de decisão*, que é causalmente precedido por eventos ocorridos nos demais robôs. Em outras palavras, na computação de uma onda ocorre a transmissão de um número finito de mensagens e então uma decisão é tomada com base nos eventos ocorridos em cada um dos processos.

3.3.2 Propagação de Informação com Realimentação

Um algoritmo de onda pode ser empregado para realizar a propagação de informações [9] em uma rede de processos. Esta operação é essencial em sistemas onde não há *broadcast* de mensagens, como no caso de um enxame de robôs que possuem apenas comunicação local.

Na propagação de informação centralizada, um único processo possui a informação que precisa ser conhecida pelos demais processos. A informação pode ser enviada para outros processos por meio de mensagens, conforme descrito pelo Algoritmo 5.

Algoritmo 5 Propagação de informação.

```
1: se Inicializador então
2:     Envie a mensagem contendo a informação para todos os vizinhos;
3: senão
4:     Aguarde o recebimento de mensagens;
5:     se Primeira mensagem contendo a informação então
6:         Envie a mensagem contendo a informação para todos os vizinhos;
7:     fim se
8: fim se
```

Durante a propagação de informação realizada conforme o Algoritmo 5, um processo inicializador contém a informação e a envia espontaneamente para os processos vizinhos. Os processos vizinhos, bem como os demais não-inicializadores, reagem ao recebimento da mensagem enviando uma cópia da informação para todos os robôs da sua própria vizinhança. Como a rede é conexa, a informação será repassada de vizinho em vizinho, alcançando assim todos os processos que constituem o sistema. Com exceção do inicializador, todos os processos recebem a informação de algum outro processo. Um processo P_A que envia a informação é chamado de *Pai* do processo P_B que recebe a informação, ou simplesmente Pai_B. De forma recíproca, P_B faz parte do conjunto de *filhos* P_A, ou $Filhos_A$.

Definição 4 *A execução de um algoritmo de propagação de informação resulta no surgimento de relações pai-filhos entre processos vizinhos.O processo de origem da primeira mensagem recebida por P_i passa a ser identificado como Pai_i. As relações pai-filhos formam uma árvore no grafo que descreve a topologia do sistema onde o algoritmo é executado, tendo com raiz o processo inicializador.*

Durante a propagação de informação, cada processo conclui sua participação no algoritmo com o envio de mensagens para sua vizinhança. Um processo pode saber que seus vizinhos receberam a informação pelo recebimento de mensagens de confirmação. Contudo, não há confirmação do recebimento da informação por outros processos que não façam parte de sua vizinhança. A Propagação de Informação com Realimentação (*Propagation of Information with Feedback*, PIF) [9] é um algoritmo de onda empregado na transmissão de uma determinada informação e que possui um evento que indica que tal informação foi recebida por todos os processos.

Algoritmo 6 Propagação de informação com realimentação

Entrada: Lista de vizinhos;
Saída: Pai, Lista de filhos;
1: **se** Inicializador **então**
2: Envie a mensagem contendo a informação para os vizinhos;
3: Receba mensagens de realimentação de todos os vizinhos;
4: Realize o evento de decisão;
5: **senão**
6: Receba mensagens;
7: **se** Primeira mensagem contendo a informação **então**
8: Defina o remetente como *Pai*;
9: Envie a informação para os vizinhos, exceto o *Pai*;
10: **senão se** Mensagem de realimentação **então**
11: Acrescente o remetente à lista de filhos;
12: **fim se**
13: Envie a mensagem de realimentação para o *Pai*;
14: Aguarde a ocorrência do evento de decisão no inicializador;
15: **fim se**

O funcionamento do PIF é descrito pelo Algoritmo 6. Sua principal diferença em relação à propagação descrita no Algoritmo 5 está na existência de dois tipos de mensagens. As mensagens contendo a informação são chamadas de *mensagens de propagação*. O funcionamento deste tipo de mensagens é idêntico ao do Algoritmo 5: um inicializador começa a propagação espontaneamente, enquanto que os não-inicializadores enviam mensagens como reação ao recebimento da informação. Já as *mensagens de realimentação* são inicialmente enviadas pelos processos que não possuem filhos (folhas da árvore no grafo) para seus respectivos processos pais. Os demais processos intermediários enviam a realimentação para os pais após receberem

a realimentação vinda dos filhos. O processo inicializador, após receber este tipo de mensagem de todos os processos filhos, realiza algum evento de decisão. Como a decisão é causalmente precedida pela realimentação enviada por todos os processos não-inicializadores, é possível afirmar que o inicializador tem conhecimento do término da propagação.

A complexidade de mensagens em um algoritmo distribuído é a contagem de mensagens transmitidas pelos processos durante sua execução. A complexidade de tempo é o tempo necessário para o término do algoritmo. Considerando a inexistência de um relógio global em um sistema assíncrono, a complexidade de tempo assíncrono é definida como sendo o tempo total de propagação de mensagens na maior cadeia causal envolvendo mensagens que são enviadas em resposta a mensagens que são recebidas. Dessa forma, a complexidade de mensagens do PIF é $O(|E|)$, onde $|E|$ é a quantidade de canais de comunicação. O número de mensagens transmitidas é $2|E|$, pois 1 mensagem de propagação e 1 mensagem de realimentação são transmitidas em cada canal de comunicação. A complexidade de tempo assíncrono (ou complexidade de tempo local no processo inicializador) no PIF é $O(N)$, onde N é a quantidade de processos. Esta propriedade pode ser verificada no caso em que todos os processos possuem no máximo um filho. Este é o pior caso em termos de complexidade de tempo, pois a propagação percorre sequencialmente todos os processos.

3.3.3 Sincronização

A onda de mensagens descrita pelo Algoritmo 6 para a propagação de informação com realimentação também pode ser empregada na *Sincronização* de algoritmos [125]. Em um sistema distribuído, pode ser de interesse que ações do tipo a_1 só ocorram após a ocorrência de ações do tipo a_0 em todos os processos. Em outras palavras, nenhuma ação a_1 deve ocorrer enquanto a_0 ainda não tiver sido encerradas em todos os processos.

A sincronização pode ser obtida por sucessivas execuções do Algoritmo 6. As mensagens de propagação não transmitem informação, mas sim a indicação de que uma determinada ação a_0 pode ocorrer localmente. O evento de decisão (após a transmissão da realimentação) faz com que uma nova onda seja propagada pelo inicializador, de forma que cada processo possa, então, executar uma ação a_1. É possível verificar que a_1 ocorre posteriormente ao evento de decisão, que por sua vez, é causalmente dependente da ocorrência de a_0 em todos os processos.

3.4 Considerações Finais do Capítulo

Neste capítulo foram abordados diversos conceitos referentes a sistemas distribuídos e como tais conceitos podem ser associados à arquitetura de comunicação apresentada por um enxame de robôs. Para um enxame composto por robôs que possuam comunicação *multicast*, o conhecimento da vizinhança permite a modelagem de algoritmos assíncronos por passagem de mensagens. Estas características de arquitetura são baseadas no robô *Kilobot*, cujo modelo simulado a base no Capítulo 4. A comunicação por passagem de mensagens possibilita a realização de algoritmos de onda, sumarizado neste capítulo pela propagação de informação com realimentação ou algoritmo PIF.

O capítulo seguinte apresenta a realização tarefas de enxame baseando-se estratégia de ondas de mensagens.

Capítulo 4

Tarefas baseadas em Ondas de Mensagens

Os algoritmos de onda possuem como uso imediato a propagação de informação e a sincronização de eventos em processos computacionais, tal como visto no Capítulo 3. Estes dois conceitos podem ser empregados como base para realização de tarefas em enxames de robôs. A proposta deste trabalho é o uso do algoritmo PIF como estratégia geral para o desenvolvimento de tarefas. Os robôs de enxame possuem características de comunicação que possibilitam que o algoritmo PIF seja executado. Este capítulo apresenta como diferentes tarefas de enxame podem ser implementadas de forma que façam uso dessa arquitetura de comunicação. Assim, os algoritmos de onda podem ser empregados como uma metodologia geral para tais tarefas.

As características de propagação e sincronização dos algoritmos de onda são exploradas neste capítulo. A onda de mensagens opera como um sincronizador para a execução de uma tarefa complexa, de forma que subtarefas ocorram em sequência com a propagação sucessiva de ondas. Por outro lado, o algoritmo PIF também pode ser a base para tarefas onde é necessária a divulgação de algum tipo de informação. O *recrutamento* e o *alinhamento* de robôs são exemplos de tarefas baseadas em ondas de mensagens. Estas duas tarefas também são empregadas como subtarefas em uma tarefa complexa de *navegação coletiva*. A onda de mensagens também é empregada no chamado *redimensionamento do enxame*, que compreende as tarefas de *agregação* e *dispersão* de robôs.

A organização deste capítulo é descrita a seguir. A Seção 4.1 descreve os conceitos de tarefa complexa e subtarefa no contexto da robótica de enxame. As Seções 4.2 e 4.3 detalham respectivamente o recrutamento e o alinhamento de robôs. A Seção 4.4 apresenta uma tarefa de movimento coordenado, que junto com o recrutamento e o alinhamento, constituem uma tarefa complexa de navegação. Na Seção 4.5 são apresentados os comportamentos de agregação e dispersão, realizados pela tarefa de redimensionamento do enxame.

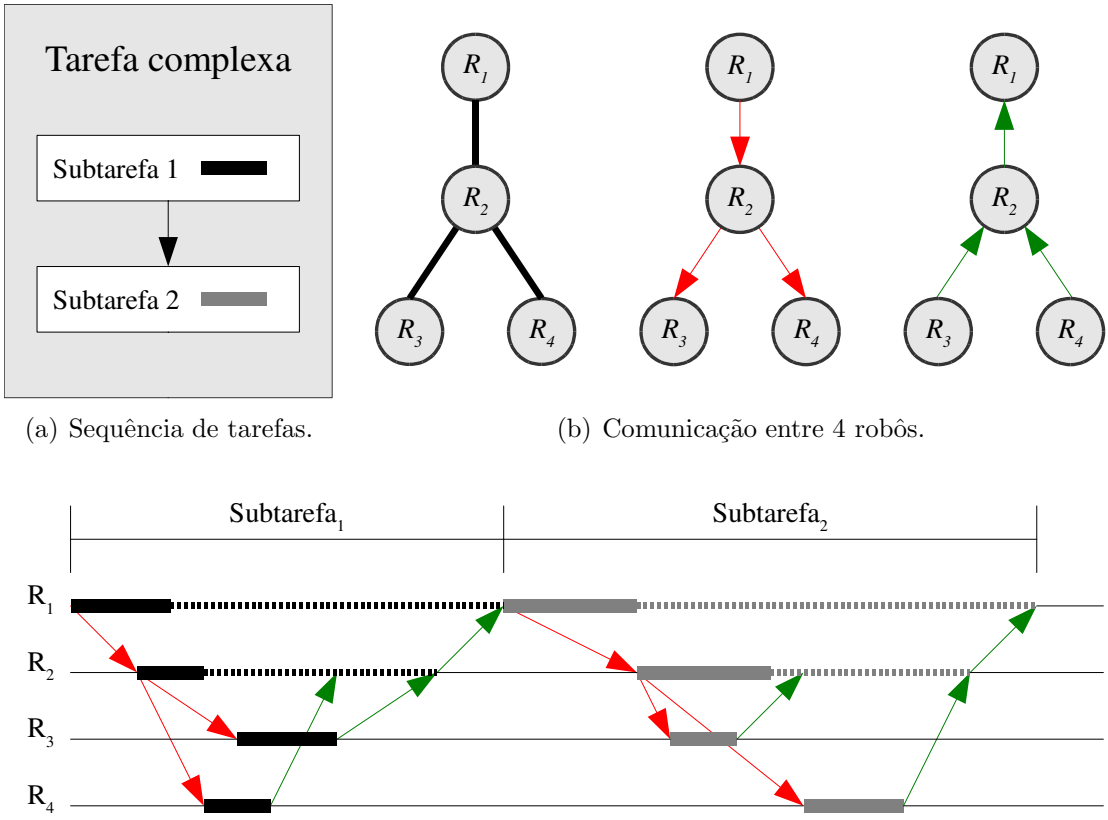
4.1 Sequência de Subtarefas

Como apresentado na Seção 2.4, existem diferentes tipos de tarefas que podem ser realizadas coletivamente por enxames de robôs. Muitas destas tarefas são constituídas por comportamentos mais simples. Por exemplo, em uma tarefa de navegação coletiva, um grupo de robôs deve ser estabelecido, para só que então a movimentação ocorra de forma coordenada. Neste caso, tem-se uma tarefa complexa que é composta por duas subtarefas. Por outro lado, uma tarefa complexa também pode ser empregada como subtarefa em outras situações. Ainda considerando a navegação coletiva como exemplo, esta pode ser empregada como parte de uma tarefa de transporte ou de forrageamento.

Definição 5 *Em um exame de robôs, uma tarefa complexa é constituída por uma sequência de comportamentos mais simples, chamados subtarefas. Uma tarefa que não é composta por subtarefas é chamada de tarefa básica.*

Subtarefas que constituem uma tarefa complexa são executadas de forma sequencial. Uma determinada subtarefa só começa a ser executada em algum dos robôs após a subtarefa anterior ter sido concluída pelos demais robôs. Tal organização evita que diferentes subtarefas sejam realizadas em diferentes robôs no mesmo tempo. Este tipo de situação é indesejável em diversas ações que envolvam a coordenação entre os robôs. A alteração no padrão de comportamento de um dos robôs devido à mudança de subtarefa pode ser interpretada de forma errônea pelos demais robôs que ainda estão executando a tarefa anterior.

O algoritmo de onda descrito no Capítulo 3 é capaz de isolar temporalmente a execução de subtarefas sequenciais. Esta operação é ilustrada na Figura 4.1. Uma determinada tarefa complexa é composta por duas subtarefas, conforme esquematizado pela Figura 4.1(a), devendo ser executada por 4 robôs. O robô R_1 é o inicializador da onda que se propaga pelo enxame, cuja relação de vizinhanças é definida pelo grafo na Figura 4.1(b). O diagrama de envio e recebimento de mensagens na Figura 4.1(c) mostra o tempo local em cada robôs (da esquerda para a direita), as mensagens de propagação e de realimentação. O tempo local de execução das subtarefas 1 e 2 é representado pelas regiões em preto e cinza, respectivamente. Com exceção de R_1, que inicia a subtarefa 1 espontaneamente, todos os robôs iniciam a execução de uma subtarefa em resposta ao recebimento de uma mensagem de propagação. O robô R_1 inicia a execução da subtarefa 2 como resposta ao recebimento da mensagem de realimentação, o que é equivalente a um evento de decisão em R_1. Desta forma, em cada um dos quatro robôs, a subtarefa 2 começa posteriormente ao término da subtarefa 1 nos demais.



(a) Sequência de tarefas.

(b) Comunicação entre 4 robôs.

(c) Diagrama de envio e recebimento para duas subtarefas em um enxame de 4 robôs.

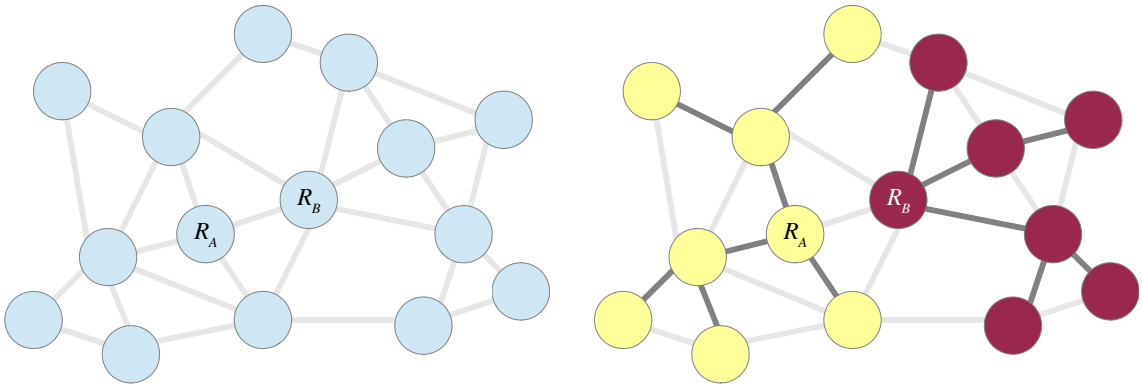
Figura 4.1: Execução de uma tarefa complexa composta por duas subtarefas.

4.2 Recrutamento de Robôs

Como apresentado na Seção 3.3, a execução do algoritmo PIF ocasiona o surgimento de uma relação pai-filhos entre os processos. Considerando um enxame de robôs, a árvore gerada pela propagação da onda de mensagens pode ser interpretada como a formação de um *grupo* de robôs, selecionados pelo inicializador para realizar uma tarefa posterior. Este processo de seleção é chamado de *recrutamento de robôs*, onde a informação propagada pelas mensagens indica a qual grupo os robôs recrutados pertencem. Em um cenário com um único robô inicializador, todos os robôs são recrutados, pois onda de mensagens é propagada por todos os robôs que constituem o enxame. Em situações com mais inicializadores, será formado um grupo de robôs por cada inicializador. A Figura 4.2 ilustra a divisão do enxame em dois grupos a partir de dois inicializadores.

4.2.1 Aplicações do Recrutamento

Em uma situação em que diferentes tarefas necessitam ser realizadas pelo enxame, é necessário definir quais robôs executarão cada uma delas. O recrutamento de



(a) Grafo de vizinhanças em um enxame.

(b) Possíveis árvores resultantes do recrutamento iniciado por R_A e R_B.

Figura 4.2: Formação de grupos em um enxame com dois robôs inicializadores.

robôs é uma tarefa que permite a divisão de trabalho, sendo usada para a formação de um ou mais grupos. Dessa forma, é de interesse usar o recrutamento como uma subtarefa inicial em uma tarefa complexa, sendo essa a etapa onde são definidos quais robôs executarão uma determinada subtarefa a ser realizada em seguida. Possíveis aplicações para o recrutamento incluem:

- Agregação: um determinado robô identifica que o enxame necessita reduzir sua área de cobertura e recruta os demais para se moverem em sua direção.
- Forrageamento: um robô encontra um objeto (ou conjunto de objetos) que precisa ser removido, e requisita ajuda para o transporte com base na vizinhança.
- Navegação: um robô precisa guiar o deslocamento de um grupo sem que haja a separação do grupo e a consequente perda de robôs.

O recrutamento se mostra necessário em situações onde um determinado robô identifica no ambiente algo que precisa ser feito por mais de um robô, ou possivelmente pelo enxame inteiro. Este robô então se comporta como inicializador da onda que recruta os demais membros do enxame, que então realizam a ação necessária como uma subtarefa seguinte ao recrutamento. Neste capítulo, a Seção 4.4 apresenta a tarefa de navegação coletiva que usa o recrutamento como uma de suas subtarefas.

4.2.2 Algoritmo Distribuído para o Recrutamento

De forma geral, o recrutamento de robôs segue o mesmo funcionamento da onda de mensagens descrito pelo Algoritmo 6 no Capítulo 3. Cada robô executa um algoritmo local que controla o recebimento e envio de mensagens.

A implementação do algoritmo para o recrutamento é representada pelo diagrama de estados da Figura 4.3. Cada robô possui duas informações iniciais: o fato de ser ou não o inicializador e a lista dos *ids* dos robôs vizinhos. Neste trabalho, a escolha do inicializador é feita de forma determinística com base em seu *id*. A descoberta da vizinhança é realizada com o envio e recebimento de mensagens, seguindo o mecanismo do Algoritmo 3 da Seção 3.2. Dois tipos de mensagens são utilizadas, conforme apresentado na Tabela 4.1. Os robôs são capazes de diferenciar estes dois tipos pelo conteúdo de msg_3, o terceiro campo da mensagem recebida.

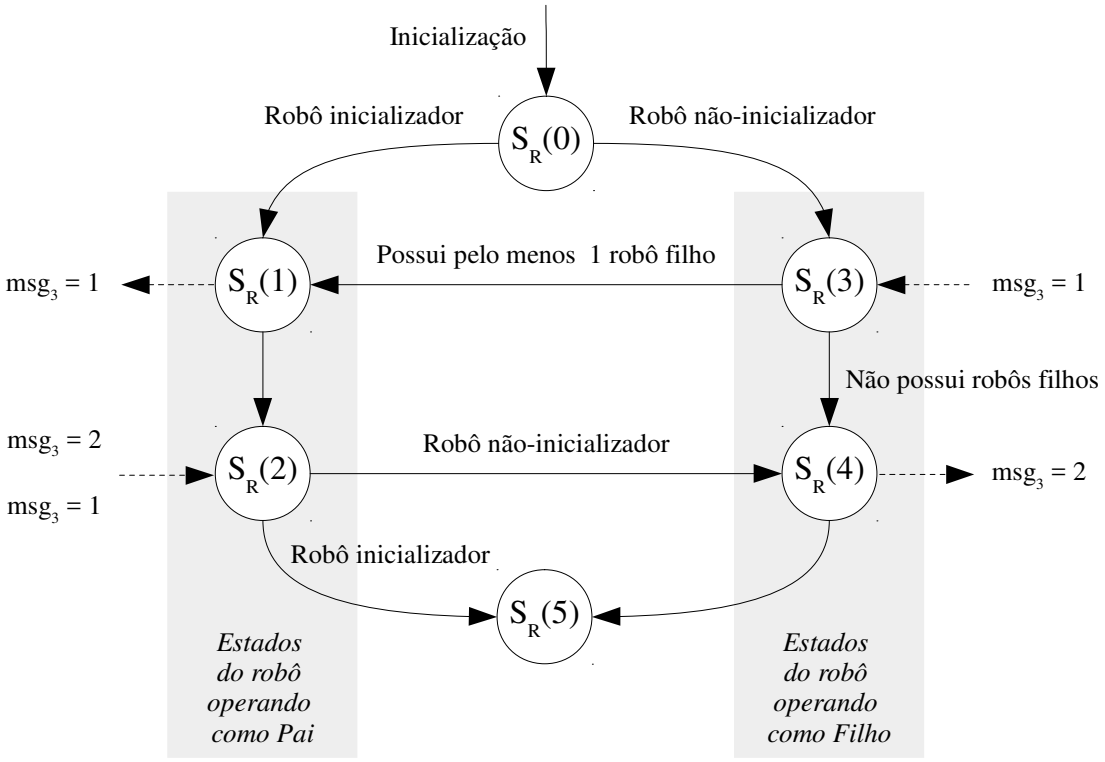


Figura 4.3: Diagrama de estados descrevendo o algoritmo de recrutamento.

Cada robô executa uma determinada sequência de estados, onde cada estado $S_R(i)$ lida com a recepção e o envio de mensagens para robôs vizinhos. Pela relação entre pais e filhos, os robôs podem assumir um dos três perfis:

- Robô Inicializador: não possui nenhum robô pai e é pai de pelo menos 1 robô. Executa a sequência de estados $S_R(0)$, $S_R(1)$, $S_R(2)$ e $S_R(5)$.
- Robôs Intermediários: não-inicializadores que possuem robô pai e pelo menos 1 robô filho. Executa todos os estados da FSM, ora operando como pai, ora como filho, na sequência $S_R(0)$, $S_R(3)$, $S_R(1)$, $S_R(2)$, $S_R(4)$ e $S_R(5)$.
- Robôs Sem Filhos: não-inicializadores que possuem 1 robô pai, mas nenhum robô filho. Executa a sequência de estados $S_R(0)$, $S_R(3)$, $S_R(4)$ e $S_R(5)$.

Os estados $S_R(1)$ e $S_R(2)$ são caracterizados pela operação do robô como pai, enviando mensagens de propagação e recebendo a realimentação dos filhos. Já os estados $S_R(3)$ e $S_R(4)$ são relacionados com a operação como robô filho. Na situação em que um robô não-inicializador possui um único vizinho, este vizinho será obrigatoriamente seu pai. Além do pai e de possíveis robôs filhos, um robô pode ser vizinho de outros membros do enxame. Estes vizinhos são filhos de outros robôs, e por isso há a necessidade de envio tanto do próprio *id* quanto do pai nas mensagens, conforme exposto pela Tabela 4.1. Um robô R que recebe uma mensagem de propagação durante o estado $S_R(3)$ após já ter recebido uma mensagem deste mesmo tipo, saberá que foi enviada por um vizinho que não é seu robô pai. Algo semelhante ocorre no estado $S_R(2)$, caso o robô receba uma mensagem de realimentação acompanhada de um *id* que não é o seu próprio. Nesta situação, o robô R recebeu uma mensagem de um robô que não é seu filho, mas cuja recepção não pode ser evitada devido a natureza *multicast* da comunicação.

Tabela 4.1: Tipos de mensagens usadas no recrutamento.

Tipo de Mensagem	msg_1	msg_2	msg_3
Propagação	id_{Pai}	id	1
Realimentação	id_{Pai}	id	2

O envio de mensagens de realimentação é iniciado pelos robôs sem filhos no estado $S_R(4)$, após receberem a mensagem de propagação de seu robô pai e eventuais mensagens de propagação dos demais vizinhos. Os demais robôs intermediários (não-inicializadores com filhos) passam ao estado $S_R(4)$ após receberem a realimentação dos filhos. Ao término do algoritmo, todos os robôs estarão no estado $S_R(5)$. Neste estado, o inicializador sabe que todos os robôs conhecem seus respectivos robôs pais e filhos. Caso exista mais de um inicializador, os robôs também saberão a qual grupo cada um pertence, e estarão prontos para a execução de uma subtarefa subsequente.

4.3 Alinhamento de Robôs

Um segundo exemplo de tarefa que pode ser implementada com uso da propagação de mensagens é o *alinhamento de robôs*. Em termos gerais, este tipo de tarefa tem por objetivo controlar o direcionamento individual de cada robô para que todos apontem para uma mesma direção. Para tal, é necessário que os robôs possuam a noção de *direção*, detectando a posição relativa de objetos em seu entorno, e a capacidade de realizar rotações em sentido horário e anti-horário.

Duas formas possíveis de realizar o alinhamento incluem o uso de referências externas e o consenso entre robôs. Havendo uma referência global para o direcionamento dos robôs (por exemplo, com o uso de bússolas), cada robô pode se

redirecionar com base nessa referência, independentemente da operação dos demais robôs. Na ausência de uma base externa, os robôs podem utilizar o direcionamento dos vizinhos como referência, resultando assim na emergência de um consenso global. Na abordagem apresentada neste capítulo, a direção de um dos robôs é usada como referência para os demais, ao mesmo tempo que cada um dos demais robôs ajusta sua direção a fim de coincidi-la com a dos vizinhos. Desta forma, todos convergem para a direção do robô de referência.

Para alcançarem o alinhamento, é necessário que cada robô possua um sistema de detecção de distância e direção (*range and bearing*). Neste tipo de sistema, cada robô se localiza na origem de seu próprio sistema de coordenadas. Quando uma mensagem é recebida, o robô é capaz de detectar a posição do vizinho em relação a seu sistema de coordenadas local. Na Figura 4.4, dois robôs R_A e R_B, estão perfeitamente alinhados. As setas indicam a frente de cada robô e são usadas como referência para cada sistema de coordenadas local. O ângulo θ_{AB} é o ângulo de R_B relativo ao sistema de coordenadas de R_A. Da mesma forma, θ_{BA} é o ângulo de R_A relativo a R_B. O alinhamento entre dois robôs podes ser verificado quando o módulo da diferença entre estes dois ângulos for igual a 180^o.

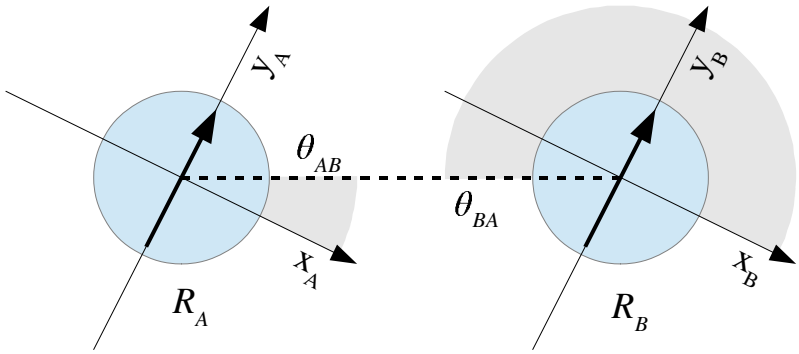


Figura 4.4: Dois robôs alinhados.

4.3.1 Aplicações do Alinhamento

Em muitas situações será necessário que os robôs estejam alinhados. Assim como o recrutamento, o alinhamento pode ser empregado como uma subtarefa em uma tarefa complexa que necessita deste padrão de posicionamento.

- Busca: em uma tarefa em que os robôs distribuídos necessitam encontrar um determinado alvo externo ao enxame, é de interesse que o primeiro robô a encontrá-lo divulgue sua direção para os demais. Desta forma, mesmo que o alvo de interesse não seja imediatamente detectável devido à distância, os robôs já estarão corretamente direcionados, reduzindo a necessidade de rotação quando o alvo estiver ao alcance de seu sistema de detecção.

- Movimento: a estrutura de rodas paralelas dos robôs móveis restringe seu deslocamento a uma combinação de movimentos em linha reta (para frente e para trás) e de rotação (sentido horário e anti-horário). Em uma situação em que robôs precisam se movimentar coletivamente, o ajuste da direção simultâneo ao movimento para frente pode perturbar a distribuição espacial de robôs. Para evitar a perda de conectividade durante a movimentação, o alinhamento prévio dos robôs auxilia a movimentação, reduzindo possíveis alterações no posicionamento inicial dos robôs.

4.3.2 Algoritmo Distribuído para o Alinhamento

O alinhamento distribuído proposto neste capítulo faz uso da onda de mensagens para que robôs ajustem seu direcionamento em relação a seu robô pai. O inicializador, cujo direcionamento será usado como referência, propaga uma onda informando a todos os robôs do enxame sobre o início do alinhamento. É considerado que uma etapa de recrutamento já aconteceu previamente. Os robôs não-inicializadores realizam dois tipos de ações: (*i*) rotacionam para alcançar o direcionamento do pai e (*ii*) participam do alinhamento de possíveis filhos.

A implementação do algoritmo local de alinhamento segue o diagrama de estados da Figura 4.5. Assim como no recrutamento, existe uma sequência de estados associados com a operação do robô como pai (estados à esquerda na Figura 4.5) e/ou como filho (direita). Cada robô possui duas variáveis lógicas empregadas como condição para a sequência de estados: *al_p* e *al_f*. A condição *al_p* será verdadeira para um determinado robô R_A se Pai_A estiver alinhado e se todos os filhos de Pai_A atenderem o direcionamento da Equação 4.1. Já a condição *al_f* será verdadeira se $Filhos_A$ estiverem alinhados com R_A, também atendendo a Equação 4.1, onde θ_{AB} é o ângulo de cada filho no sistema de coordenadas de R_A, tal que

$$|\theta_{BA} - \theta_{AB}| = 180^\circ \tag{4.1}$$

Quatro tipos de mensagens são empregadas no alinhamento. Estas mensagens estão relacionadas com a inicialização do algoritmo, o envio e o recebimento dos ângulos relativos dos vizinhos e a confirmação do término do alinhamento. Estas mensagens são sumarizadas na Tabela 4.2.

Um robô operando como pai no estado $S_A(01)$ envia uma mensagem do tipo 3 informando a seus filhos, então no estado $S_A(09)$, que o alinhamento será iniciado. Este robô então aguarda no estado $S_A(02)$ a confirmação enviada pelos filhos, que encontram-se no estado $S_A(10)$. A mensagem de confirmação, também do tipo 3, tem por objetivo fazer com que o robô saiba a posição relativa de cada filho. Da

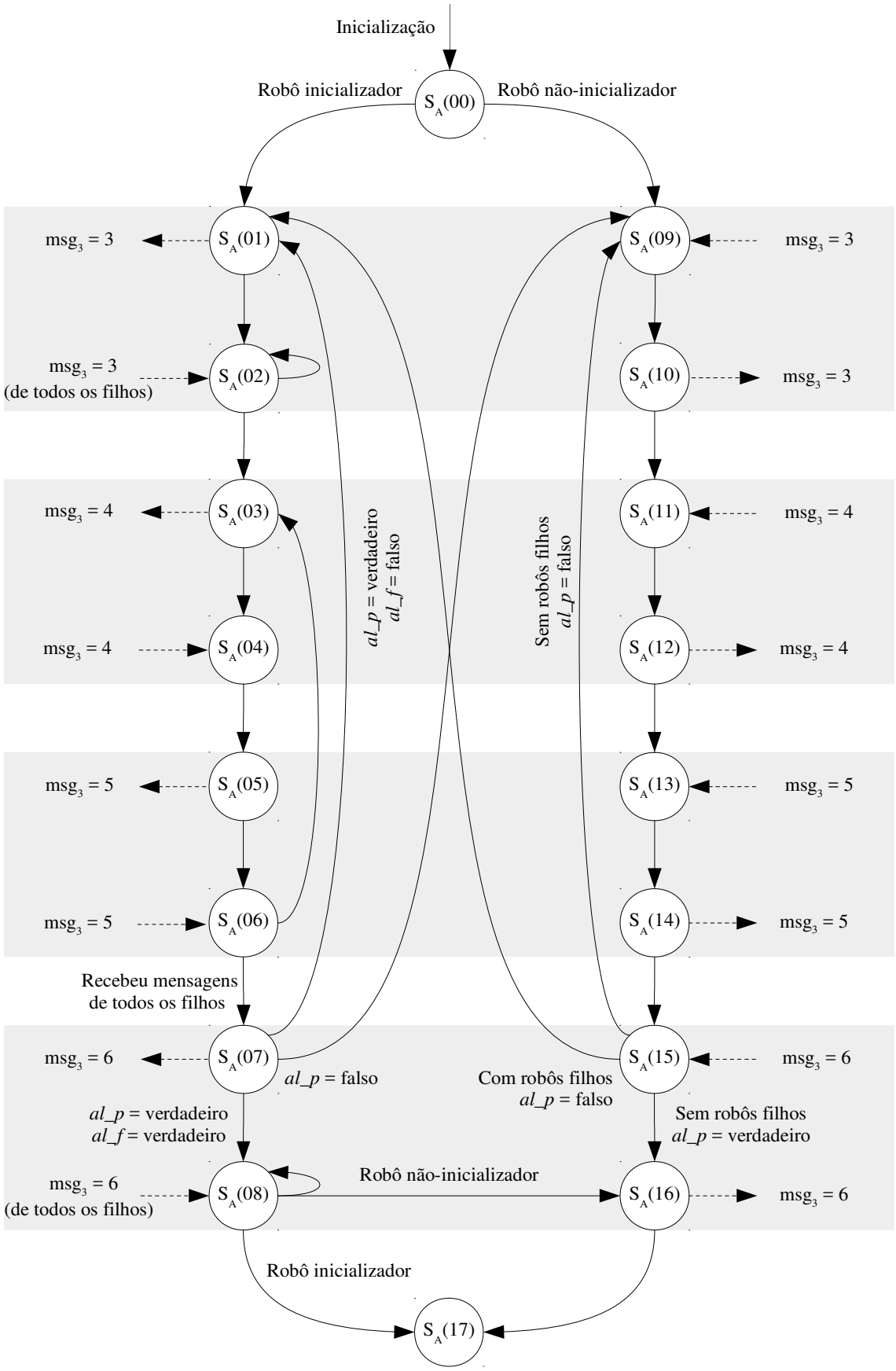


Figura 4.5: Diagrama de estados descrevendo o algoritmo de alinhamento.

mesma forma, a mensagem de propagação é empregada também para que os filhos tomem conhecimento da posição do pai.

Tabela 4.2: Tipos de mensagens usadas no alinhamento.

Tipo de Mensagem	msg_1	msg_2	msg_3
Propagação	id	–	3
Confirmação	id_{Pai}	id	3
$sen(\theta_{Filho})$ para cada filho	id_{Filho}	$sen(\theta_{Filho})$	4
$sen(\theta_{Pai})$ para o pai	id	$sen(\theta_{Pai})$	4
$cos(\theta_{Filho})$ para cada filho	id_{Filho}	$cos(\theta_{Filho})$	5
$cos(\theta_{Pai})$ para o pai	id	$cos(\theta_{Pai})$	5
Confirmação de Alinhamento	id	–	6
Realimentação	id_{Pai}	id	6

Para que um robô possa verificar o alinhamento em relação a um vizinho, é preciso conhecer a posição deste vizinho no sistema de coordenadas local. Também é necessário conhecer sua própria posição em relação ao sistema de coordenadas do vizinho. A verificação do alinhamento é realizada pelo uso da Equação 4.1. Um robô operando como pai detecta a posição de um filho com o recebimento da mensagem tipo 3 durante o estado $S_A(02)$. O ângulo detectado é composto por dois valores, $sen(\theta_{AB})$ e $cos(\theta_{AB})$, onde θ_{AB} é o ângulo da posição do filho em relação ao pai. Embora a distância entre os dois robôs também seja detectada durante o recebimento de mensagens, tal valor não é usado neste momento. Seguindo este mecanismo, o robô filho também detecta a posição do pai com o recebimento da mensagem no estado $S_A(09)$.

Os valores de $sen(\theta_{AB})$ e $cos(\theta_{AB})$ são enviados de um robô nos estados $S_A(03)$ e $S_A(05)$ por meio de mensagens do tipo 4 e 5. Os filhos recebem estas mensagens durante os estados $S_A(11)$ e $S_A(13)$. Os filhos enviam os valores de seno e cosseno durante os estados $S_A(12)$ e $S_A(14)$ para seus respectivos pais, que se encontram nos estados $S_A(04)$ e $S_A(06)$. Devido ao formato das mensagens usadas pelo modelo de robô adotado neste trabalho (descrito na Seção 3.2.5), há uma limitação no tamanho da informação que pode ser transmitida. Os valores de seno e cosseno são no intervalo $[-1, 1]$, enquanto que cada mensagem é composta por 3 *bytes*. Como mostrado na Tabela 4.2, 2 *bytes* são usados para a identificação do robô filho e o tipo da mensagem, restando somente 1 *byte* para a transmissão dos valores. Uma codificação por excesso é empregada para o envio do seno e cosseno, seguindo a Equação 4.2. Assim, os valores transmitidos são números naturais entre 0 e 255. Estes valores são recuperados com a operação inversa, conforme a Equação 4.3.

$$msg_{enviada} = \lceil sen(\theta)_{enviada} \times 128 \rceil + 128 \tag{4.2}$$

$$sen(\theta)_{recebida} = \frac{msg_{recebida} - 128}{128} \tag{4.3}$$

Conhecendo o ângulo associado com a posição do vizinho em seu sistema de coordenadas, e o ângulo de sua própria posição no sistema de coordenadas do vizinho, é possível que um robô verifique seu alinhamento em relação a este vizinho. Para tal, é feito uso das duas relações trigonométricas expressas pelas Equações 4.4 e 4.5, onde θ_{AB} é o ângulo de um robô filho no sistema de coordenadas do robô pai, e θ_{BA}é o ângulo do robô pai no sistema de coordenadas deste mesmo robô filho.

$$sen(\theta_{BA} - \theta_{AB}) = (sen(\theta_{BA}) \times cos(\theta_{AB})) - (cos(\theta_{BA}) \times sen(\theta_{AB})) \tag{4.4}$$

$$cos(\theta_{BA} - \theta_{AB}) = (cos(\theta_{BA}) \times cos(\theta_{AB})) + (sen(\theta_{BA}) \times sen(\theta_{AB})) \tag{4.5}$$

O movimento necessário para o alinhamento é a rotação do robô em torno de seu centro, o que resulta na rotação do sistema de coordenadas local. Um robô rotaciona para alcançar seu alinhamento em relação a seu pai. Esta ação segue as regras expostas no Algoritmo 7, sendo executadas por um robô filho no estado $S_A(15)$. Um robô operando como pai também verifica estas condições no estado $S_A(07)$. Nesta situação, contudo, o objetivo não é realizar rotações, mas sim verificar se os filhos atendem às condições de alinhamento. Caso os filhos estejam alinhados e a variável lógica *al_p* seja verdadeira, então a condição *al_f* também se torna verdadeira, e uma mensagem de confirmação é enviada aos filhos. Ao receberem esta mensagem do tipo 6 no estado $S_A(15)$, a variável *al_p* dos filhos passa a ser verdadeira. Caso contrário, o processo é repetido mais uma vez.

Algoritmo 7 Regra para rotação do robô filho.

Entrada: θ_{AB}, θ_{BA};
1: **se** $sen(\theta_{BA} - \theta_{AB}) < 0$ **então**
2: Rotacione no sentido anti-horário;
3: **senão se** $sen(\theta_{BA} - \theta_{AB}) > 0$ **então**
4: Rotacione no sentido horário;
5: **senão se** $cos(\theta_{BA} - \theta_{AB}) > 0$ **e** $sen(\theta_{BA} - \theta_{AB}) = 0$ **então**
6: Interrompa a Rotação;
7: **fim se**

O alinhamento possui uma estrutura, no que diz respeito à sequência de estados, similar a do recrutamento. A maior quantidade de estados se deve à necessidade de transmissão dos ângulos de pais para filhos e vice-versa. Contudo, os dois processos são similares, com robôs intermediários operando ora como pai, ora como filho. Quando um destes robôs intermediários passa a estar alinhado em relação a seu pai, sua rotação é interrompida, passando então a operar apenas como pai e verificando o alinhamento de seus filhos. Quando um robô sem filhos se torna alinhado, isto é,

sua rotação é interrompida pela regra do Algoritmo 7 e havendo o recebimento da mensagem tipo 6 enviada pelo pai, então sua participação no alinhamento se encerra. No estado $S_A(16)$, este robô sem filhos envia uma mensagem tipo 6 para seu pai, que neste caso servirá como mensagem de realimentação. Os demais robôs intermediários operam nos estados $S_A(08)$ e $S_A(16)$ com o mesmo mecanismo de propagação reversa do recrutamento, recebendo mensagens do tipo 6 dos filhos e enviando mensagens deste mesmo tipo para seus respectivos pais. Com o recebimento das mensagens do tipo 6, o robô inicializador encerra o alinhamento no estado $S_A(17)$, com todo o enxame possuindo o mesmo direcionamento.

4.4 Navegação Coletiva

A *navegação coletiva* é um exemplo de tarefa complexa, onde múltiplos robôs deslocam-se em um determinado ambiente de forma conjunta, desviando de obstáculos e sem que haja perda de robôs membros durante sua movimentação. Conforme discutido na Seção 4.1, uma tarefa complexa é composta por uma sequência de subtarefas. Uma possível implementação da tarefa de navegação coletiva faz uso do recrutamento e alinhamento como subtarefas.

- Subtarefa 1: Recrutamento de robôs;
- Subtarefa 2: Alinhamento;
- Subtarefa 3: Movimentação Coordenada.

A *movimentação coordenada* é a subtarefa responsável pelo deslocamento dos robôs. A estratégia de controle do movimento do grupo é baseada em um esquema de seguidor de líder (*leader-follower*). A ideia principal é manter a formação durante o deslocamento, isto é, disposição espacial apresentada pelos robôs ao término do alinhamento. O algoritmo de onda executado inicialmente na subtarefa de recrutamento resulta na formação de um grupo e na relação de dependência entre os robôs pais e robôs filhos. Durante o alinhamento, o direcionamento dos robôs é ajustado por meio de rotações e medições das posições dos vizinhos. Ao término do alinhamento, todos os robôs conhecem seus respectivos pais e filhos (caso existam) e a posição destes vizinhos em relação a seu próprio sistema de coordenadas local.

4.4.1 Aplicações da Navegação Coletiva

Tarefas de navegação coletiva devem ser executadas quando há necessidade do deslocamento conjunto de vários robôs, ou mesmo do enxame completo. Muitas situações podem requerer que o enxame descreva este tipo de comportamento:

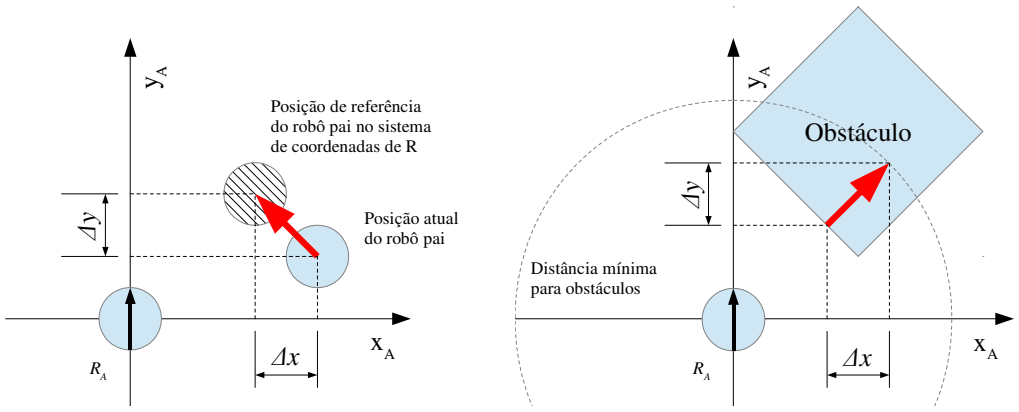
- Mudança entre ambientes: em casos onde o enxame opera em um local composto por várias regiões separadas (por exemplo, diferentes salas ou andares em uma edificação), pode ser interessante que a tarefa seja realizada em cada ambiente antes que o enxame siga para os demais. Este deslocamento entre os ambientes necessita que todos os robôs se movam sem que nenhum se perca dos demais.

- Transporte: na situação em que vários robôs devem transportar um único objeto, é necessário haver uma coordenação entre os membros do enxame para que o objetivo seja alcançado.

4.4.2 Algoritmo para Movimentação Coordenada

Ao término do alinhamento, os robôs iniciam a movimentação. Uma vez que todos apresentam o mesmo direcionamento, bastaria que o movimento se desse de forma linear, com todos os robôs apresentando a mesma velocidade. Contudo, esta abordagem levaria ao colapso da formação desejada, uma vez que inúmeros fatores afetam a movimentação individual dos robôs. Estes fatores incluem o ruído no sistema de movimentação, irregularidades na superfície ao longo do trajeto e a possibilidade de colisão com obstáculos. Para evitar este tipo de problemas, é necessário um mecanismo para que a movimentação transcorra mantendo a formação do enxame, mas, ao mesmo tempo, sendo flexível durante o desvio de obstáculos.

Durante a movimentação coordenada, o robô inicializador é considerado o líder do grupo e descreve um movimento independente dos demais. Este movimento pode ser livre ou direcionado a algum alvo. Os demais robôs iniciam sua movimentação ao detectarem uma alteração na posição de seus respectivos robôs pais. Um único estado pode ser empregado para esta subtarefa: cada robô envia um determinado



(a) Rastreio do pai

(b) Desvio de obstáculos

Figura 4.6: Duas situações em que o movimento de R_A é necessário para minimizar as distâncias Δx e Δy.

Algoritmo 8 Regra para ajuste de posição dos robôs (seguidor de líder).

Entrada: x_{Pai}, y_{Pai};
Saída: Velocidade dos motores;
1: Realize a medição de x_{Atual} e y_{Atual};
2: $\Delta x = x_{Pai} - x_{Atual}$;
3: $\Delta y = y_{Pai} - y_{Atual}$;
4: **se** $\Delta x > 0$ **então**
5: Rotacione no sentido horário;
6: **senão**
7: Rotacione no sentido anti-horário;
8: **fim se**
9: **se** $\Delta y > 0$ **então**
10: Aumente a velocidade;
11: **senão**
12: Reduza a velocidade;
13: **fim se**

tipo de mensagem para os filhos, ao mesmo tempo que aguarda este mesmo tipo de mensagens vinda de seus pais. Esta mensagem deve se diferenciar daquelas empregadas nas subtarefas anteriores, logo será identificada como sendo do tipo 7. A mensagem também contém o *id* do robô remetente, para que um robô filho saiba se a mensagem recebida foi de fato enviada por seu robô pai. Ao mesmo tempo, é considerado que os robôs possuem sensores de distância e são capazes de detectar e desviar de obstáculos presentes em sua vizinhança.

Duas regras controlam o movimento dos robôs: o rastreio do pai (*father-following*) e o desvio de obstáculos (*obstacle avoidance*). Estas duas situações são ilustradas na Figura 4.6. O robô inicializador não segue nenhum outro, então permanecerá em sua trajetória até detectar algum obstáculo. Com o recebimento de uma mensagem do tipo 7, os robôs não-inicializadores sabem a posição atual de seu robô pai. A diferença entre a posição atual e a posição de referência (previamente registrada ao término do alinhamento) indica o quanto o robô deve se mover a fim de ajustar sua posição. A Figura 4.6(a) mostra como a movimentação dos robôs resulta em uma alteração na posição do pai no sistema de coordenadas local do robô R_A. Este robô precisará se movimentar de tal forma a minimizar Δx e Δy, fazendo assim com que a posição atual de seu robô pai se aproxime da posição de referência. Contudo, devido à natureza não-holonômica do mecanismo de movimentação composto por dois motores paralelos (*car-like*), o robô pode descrever diretamente um deslocamento em y, mas não em x. Em outras palavras, o robô possui liberdade para mover-se para frente e para trás, mas não para a esquerda e para a direita. O deslocamento frontal é obtido configurando ambos os motores com a mesma velocidade. O deslocamento lateral pode ser alcançado com a combinação do movimento frontal com rotações, definindo a velocidade de cada motor com valores diferentes,

resultando em trajetórias curvas para a esquerda e para a direita. A regra de ajuste de posições é descrita no Algoritmo 8.

Uma segunda situação em que o robô necessita se deslocar é na presença de obstáculos. Cada robô possui ao seu redor um *raio de segurança* que indica a distância mínima aceitável para a presença de obstáculos. Um obstáculo pode ser um objeto no ambiente, as paredes que delimitam a área onde os robôs atuam, ou mesmo outros robôs que eventualmente se aproximam mais do que a distância limite. O ajuste da posição para o desvio de obstáculos faz uso das mesmas regras do Algoritmo 8. Como ilustrado pela Figura 4.6(b), Δx e Δy são, neste caso, a diferença entre a distância atual para o obstáculo e a distância mínima aceitável no raio de segurança.

4.5 Redimensionamento do Enxame

Um outro comportamento coletivo que pode ser implementado com o uso da estratégia de ondas de mensagens é o redimensionamento do enxame. Esta tarefa difere das apresentadas previamente por necessitar de um modelo de robô com características de detecção não usadas na navegação coletiva. Tal modelo deve ser capaz de observar a posição de todos os robôs em sua vizinhança. Conhecendo a informação espacial de sua vizinhança, cada robô ajusta sua posição, reduzindo ou aumentando a distância uns dos outros. Dependendo dos parâmetros usados, o redimensionamento pode resultar em dois comportamentos: agregação ou dispersão.

4.5.1 Detecção da Vizinhança

O modelo o robô *Kilobot* realiza a verificação de distâncias com base na intensidade do sinal recebido durante a comunicação. Assim, o modelo simulado possui uma medição de distâncias associada com a comunicação entre robôs, emulando assim o mecanismo de medição presente no modelo físico do *Kilobot*. Esta medição observa apenas a distância entre os robôs, mas não seu direcionamento. A subtarefa de movimentação coordenada, discutida na Seção 4.4, requer a informação da vizinhança para ser realizada. Cada robô utiliza a posição relativa de seu pai para definir a velocidade e direção de seu movimento. Desta forma, o modelo original do *Kilobot* disponível no simulador *V-REP* foi modificado de forma que cada robô possa observar o posicionamento de seu respectivo robô pai durante o recebimento de mensagens. Contudo, esta modelagem ignora a localização dos demais robôs presentes na vizinhança, pois tal informação não é necessária para o esquema de seguidor de líder. Por outro lado, este modelo modificado também faz uso de um sensor de distância, que identifica a posição do obstáculo mais próximo.

No estudo do redimensionamento, observou-se a necessidade de detecção da posição de todos os robôs e obstáculos dentro de uma determinada região ao redor do robô, com o objetivo de evitar colisões e/ou perda de conectividade entre robôs. Isto difere da movimentação coordenada apresentada previamente, onde a totalidade da informação da vizinhança não mostrou-se imprescindível.

Sistema Modificado de Detecção de Robôs

Nas tarefas de agregação e dispersão de robôs, a detecção da vizinhança é uma expansão da detecção empregada na subtarefa de movimentação coordenada. O objetivo da modificação é permitir que cada robô identifique, com uma única leitura do sistema de detecção, a presença de todos os robôs e obstáculos presentes em uma certa área de detecção em seu entorno. Esta área é considerada idêntica à área de alcance de comunicação, embora neste caso os sistemas de detecção e comunicação sejam completamente independentes. De posse da informação espacial da vizinhança, os robôs são capazes de evitar colisões entre si durante a agregação. Também é possível evitar a perda de conectividade durante a dispersão do enxame. Assim, será considerado que o enxame simulado de *Kilobots* possui dois sistemas independentes de interação entre robôs:

- Um *sistema de comunicação*, necessário para a propagação da onda de mensagens que definem o início e término das tarefas. Este sistema de comunicação é o mesmo usado nas tarefas discutidas previamente.

- Um *sistema de detecção* capaz de identificar a posição dos demais robôs presentes em uma vizinhança local no entorno de um determinado robô.

Algoritmo para Detecção da Vizinhança

O modelo real do robô *Kilobot* não possui um sistema dedicado de localização de múltiplos vizinhos e obstáculos. Contudo, certos modelos de robôs apresentados no Capítulo 2 possuem *hardware* capaz de tal detecção. Os robôs *E-puck* e *Khepera*, por exemplo, possuem em seu entorno diversos sensores IR com os quais são capazes de identificar outros membros do enxame. Assim, a modificação do modelo simulado do *Kilobot* introduz uma funcionalidade presente em outros modelos de robôs móveis.

No simulador *V-REP*, cada robô pode ter acesso a todas as informações existente no ambiente. Esta capacidade precisa ser restrita, caso contrário a simulação torna-se infiel à realidade. O Algoritmo 9 descreve o funcionamento de um sistema de detecção de posições. Cada robô restringe a observação do ambiente a apenas um raio de detecção predefinido em seu entorno. Os objetos simulados são então identificados como robôs vizinhos ou obstáculos. A informação de posição dos robôs

é empregada no cálculo do deslocamento de cada robô seguindo as chamadas regras de *flocking*, tratadas na Seção 4.5.2.

Algoritmo 9 Detecção de elementos de simulação na vizinhança.

Entrada: Informações de posição de todos os elementos simulados;
Saída: Posição de robôs dentro do raio de detecção;
Saída: Posição de obstáculos dentro do raio de detecção;

```
1: j := 0; k := 0;
2: para i := 1 até número de elementos simulados faça
3:    se elemento[i].distância < raio de detecção então
4:       se elemento[i].tipo = robô então
5:          j := j + 1;
6:          robô[j].informações = informações de posição do elemento i;
7:       senão se elemento[i].tipo = obstáculo então
8:          k := k + 1;
9:          obstáculo[k].informações = informações de posição do elemento i;
10:      fim se
11:   fim se
12: fim para
```

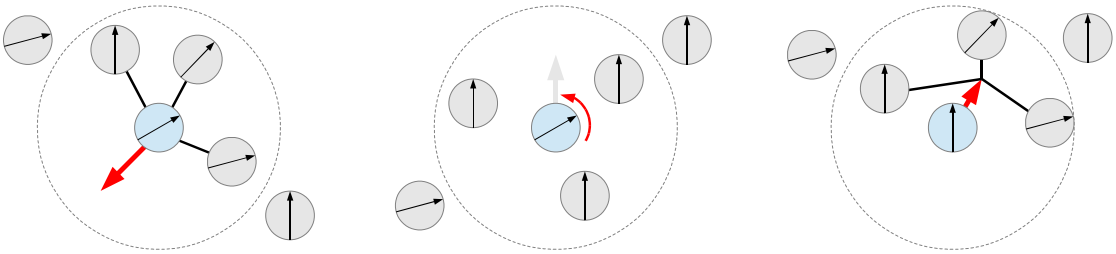
4.5.2 *Flocking* de Agentes

Conforme apresentado previamente na Seção 2.4.2, a expressão *flocking* (voo em bando, revoada) é empregada no contexto de enxames artificiais para descrever a modelagem computacional do movimento coletivo de agentes. Este conceito foi inicialmente desenvolvido por Reynolds [105] para a criação de animações de deslocamentos realísticos de bandos de animais.

Cada indivíduo no *flocking* artificial opera com base na observação de sua vizinhança. Um membro do grupo identifica os demais indivíduos e usa a posição e velocidade destes vizinhos para calcular sua própria velocidade. Este processo de (*i*) leitura, (*ii*) cálculo e (*iii*) atualização das velocidades repete-se indefinidamente durante o *flocking*.

Regras de *Flocking*

A velocidade de cada indivíduo (ou agente) é definida pela contribuição de diferentes regras. Cada regra diz respeito a alguma característica do grupo que se deseja reforçar ou atenuar. As três regras básicas são a *separação*, o *alinhamento* e a *coesão*. O funcionamento destas três regras é representado na Figura 4.7. Em todos os casos, a linha pontilhada indica a área de detecção no entorno do indivíduo em destaque. Desta forma, este indivíduo interage com apenas três vizinhos. A posição dos demais agentes não contribui para o deslocamento do indivíduo central.



(a) Separação. (b) Alinhamento. (c) Coesão.

Figura 4.7: Três regras básicas de *flocking*.

A separação entre indivíduos é representada pela Figura 4.7(a). Nesta regra o indivíduo tende a se afastar dos demais. Assim, evita-se o surgimento de regiões com muitos agentes, ou mesmo colisões. No alinhamento apresentado na Figura 4.7(b), o indivíduo ajusta sua posição por meio de rotações a fim de alcançar o direcionamento médio da vizinhança. Por fim, a regra de coesão ilustrada na Figura 4.7(c) tem por intuito impedir que algum indivíduo se perca dos demais. Para tal, o movimento calculado ocorre na direção da posição média dos vizinhos.

Ajuste de Posição

De forma geral, o processo de *flocking* é obtido pela repetição das etapas de observação da vizinhança e ajuste de posição, como ilustrado no Algoritmo 10. As regras de *flocking* resultam em três componentes de velocidade que definem a movimentação do indivíduo, como apresentado no Algoritmo 11.

Cada agente move-se com base na observação de sua própria vizinhança. Contudo, tal movimento resulta na modificação das posições dos agentes. Logo, a cada medição, um indivíduo observa seus vizinhos em posições gradualmente diferentes. A contribuição de cada regra de *flocking* é ponderada pelos parâmetros w_1, w_2 e w_3 no Algoritmo 11. O parâmetro w_0 indica o peso da velocidade atual do indivíduo em relação às três velocidades calculadas. Este fator indica a inércia do indivíduo, e sua ausência acarretaria em variações bruscas de velocidade devido à movimentação da vizinhança.

Algoritmo 10 *Flocking* de indivíduos.

```
1: Inicialize aleatoriamente a velocidade do indivíduo;
2: enquanto Verdadeiro faça
3:     Detecte a vizinhança;
4:     Movimente-se com base nas regras de flocking;
5: fim enquanto
```

O ajuste dos parâmetros de *flocking* faz com que o grupo de indivíduos apresente diferentes comportamentos. Por exemplo, a variação de w_0 pode aumentar

Algoritmo 11 Movimento de um indivíduo a partir das regras de *flocking*.

Entrada: Informações de posição da vizinhança;

1: v_1 = velocidade de separação;
2: v_2 = velocidade de alinhamento;
3: v_3 = velocidade de coesão;
4: velocidade = $w_0 \times v_{atual} + (w_1 \times v_1) + (w_2 \times v_2) + (w_3 \times v_3)$;
5: posição = posição + velocidade$\times \Delta t$;

ou diminuir a velocidade resultante do grupo como um todo. Por outro lado, uma mudança na proporção entre separação e coesão resulta em reajustar a disposição espacial entre os agentes até alcançarem uma condição de equilíbrio. A contribuição da componente de coesão faz com que os agentes se aproximem uns dos outros, enquanto que a separação resulta em uma repulsão mútua. Esta característica de equilíbrio entre coesão e separação é a base para as tarefas de redimensionamento de enxames de robôs.

4.5.3 Agregação e Dispersão de Robôs

O *redimensionamento do enxame* é um comportamento de organização espacial, onde os robôs alteram a distância em relação aos vizinhos de forma a alcançar alguma característica específica. Duas tarefas resultantes são a *agregação* e a *dispersão*. Na tarefa de agregação, todos os robôs aproximam-se uns dos outros, reduzindo a área do ambiente ocupada pelo enxame. Na dispersão, ocorre o inverso: os robôs repelem-se mutuamente de forma a aumentar a área de atuação do enxame.

Conforme discutido na Seção 4.5.2, as regras de *flocking* fazem com que um enxame apresente diferentes comportamentos espaciais. As tarefas de agregação e dispersão, apresentadas neste capítulo, fazem uso destas regras no controle da movimentação dos robôs que constituem o enxame. Além disso, o redimensionamento do enxame com base em regras de *flocking* obedece à estratégia de ondas de mensagens. Desta forma, o redimensionamento apresenta as seguintes características:

- Uma subtarefa de recrutamento é executada antes do início da movimentação. Isto cria a relação entre robôs pais e robôs filhos, necessária para a comunicação por mensagens entre os membros do enxame.
- Uma onda de mensagens determina o início e o término do redimensionamento.
- A movimentação não afasta robôs pais e seus respectivos filhos para além de suas áreas de comunicação.
- Durante a agregação, um robô com dois ou mais vizinhos não deve se afastar em exagero de um para garantir a aproximação de outro. Isto pode resultar no surgimento de regiões sem robôs no enxame.

- De forma análoga, durante a dispersão um robô deve evitar a aproximação de outros enquanto se afasta dos demais. O afastamento deve ocorrer de forma mútua em todo o enxame.

Na execução do redimensionamento, a movimentação dos robôs é definida por regras de *flocking* com o uso do sistema de detecção. Cada robô observa seu entorno e define seu deslocamento com base na sua vizinhança. Ao mesmo tempo, o sistema de comunicação por mensagens também é usado, garantindo que a agregação ou dispersão seja interrompida quando alguma condição de parada for alcançada.

4.5.4 Aplicações do Redimensionamento

A agregação de robôs é necessária quando um enxame, inicialmente disperso, precisa ocupar uma região menor do espaço do que a ocupada atualmente. Assim como o recrutamento, a agregação pode ser empregada como uma subtarefa em aplicações diversas, tais como:

- Recolhimento do enxame: os robôs espalhados no ambiente durante a realização de uma determinada tarefa precisam ser agrupados em uma região para serem recuperados pelo usuário ao término da tarefa.

- Transporte de objetos: os robôs identificam e capturam objetos no ambiente, transportando-os então para uma uma região específica.

- Perseguição e captura: a agregação pode ser empregada quando se deseja interceptar algum alvo, concentrando os robôs em seu entorno e impedido sua movimentação e fuga.

- Mudança de ambiente: um grupo de robôs em movimento pode precisar atravessar uma região estreita, ou mesmo se deslocar de um ambiente para outro de menor área.

Na dispersão ocorre o comportamento inverso ao da agregação, com os robôs inicialmente concentrados, e precisando se espalhar pela área onde o enxame atuará posteriormente. Uma subtarefa baseada na dispersão de robôs pode ser usada em diferentes situações:

- Dispersão durante a inicialização: o enxame é inicialmente colocado no ambiente em uma região próxima ao usuário, e então dispersa-se por uma área mais extensa. Este comportamento pode ser necessário em tarefas de busca, onde os robôs devem se espalhar por um ambiente com o objetivo de identificar uma região de característica distinta, ou mesmo a localização de objetos.

- Cobertura: em muitos casos é de interesse que o enxame se espalhe, ocupando a maior área possível do ambiente sem que haja a fragmentação da topologia de comunicação, levando a uma perda de conectividade entre os membros do enxame.
- Uma vez dispersos, alguns dos robôs podem permanecer estáticos para que suas posições se tornam referências em um sistema de localização.

4.5.5 Algoritmo para o Redimensionamento

O algoritmo distribuído que rege o redimensionamento do enxame possui duas componentes que operam de forma coordenada. Uma das operações é a movimentação dos robôs realizada com base nas regras de *flocking*. O deslocamento de cada robô é definido pelo posicionamento dos robôs vizinhos. O outro processo é um algoritmo de onda de mensagens que indica aos membros do enxame o início e o término do redimensionamento. De modo geral, a estratégia de ondas de mensagens empregada no redimensionamento do enxame é idêntica a usada durante alinhamento. Tem-se uma onda de mensagens inicial responsável pelo recrutamento dos robôs e que define a relação entre pais e filhos. A seguir, inicia-se uma nova onda a partir do robô inicializador. O recebimento das mensagens leva os demais membros do enxame a iniciar a movimentação que resulta na agregação ou na dispersão.

Sequência de Estados

O algoritmo que regula o envio e recebimento de mensagens durante o redimensionamento pode ser representado pelo diagrama de estados da Figura 4.8. Tal como o alinhamento, os robôs possuem duas variáveis lógicas que direcionam a sequência de estados. A condição *red_p* indica que o robô em questão alcançou algum critério de redimensionamento predefinido. Já a condição *red_f* indica que seus robôs filhos alcançaram tal critério.

Três tipos diferentes de mensagens são empregadas durante o redimensionamento, conforme apresentado na Tabela 4.3. A mensagem do tipo 7 é empregada por um robô operando como pai na propagação da onda que inicia a operação de seus robôs filhos. Estes robôs filhos confirmam o recebimento da mensagem com o envio de uma mensagem do mesmo tipo para o pai. Este processo de envio e recebimento de mensagens ocorre nos estados $S_{AD}(01)$ e $S_{AD}(07)$ para o robô operando como pai, e $S_{AD}(02)$ e $S_{AD}(08)$ quando operando como filho. O envio de mensagens se repete nos estados $S_{AD}(03)$, $S_{AD}(04)$, $S_{AD}(09)$ e $S_{AD}(10)$, mas com o uso de mensagens do tipo 8. Esta operação é necessária para garantir que ambos os robôs filhos e pais confirmem o recebimento das mensagens. As mensagens do tipo 8 atuam como aceitação em um esquema de *handshake* duplo.

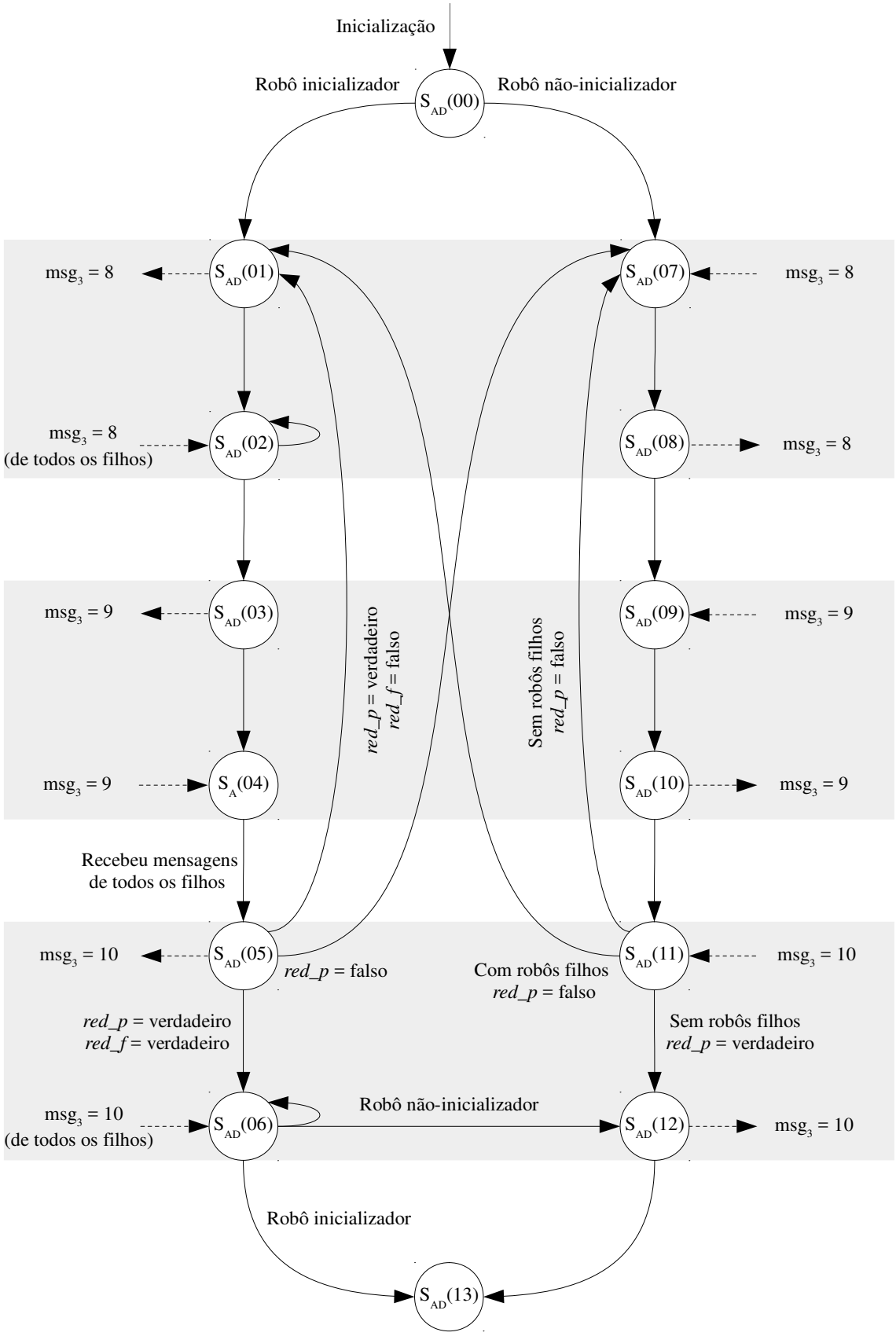
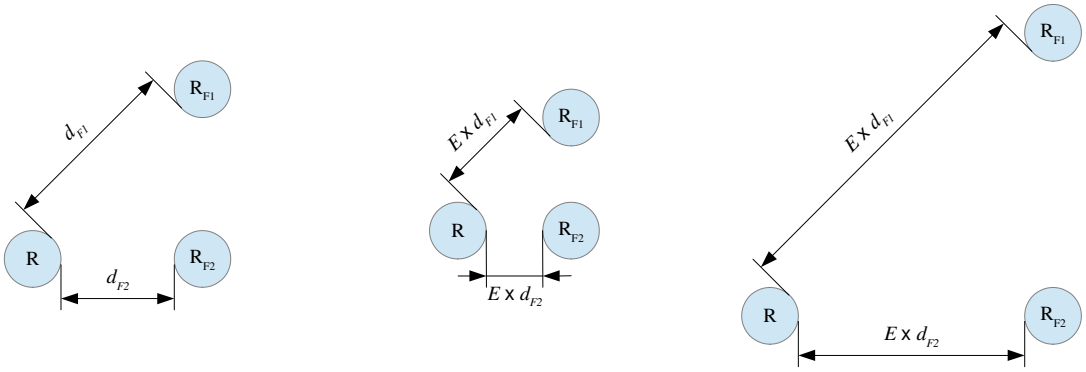


Figura 4.8: Diagrama de estados descrevendo o algoritmo de redimensionamento.

Tabela 4.3: Tipos de mensagens usadas durante o redimensionamento.

Tipo de Mensagem	msg_1	msg_2	msg_3
Propagação	id	–	7
Confirmação	id_{Pai}	id	7
Propagação	id	–	8
Confirmação	id_{Pai}	id	8
Confirmação do redimensionamento	id	–	9
Realimentação	id_{Pai}	id	9

Durante os estados iniciais, todos os robôs executam a operação de movimentação. Isto se repete até que uma determinada condição de redimensionamento seja alcançada. Esta condição é definida pelo parâmetro E, chamado de *fator de escala*. A Figura 4.9 ilustra o efeito de E no redimensionamento do enxame. Cada robô conhece a distância d_F em relação a seus robôs filhos. Esta distância de referência é detectada durante a subtarefa inicial de recrutamento.



(a) Distâncias de referência entre um robô e seus respectivos filhos.

(b) Comportamento de agregação do enxame com $E < 1$.

(c) Comportamento de dispersão do enxame com $E > 1$.

Figura 4.9: Impacto do fator de escala E no redimensionamento do enxame.

Os robôs verificam a distância em relação aos filhos no estado $S_{AD}(04)$. Dependendo dos valores obtidos, os robôs podem seguir para diferentes estados a partir de $S_{AD}(05)$. Caso a condição de redimensionamento seja atendida por cada filho e a condição *red_p* seja verdadeira, o robô então encerra sua movimentação e envia uma mensagem do tipo 9 para seus filhos. Esta mensagem indica aos filhos que o redimensionamento foi alcançado. A condição *red_f* do robô em questão passa a ser verdadeira, enquanto que a condição *red_p* de cada filho também passa a ser verdadeira. Por outro lado, se o redimensionamento não tiver sido alcançado, duas situações são possíveis. Um robô intermediário que possui as condições *red_p* e *red_f* falsas volta a operar como filho no estado $S_{AD}(07)$, para então tentar se posicionar a uma distância adequada de seu respectivo pai. Caso a condição *red_p* seja verdadeira, mas *red_f* seja falsa, o robô encerra sua movimentação e retorna ao estado

$S_{AD}(01)$. Quando as condições de redimensionamento são alcançadas, o inicializador e os robôs intermediários transitam do estado $S_{AD}(05)$ para $S_{AD}(06)$, onde aguardam a mensagem do tipo 10. Esta é uma mensagem de realimentação enviada pelos robôs filhos, que se encontram no estado $S_{AD}(12)$. O algoritmo termina quando todos os robôs já se encontram no estado $S_{AD}(13)$. Neste estado, o inicializador tem certeza de que o enxame completou o redimensionamento.

Movimentação dos Robôs

A confirmação se o enxame alcançou o redimensionamento esperado depende da distância dos robôs filhos em relação a seu pai, conforme ilustrado pela Figura 4.9. Entretanto, se os robôs empregassem apenas esta medição de distância, haveria a possibilidade de aproximação demasiada em relação a outros robôs vizinhos. Assim, buscou-se desenvolver uma regra de movimentação que considerasse o posicionamento de todos os robôs vizinhos. Em outras palavras, um robô se move em reação à distância dos demais robôs localizados dentro da área de seu sistema de detecção.

A estratégia de movimentação é inspirada no *flocking* de agentes introduzida na Seção 4.5.2. Nesta abordagem, são empregadas as regras de separação e coesão. A regra de alinhamento não foi necessária, mostrando-se mais útil nas situações em que o enxame como um todo precisa se deslocar. Por outro lado, outras duas regras foram acrescentadas, baseadas no rastreio do robô pai e no desvio de obstáculos.

Para o comportamento de coesão, cada robô observa sua vizinhança e calcula a posição média dos robôs dentro da área de detecção, seguindo a Equação 4.6. A *distância de coesão* $d_{x,y}(coe)$ possui componentes (x, y) no sistema de coordenadas do robô que está detectando a vizinhança. A soma das distâncias $d_{x,y}(vizinho)$ é normalizada pelo alcance máximo do sistema de detecção ($raio\_sensor$) e pela quantidade de vizinhos ($num\_vizinhos$). Assim, cada coordenada da distância calculada possui valor absoluto no intervalo $[0, 1]$, onde o valor 0 representa a distância mais próxima entre os robôs, e o valor 1 representa a distância mais afastada detectável.

$$d_{x,y}(coe) = \frac{\sum_{i=1}^{num\_vizinhos} d_{x,y}(vizinho_i)}{num\_vizinhos \times raio\_sensor} \tag{4.6}$$

A distância $d_{x,y}(coe)$ é empregada no cálculo da velocidade dos dois motores do robô. As expressões na Equação 4.7 mostram quatro componentes de velocidade: duas associadas com a distância em x, e duas associadas com a distância em y. É importante ressaltar que o valor d_x da distância de coesão acarreta em rotações do robô, enquanto que o valor d_y afeta na movimentação para frente ou para trás. Este comportamento é similar ao descrito pelo Algoritmo 8 para a navegação coletiva.

$$
\begin{aligned}
velocidade_{x1} &= v_{max} \times d_x & velocidade_{y1} &= v_{max} \times d_y \\
velocidade_{x2} &= -v_{max} \times d_x & velocidade_{y2} &= v_{max} \times d_y
\end{aligned}
\tag{4.7}
$$

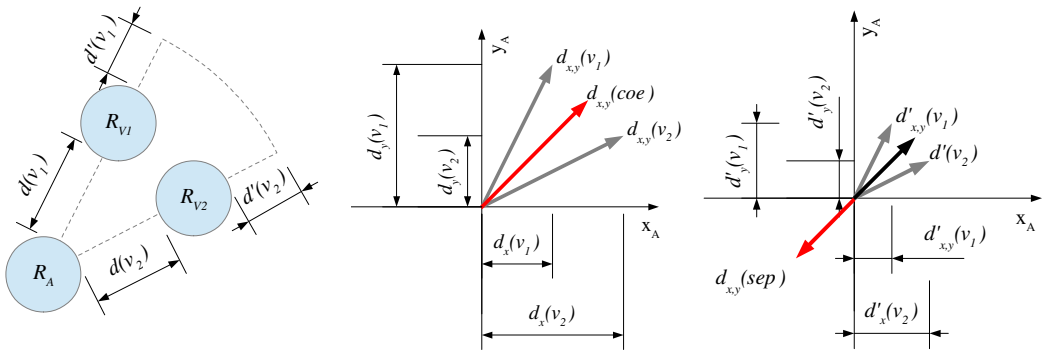
A velocidade de cada motor é composta pelas velocidades associadas com x e y, como expresso nas Equações em 4.8. A velocidade v_{cw} movimenta o motor responsável pela rotação no sentido horário (*clockwise*), enquanto que o movimento no sentido anti-horário (*counterclockwise*) é controlado por v_{ccw}, tal que

$$
\begin{aligned}
v_{cw} &= 0,5 \times velocidade_{x1} + 0,5 \times velocidade_{y1} \\
v_{ccw} &= 0,5 \times velocidade_{x2} + 0,5 \times velocidade_{y2}
\end{aligned}
\tag{4.8}
$$

Na regra de separação, o valor usado para o cálculo de velocidades não é a distância entre robôs, mas sim a distância $d'_{x,y}(vizinho)$ entre um vizinho e o raio máximo de detecção nesta direção. Este valor é somado para todos os vizinhos, resultando na distância $d_{x,y}(sep)$ expressa na Equação 4.9, tal que

$$
d_{x,y}(sep) = -\frac{\sum_{i=1}^{num\_vizinhos} d'_{x,y}(vizinho_i)}{num\_vizinhos \times raio\_sensor}
\tag{4.9}
$$

Robôs mais afastados exercem menor repulsão uns nos outros. O oposto ocorre com robôs mais afastados da borda, e consequentemente, mais próximos do robô que realiza a detecção. Neste caso, a velocidade de separação deve ser máxima. O sinal negativo indica que a movimentação final deve ocorrer na direção oposta do somatório, por se tratar de um comportamento de separação. Assim, as mesmas equações de velocidade empregadas para a coesão também podem ser usadas para a separação. Tanto $d_{x,y}(coe)$ quanto $d_{x,y}(sep)$ são representados na Figura 4.10.



(a) Representação dos robôs R_{V1} e R_{V2}, vizinhos de R_A.

(b) Distâncias dos robôs R_{V1} e R_{V2} em relação ao robô R_A.

(c) Distâncias dos robôs R_{V1} e R_{V2} em relação ao raio de detecção de R_A.

Figura 4.10: Distâncias usadas no cálculo de $d_{x,y}(coe)$ e $d_{x,y}(sep)$ para um robô R_A com dois vizinhos, R_{V1} e R_{V2}.

Outras duas regras são empregadas para a definição da velocidade dos robôs no redimensionamento. A primeira está associada com o rastreio do robô pai. Esta regra tenta impedir que os filhos se afastem em demasia de seus respectivos pais, o que poderia causar a perda de conectividade no enxame. A segunda é uma regra de desvio de obstáculos. O objetivo desta regra é impedir que robôs colidam caso a distância mútua se torne menor que um determinado limiar. Ambas são similares às regras de movimentação usadas na tarefa de navegação da Seção 4.4. Neste caso, cada uma destas regras resultam da detecção de um único vizinho, sendo ou o robô pai, ou um robô com risco de colisão. Estas duas detecções de distância, $d_{x,y}(Pai)$ e $d_{x,y}(colisão)$, resultam em componentes de velocidade por meio da Equação 4.7.

A velocidade de cada robô é composta pelas velocidades obtidas pelas 4 regras. Na Equação 4.10, w_1, w_2, w_3 e w_4 designam os pesos associados com as regras de 1 à 4. O valor máximo possível de v_{cw} e v_{ccw} é v_{max}, a velocidade máxima suportada pelo robô.

$$\begin{aligned} velocidade_{cw} &= w_1 \times v_{cw}(1) + w_2 \times v_{cw}(2) + w_3 \times v_{cw}(3) + w_4 \times v_{cw}(4) \\ velocidade_{ccw} &= w_1 \times v_{ccw}(1) + w_2 \times v_{ccw}(2) + w_3 \times v_{ccw}(3) + w_4 \times v_{ccw}(4) \end{aligned} \tag{4.10}$$

Pela Equação 4.7 vê-se que v_{max} é ponderada pela distância calculada em cada regra. Estas distâncias encontram-se no intervalo $[0, 1]$, por se tratarem de grandezas normalizadas. Logo, as velocidades v_{cw} e v_{ccw} de cada regra não serão máximas na maioria das situações. Por exemplo, se os robôs se aproximam, $d_{x,y}(coe)$ diminui, reduzindo assim a velocidade associada com esta regra. Isto também ocorre com $d_{x,y}(sep)$ e a velocidade de separação no caso em que os robôs se afastam.

4.6 Considerações Finais do Capítulo

Neste capítulo foram apresentadas possíveis implementações de tarefas de enxame baseadas na estratégia de ondas de mensagens. Os robôs executam algoritmos distribuídos que usam a propagação de informação com realimentação para controlar o início e término de cada tarefa. A tarefa de navegação é composta por três subtarefas executadas em sequência: o recrutamento, o alinhamento e a movimentação. Neste capítulo também foram discutidas as tarefas de agregação e dispersão, que são casos particulares do redimensionamento do enxame. Os algoritmos executados localmente por cada robô são representados por máquinas de estado, onde as transições de estado ocorrem em reação ao recebimento de mensagens, ou devido a eventos internos.

O capítulo seguinte apresenta os resultados de simulação das tarefas propostas.

Capítulo 5

Avaliação Experimental

Os comportamentos baseados em ondas de mensagens, bem como o conceito de sequência de subtarefas, são avaliados neste capítulo por meio de simulações computacionais. Primeiramente, diversos cenários são empregados para expor a capacidade de execução do recrutamento e alinhamento. É verificado o tempo de simulação destas duas tarefas em enxames com diferentes quantidades de robôs. Em seguida, a tarefa de navegação é testada em cenários considerando diversos padrões de obstáculos. Depois, as tarefas de agregação e dispersão também são avaliadas considerando o impacto de diferentes fatores, como a quantidade de robôs e o espaçamento entre os mesmos, no tempo total de execução.

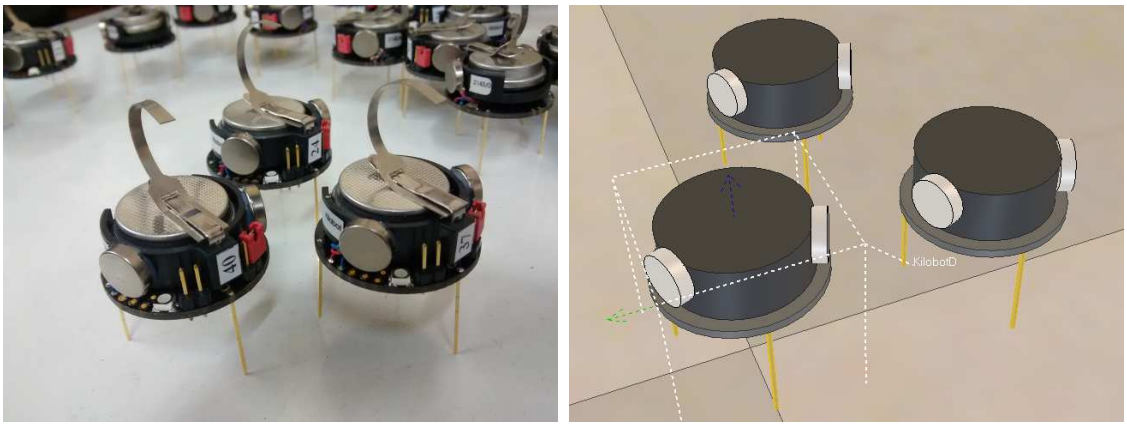
Os enxames apresentados neste capítulo são constituídos de robôs *Kilobots*, simulados no ambiente *V-REP*. Todas as simulações apresentadas neste trabalho foram realizadas em computadores com processador *Intel Core i7 950 3 gigahertz*, *8 gigabytes* de memória e sistema operacional *Microsoft Windows 7 Home Premium*.

Este capítulo está organizado conforme a seguinte descrição. A Seção 5.1 introduz o modelo simulado de robô usado nos experimentos, ressaltando as modificações realizadas para superar as limitações existentes no modelo físico. A contagem de tempo no simulador, usada como métrica de desempenho dos experimentos, é descrita na Seção 5.2. As Seções 5.3 a 5.6 apresentam os resultados obtidos nas simulações de recrutamento, alinhamento, navegação e redimensionamento, respectivamente. Estas seções também descrevem as configurações usadas em cada conjunto de simulações.

5.1 Enxame Simulado de *Kilobots*

A avaliação da estratégia proposta é realizada usando um modelo simulado baseado no robô *Kilobot*, apresentado na Figura 5.1, usando o *software* de simulação *V-REP*. O modelo real de *Kilobot* possui 3 hastes que sustentam a placa de circuito acima da superfície. O robô se movimenta por meio de vibrações resultantes da ativação

de dois motores de passo presentes em cada lado da placa. A vibração permite que o robô realize deslocamentos para frente e faça curvas. Há também um sistema de comunicação IR que permite o envio de mensagens para robôs em uma vizinhança próxima. Um LED colorido posicionado na parte superior do robô permite a sinalização visual de determinados estados internos do algoritmo de controle.



(a) Robô *Kilobot*. (b) *Kilobot* simulado no *V-REP*.

Figura 5.1: Modelo real e simulado do robô *Kilobot*.

O *Kilobot* foi adotado como modelo base para esta pesquisa devido a sua capacidade de comunicação por mensagens. Conforme previamente descrito na Seção 3.2.5, uma mensagem enviada por um robô pode ser recebida por um ou mais robôs presentes em sua vizinhança. Embora esta característica seja essencial para a estratégia de ondas de mensagens, o *Kilobot* apresenta restrições quanto a seu movimento e a sua capacidade de detecção. O uso de um modelo simulado permite superar tais limitações, acrescentando ao modelo de robô recursos inexistentes em sua versão física atualmente existentes no mercado. As alterações no modelo simulado de *Kilobot* empregado nesta pesquisa em relação ao modelo físico incluem:

- O aumento da área de alcance para transmissão de mensagens. Tanto o robô físico quanto o modelo simulado fornecido pelo *V-REP* possuem uma área de alcance muito limitada, necessitando que os robôs estejam muito próximos para que a propagação de mensagens ocorra. Esta proximidade, por sua vez, dificulta a movimentação dos robôs. Assim, a área de comunicação foi aumentada de forma que robôs vizinhos tenham liberdade de movimentação na presença de obstáculos e sem que haja perda da conectividade.

- A inclusão de um sensor de proximidade para uso como sistema de detecção de distância e direção. O *Kilobot* é capaz de detectar a distância em relação a um vizinho a partir da intensidade do sinal da mensagem recebida. Contudo, não há como identificar a posição deste vizinho, nem a presença de outros

vizinhos que não estão enviando mensagens. Foi necessário acrescentar ao modelo simulado um sistema de detecção, que conforme descrito no Capítulo 4, é fundamental em tarefas como a navegação e o redimensionamento.

- A possibilidade de realizar rotações e movimento para trás. O robô físico realiza apenas deslocamento para frente e curvas. A tarefa de alinhamento requer que os robôs possam rotacionar sem que desloquem. Da mesma forma, durante as tarefas de navegação e redimensionamento, é necessário que os robôs apresentem velocidades negativas. O modelo simulado de robô foi modificado para permitir tais tipos de movimentação.

5.2 Tempo de Simulação

A métrica de desempenho avaliada nos testes com os enxames de robôs é o *tempo de simulação*. A contagem de tempo é realizada individualmente pelos robôs. O tempo local do inicializador é tomado como referência. Uma vez que este é o primeiro robô a enviar mensagens e o último a recebê-las, o inicializador é o robô que opera por mais tempo em uma tarefa empregando ondas de mensagens.

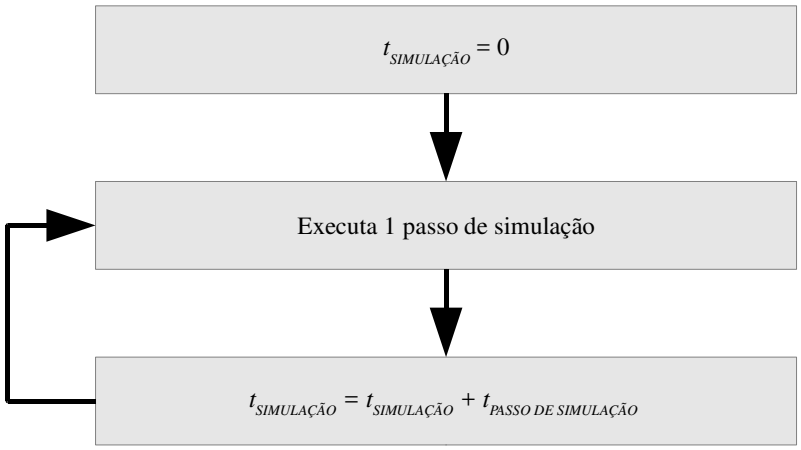
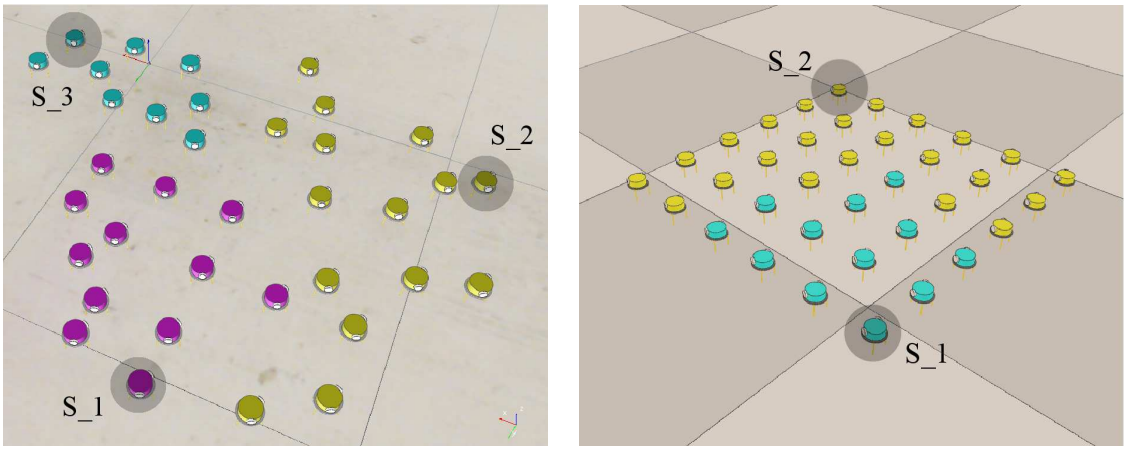


Figura 5.2: Contagem do tempo de simulação no ambiente *V-REP*.

O simulador *V-REP* opera avançando o tempo de simulação em passos constantes, conforme esquematizado pela Figura 5.2. A cada passo de simulação a imagem exibida pelo simulador é atualizada. Vale ressaltar que o tempo de simulação, contado dentro do ambiente simulado, difere do tempo real de execução do simulador. O simulador tenta equivaler o tempo de simulação com o tempo necessário para a execução da mesma ação em um robô físico. Contudo, o aumento da complexidade do ambiente simulado torna o tempo de execução mais lento que o tempo de simulação. Em outras palavras, simulações envolvendo um maior número de robôs necessitarão de um maior tempo de execução para serem concluídas.

5.3 Simulações de Recrutamento

Nesta seção são apresentados os resultados de simulação do recrutamento de robôs. A Figura 5.3 apresenta capturas de tela do simulador *V-REP* ilustrando dois exemplos de execução do recrutamento. Na Figura 5.3(a), três grupos são formados em um enxame composto por 36 robôs dispostos de forma irregular no ambiente[1]. Os três inicializadores, marcados como $S1$, $S2$ e $S3$, começam suas respectivas ondas a partir de regiões distintas do enxame. Quando as mensagens de realimentação são enviadas dos robôs filhos para os pais, os robôs mudam a coloração de seu LED para uma cor associada com o inicializador, indicando assim a qual grupo pertencem. Ao final da execução do recrutamento, o enxame está dividido em três regiões, cada uma contendo robôs recrutados por um inicializador. Na Figura 5.3(b), o enxame está organizado em uma malha de 6×6 robôs[2]. Os dois robôs marcados por $S1$ e $S2$ são inicializadores. O recrutamento mostrou-se possível em ambos os casos, mesmo havendo diferenças nos padrões de vizinhança.



(a) Formação irregular. (b) Robôs dispostos em uma malha.

Figura 5.3: Recrutamento de robôs no simulador *V-REP*.

5.3.1 Configurações da Simulação

As simulações de recrutamento visam analisar o comportamento desta tarefa sob diferentes configurações do enxame. Assim, simulações foram realizadas com enxames de diferentes quantidades de robôs. O efeito da quantidade de inicializadores também foi objeto de interesse. A Tabela 5.1 mostra os diferentes valores adotados para a quantidade total de robôs e de inicializadores.

Duas características foram observadas em cada simulação: o *tempo de recrutamento* e o *tamanho do grupo*. O tempo de recrutamento (τ_R) equivale ao tempo

[1] `https://youtu.be/qnH3UFHwG10`
[2] `https://youtu.be/0JEeGVlOntQ`

Tabela 5.1: Parâmetros de Simulação para o recrutamento.

Tamanho do enxame	4, 8 ,12, 16, 20, 24, 28, 32, 36, 40
Quantidade de inicializadores	1, 2, 3, 4

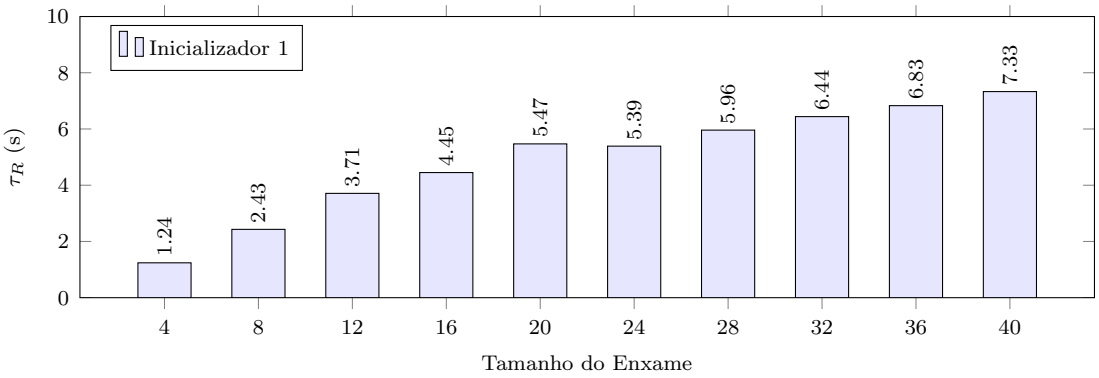
de simulação registrado pelos inicializadores desde o início da transmissão das mensagens até o recebimento da última mensagem de realimentação. O tamanho do grupo (η_G) é a quantidade de robôs na árvore resultante da execução do algoritmo PIF. Esta quantidade compreende o inicializador e os demais robôs que participam da onda por ele propagado. Havendo um único inicializador, η_G será a quantidade total de robôs que constituem o enxame. Em simulações com mais inicializadores, o valor de η_G para cada grupo será uma fração do tamanho do enxame.

Tanto τ_R quanto η_G são grandezas muito sensíveis a variações da disposição espacial dos robôs. Desta forma, para realizar uma avaliação justa, diferentes simulações foram realizadas seguindo uma mesma topologia de malha.

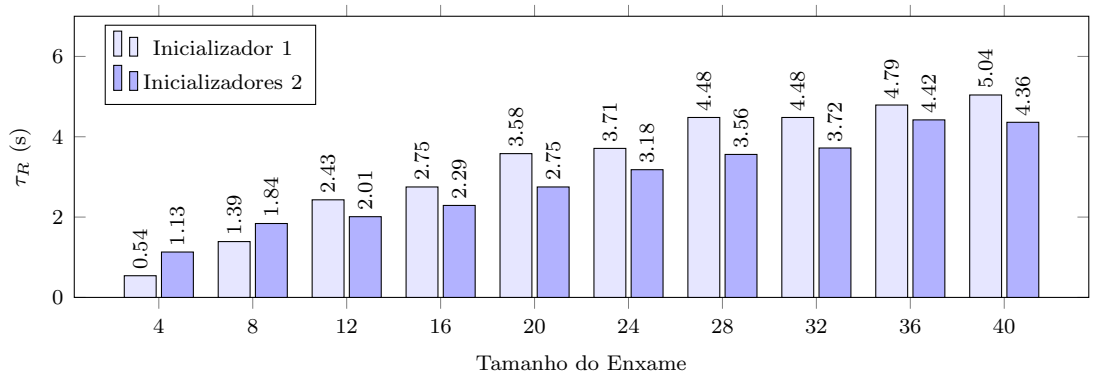
5.3.2 Resultados de Tempo de Recrutamento

A Figura 5.4 apresenta os resultados de tempo de recrutamento (em segundos) para enxames com diferentes quantidades de robôs e de inicializadores [15]. Com 1 inicializador, τ_R cresce com o tamanho do enxame, conforme observado na Figura 5.4(a). Quanto maior a quantidade de robôs que compõem o enxame, maior é o tempo necessário para transmitir as mensagens através dos robôs e completar o recrutamento. Este aumento de τ_R em função do tamanho do enxame ocorre também em simulações com mais inicializadores. As Figuras 5.4(b), 5.4(c) e 5.4(d) apresentam os valores obtidos para τ_R em simulações de enxames com 2, 3 e 4 inicializadores. Em uma simulação com um determinado tamanho de enxame, cada inicializador registra seu próprio valor obtido de τ_R. A diferença nos valores de τ_R indicam que os inicializadores recrutam diferentes quantidades de robôs. Isto é esperado, por exemplo, nos casos em que o tamanho do enxame dividido pela quantidade de inicializadores não resulta em um valor inteiro. Além disto, um inicializador específico não tem nenhuma preferência sobre os demais inicializadores durante o recrutamento. Logo, este resultado também pode ser afetado por eventuais atrasos na transmissão de mensagens.

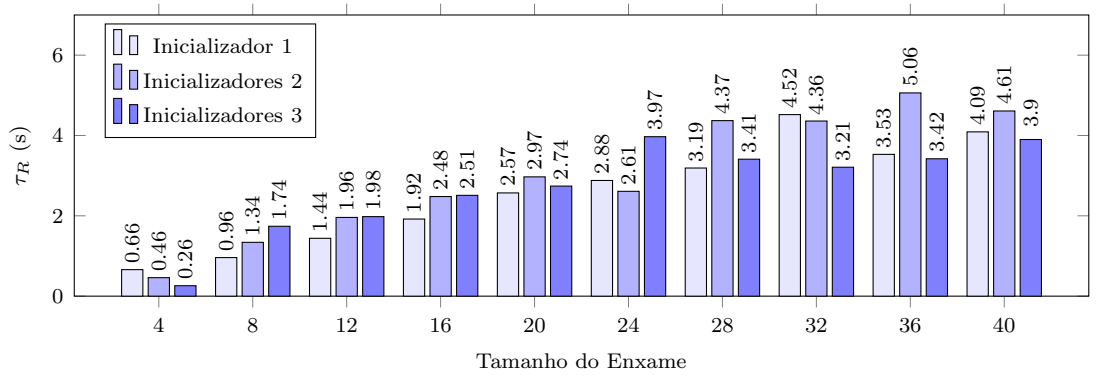
Em uma segunda análise, o *Tempo Médio de Recrutamento* ($\overline{\tau_R}$) é avaliado considerando as simulações com 2, 3 e 4 inicializadores. O valor de $\overline{\tau_R}$ é a média aritmética dos resultados obtidos por diferentes inicializadores em simulações com a mesma configuração de enxame. O uso desta métrica visa reduzir o efeito de grupos com diferentes quantidades de robôs. A Figura 5.5 relaciona $\overline{\tau_R}$ com o tamanho do enxame em simulações com mais de um inicializador. Como observado para τ_R



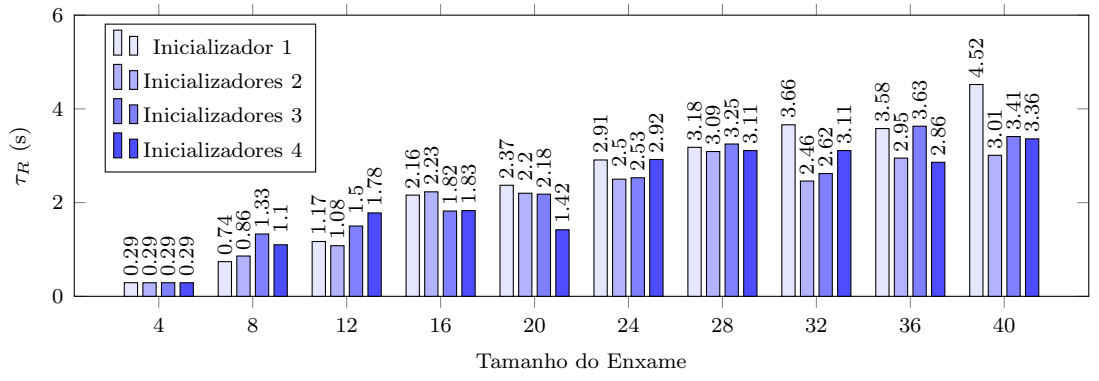
(a) Recrutamento com 1 inicializador.



(b) Recrutamento com 2 inicializadores.



(c) Recrutamento com 3 inicializadores.



(d) Recrutamento com 4 inicializadores.

Figura 5.4: Impacto do número de robôs e do número de inicializadores em τ_R.

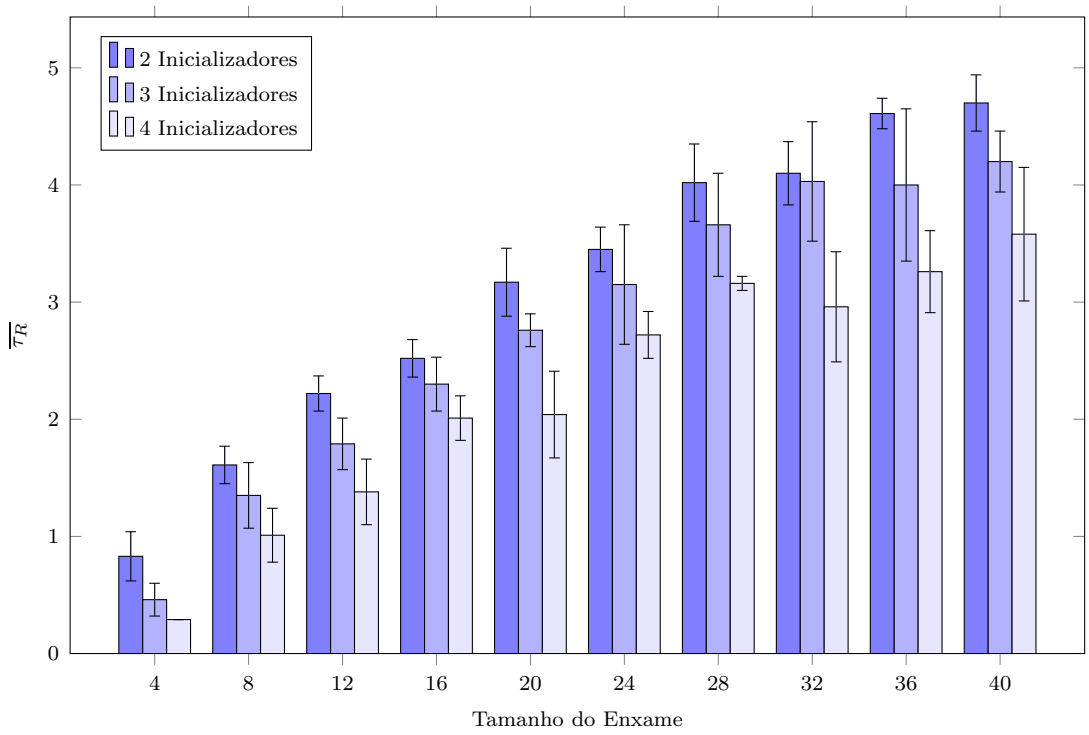


Figura 5.5: Tempo médio de recrutamento em testes com mais de 1 inicializador.

de cada inicializador, o tempo médio de recrutamento aumenta com a quantidade de robôs presentes no enxame. No entanto, a informação principal presente nesta figura é a comparação entre os resultados obtidos para diferentes quantidades de inicializadores. Quanto maior for o número de inicializadores presentes, menor será $\overline{\tau_R}$. Este efeito pode ser explicado observando o tamanho dos grupos, ou a quantidade de robôs recrutados em cada simulação. Com apenas um inicializador, todos os membros do enxame serão recrutados por um mesmo robô. Assim, o tamanho do grupo será igual ao tamanho do enxame. Nas situações com mais inicializadores, mas com o mesmo número total de robôs, a quantidade de membros do enxame é dividida em grupos. Assim, quanto mais grupos existirem, menor será a quantidade de robôs em cada grupo, e menor o tempo para o recrutamento dos mesmos.

Uma terceira análise envolve a relação entre o tempo de recrutamento (τ_R) necessário para a formação de cada grupo e o tamanho destes grupos. Na Figura 5.6, cada ponto indica o tempo de recrutamento de um determinado grupo e o tamanho deste grupo. É relevante observar que todas as simulações seguem uma mesma tendência, independente do número de inicializadores presentes. A dispersão dos pontos associados aos testes com 2, 3 e 4 inicializadores ocorre pois, em uma mesma simulação, cada inicializador recruta um diferente número de robôs. Nos casos em que a divisão é mais balanceada, os pontos tendem a se aproximar da reta resultante do caso com um único inicializador.

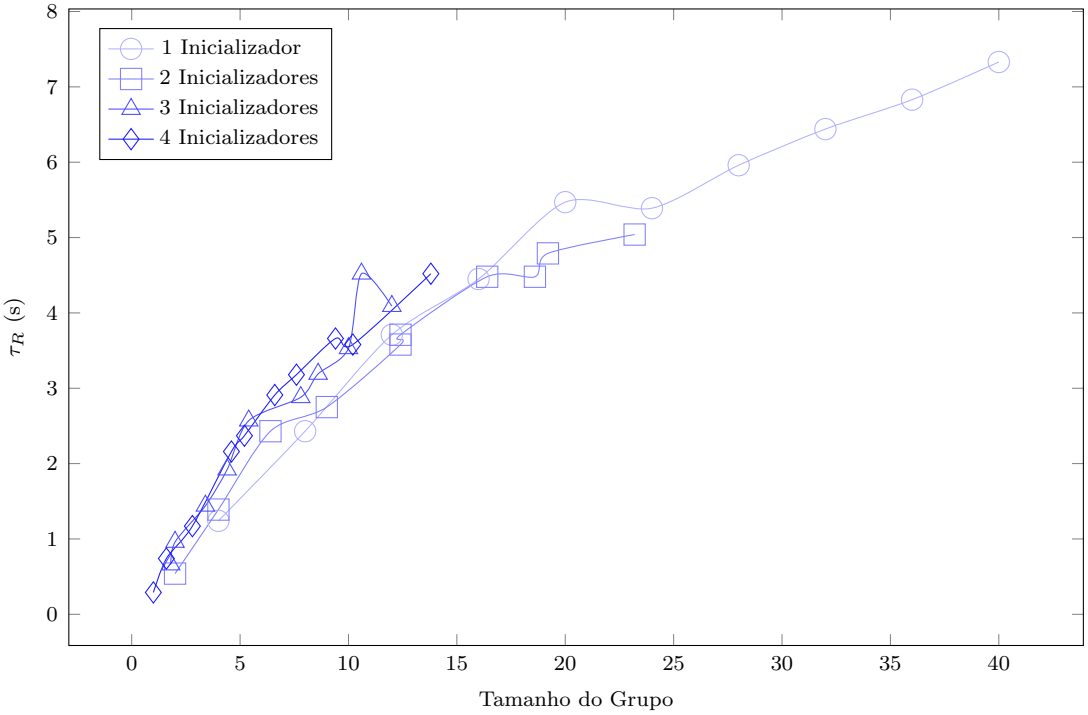


Figura 5.6: Tempo necessário para formação dos grupos com diferentes tamanhos.

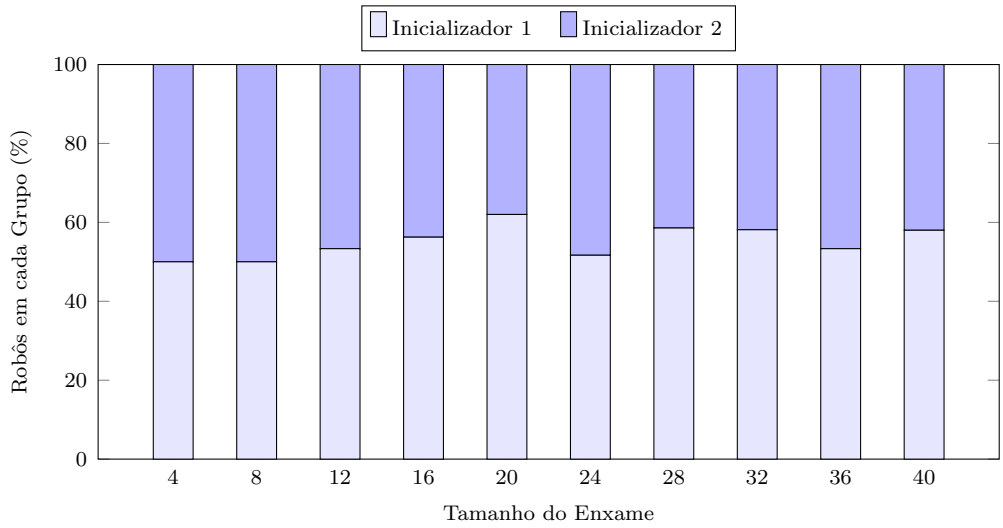
5.3.3 Tamanho dos Grupos

Pelos resultados obtidos, é possível observar que o recrutamento não é balanceado em relação ao tamanho dos grupos. Esta característica pode ser definida como a capacidade do enxame se organizar em grupos do mesmo tamanho, ou pelo menos, sem uma diferença significativa entre o número de robôs de cada grupo. O balanceamento, ou qualquer outro controle no tamanho dos grupos, é uma característica importante na divisão do trabalho. Contudo, o tamanho de cada grupo é algo que dependerá da tarefa para a qual os robôs estão sendo recrutados.

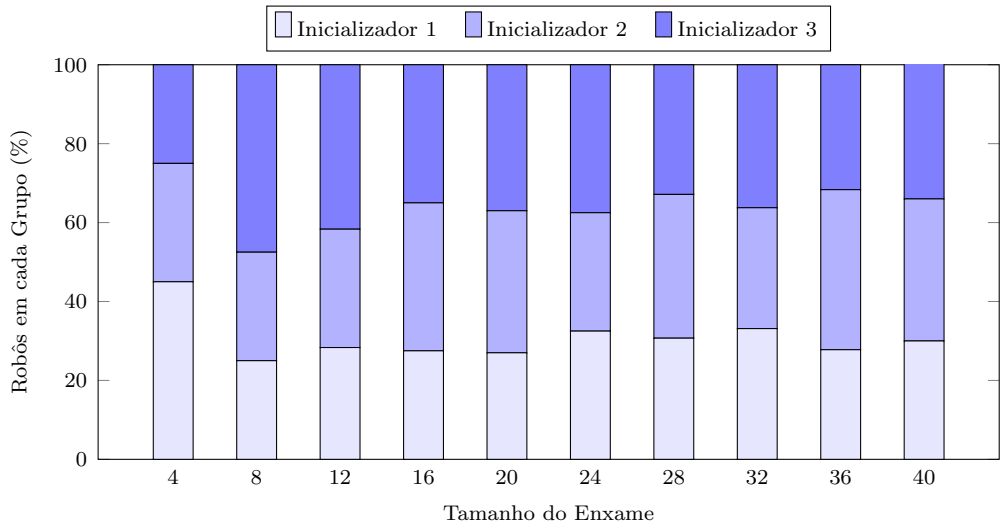
O recrutamento é executado sem um controle específico sobre a origem da onda. Um robô não-inicializador que recebe uma mensagem de recrutamento participa da onda do inicializador que começou a propagação das mensagens. Havendo mais de um inicializador, os não-inicializadores apenas aguardarão pelo recebimento da primeira mensagem, que pode ter como origem qualquer um dos inicializadores.

Mesmo não havendo um controle específico para o tamanho dos grupos, a porcentagem dos robôs recrutados pouco difere de um inicializador para outro nas simulações realizadas. Isto pode ser visto na Figura 5.7 para 2, 3 e 4 inicializadores. Alguns fatos podem explicar este comportamento:

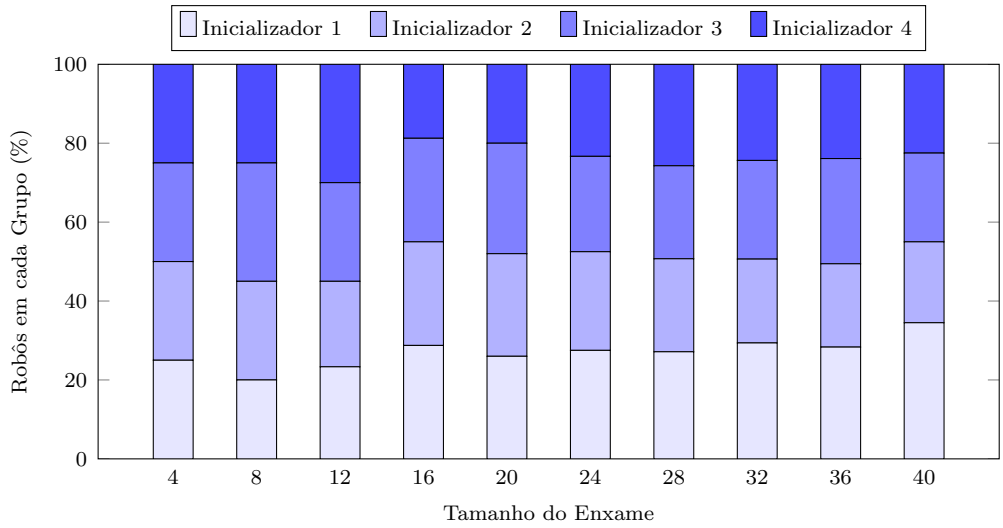
- *Todos os inicializadores começam sua transmissão simultaneamente.* Embora cada robô execute seu próprio algoritmo isoladamente, dependendo apenas da recepção de mensagens para prosseguir para o estado seguinte, a simulação



(a) Enxame com 2 inicializadores.



(b) Enxame com 3 inicializadores.



(c) Enxame com 4 inicializadores.

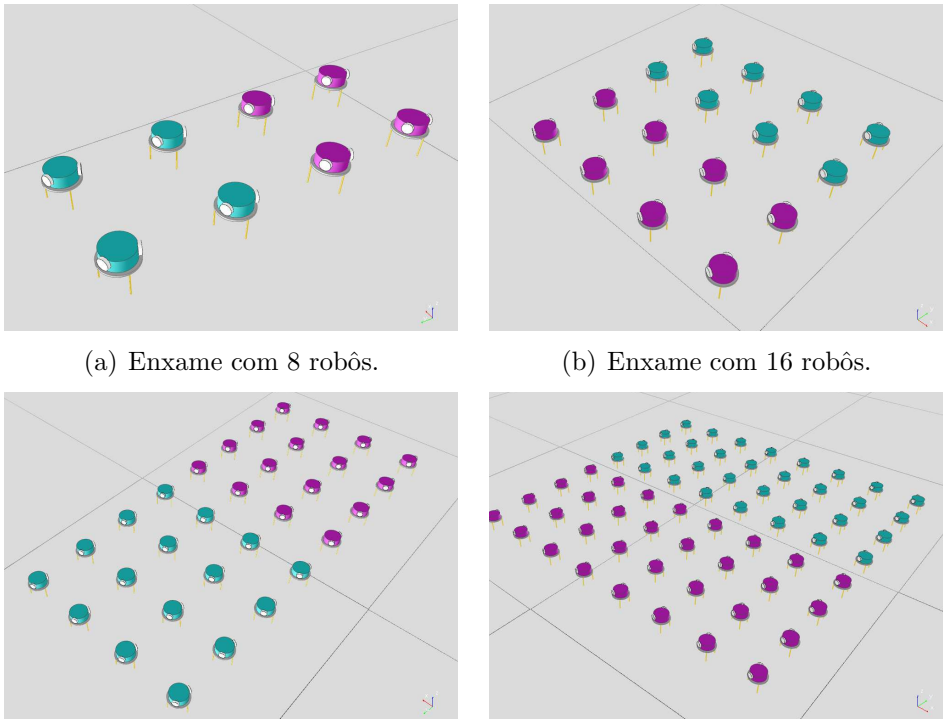
Figura 5.7: Proporção dos grupos recrutados por cada inicializador.

começa no mesmo instante para todos os robôs. O enxame como um todo é um sistema com comunicação assíncrona, o que explica porque alguns grupos possuem mais robôs que outros. Caso um inicializador específico começasse sua onda antes dos demais inicializadores, então este seria capaz de recrutar mais robôs que os demais.

- *Os robôs estão organizados em uma malha.* A onda de mensagens mostrou-se capaz de recrutar robôs em diferentes disposições espaciais, como ilustrado pela Figura 5.3. O recrutamento ocorre independente da organização de vizinhança, tendo como única restrição a conectividade entre robôs. No entanto, visando organizar as simulações e ter uma melhor avaliação dos resultados, o enxame foi disposto em uma malha (estrutura regular). Robôs ao centro do enxame possuem todos quatro vizinhos ao seu redor, enquanto que os robôs posicionados na borda possuem três ou dois vizinhos. Esta regularidade na vizinhança permite que a onda tenha uma propagação mais uniforme pelo enxame. Caso fossem adotadas topologias irregulares, a quantidade de robôs recrutados em cada grupo seria imprevisível.

- *Inicializadores estão localizados nos cantos da malha.* Para realizar simulações que fossem todas similares entre si, os cantos do enxame foram escolhidos como posição para os inicializadores. Quanto maior a distância entre os iniciadores, menor será o efeito de uma onda que recruta robôs próximos a outro inicializador, resultando assim em uma divisão mais uniforma do enxame.

Embora a Figura 5.7 mostre que existe um eventual balanceamento no tamanho dos grupos, este é influenciado por diversos fatores, conforme descrito acima. É possível citar ainda como característica que permitirá ou não o balanceamento, o posicionamento dos inicializadores [18]. A Figura 5.8 [14] mostra o recrutamento realizado por dois inicializadores em enxames com 8, 16, 32 e 64 robôs. A cor dos robôs indica a qual grupo cada um pertence. Os enxames se encontram dispostos em uma formação retangular, com os dois inicializadores localizados nos cantos superior esquerdo e superior direito dos enxames ilustrados na Figura 5.8. O posicionamento dos inicializadores proporciona a formação de grupos simétricos e balanceados, isto é, ambos com a mesma quantidade de robôs. Este balanceamento pode, contudo, não ocorrer se os inicializadores estiverem em outras posições. Isto é exemplificado na Figura 5.9. Nos quatro casos, os inicializadores estão em posições opostas nos cantos da malha. Em cada exemplo, um dos inicializadores recruta os robôs na diagonal do enxame, resultando assim grupos de tamanhos diferentes.



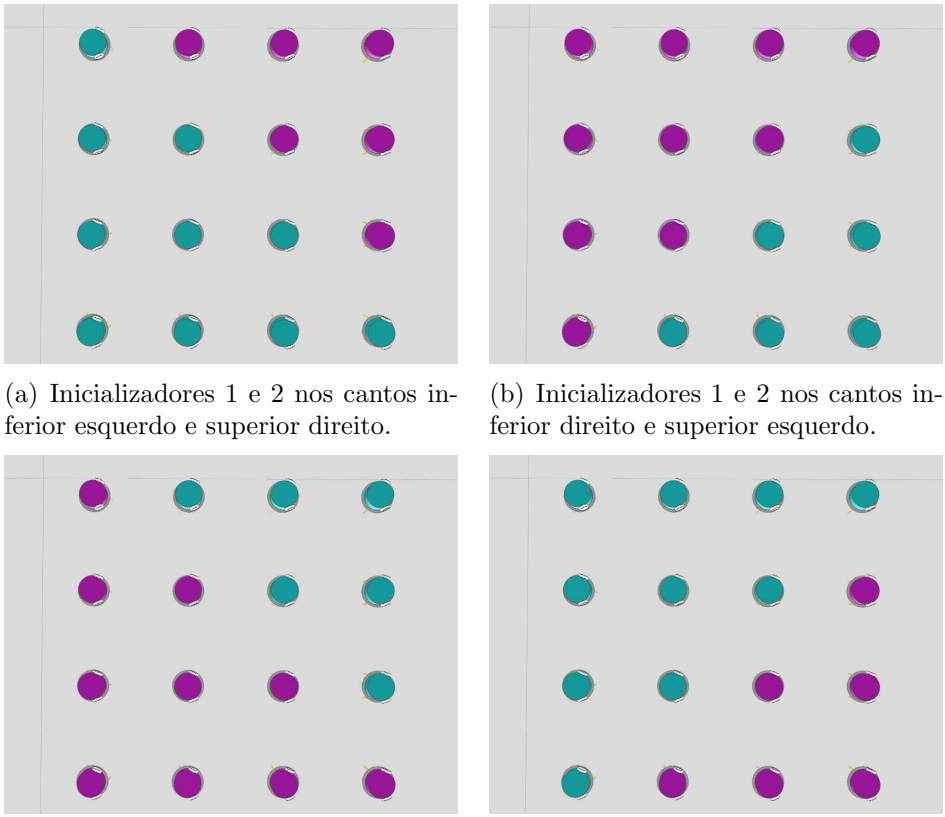
(a) Enxame com 8 robôs.

(b) Enxame com 16 robôs.

(c) Enxame com 32 robôs.

(d) Enxame com 64 robôs.

Figura 5.8: Enxames com dois inicializadores e grupos de mesmo tamanho.



(a) Inicializadores 1 e 2 nos cantos inferior esquerdo e superior direito.

(b) Inicializadores 1 e 2 nos cantos inferior direito e superior esquerdo.

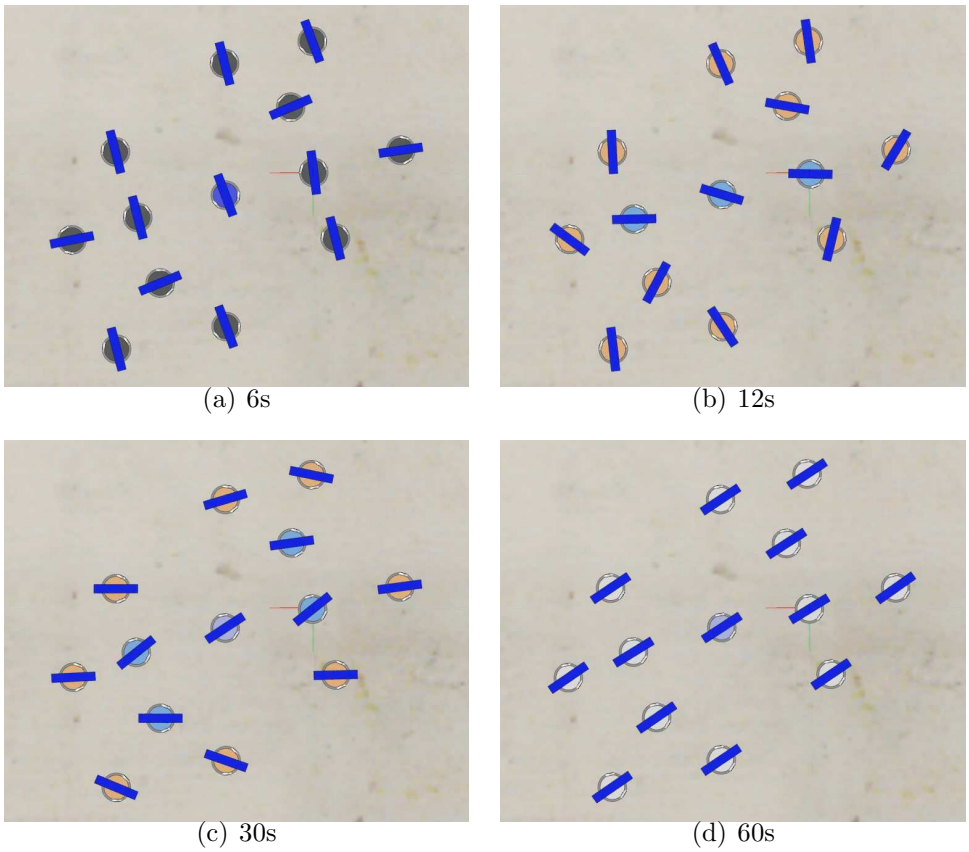
(c) Inicializadores 1 e 2 nos cantos superior direito e inferior esquerdo.

(d) Inicializadores 1 e 2 nos cantos superior esquerdo e inferior direito.

Figura 5.9: Enxames com dois inicializadores e grupos de tamanhos diferentes.

5.4 Simulações de Alinhamento

Nesta seção os resultados de simulações do alinhamento de robôs são apresentados [17]. A Figura 5.10 ilustra um exemplo de alinhamento no *V-REP*[3]. A marcação acima de cada robô é usada para indicar o direcionamento de cada um dos 13 robôs. Os robôs se alinham em relação ao inicializador central, que inicialmente encontra-se em rotação. Ao término do alinhamento, o inicializador retoma a rotação, reiniciando o processo.



(a) 6s (b) 12s

(c) 30s (d) 60s

Figura 5.10: Exemplo de simulação de alinhamento.

5.4.1 Caminho Crítico

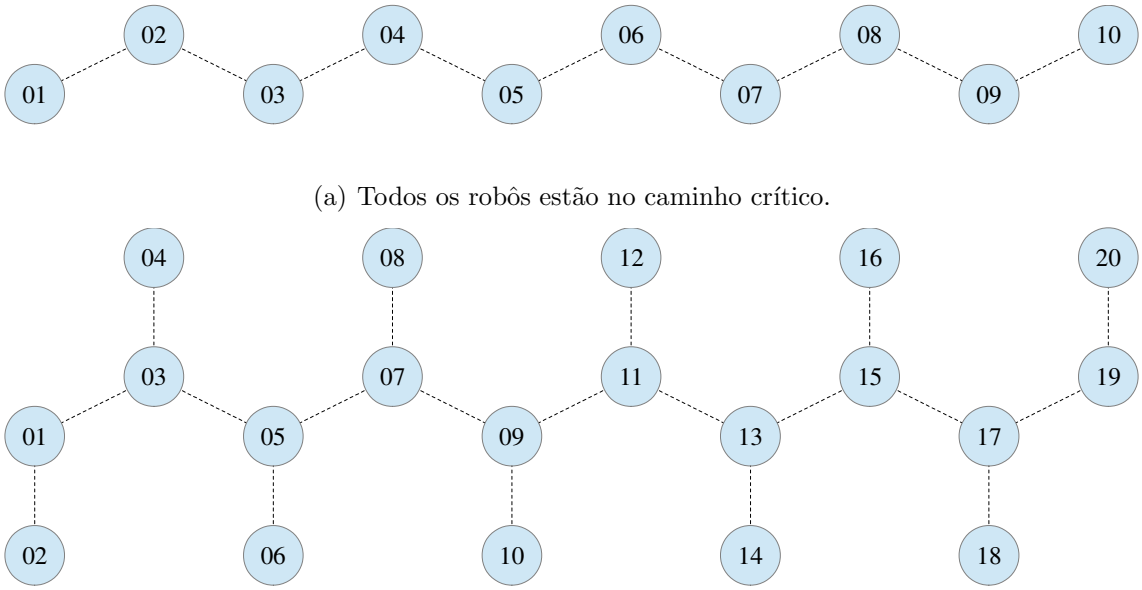
As simulações de alinhamento têm por objetivo avaliar o impacto do tamanho do enxame no *Tempo de Alinhamento* (τ_A). O alinhamento é iniciado após uma subtarefa de recrutamento inicial, executada para que os robôs conheçam sua vizinhança e a relação entre robôs pais e filhos.

Assim como as simulações envolvendo o recrutamento, também foi adotada uma topologia de malha para os diferentes testes de alinhamento. Esta restrição é necessária para minimizar o impacto de diferentes topologias na propagação da onda

[3] `https://youtu.be/axnyoBvqNBI`

de mensagens. Em testes iniciais, contudo, o valor de τ_A não apresentou um comportamento diretamente relacionado com o número de robôs no enxame. Em certas situações, o aumento do número de robôs em pouco influenciou o valor de τ_A. Em outras situações, τ_A variou bruscamente em testes envolvendo o mesmo número de robôs, mas com o inicializador localizado em diferentes posições. Este comportamento indicou que o tempo de alinhamento depende não apenas da quantidade de robôs, mas também do *caminho crítico* presente no enxame.

Definição 6 *O caminho crítico em um enxame é o percurso a partir do inicializador até algum robô sem filhos que possui o maior número de robôs intermediários. É a maior cadeia causal entre robôs pais e robôs filhos. O número de robôs no caminho crítico é representado por* ρ_C.



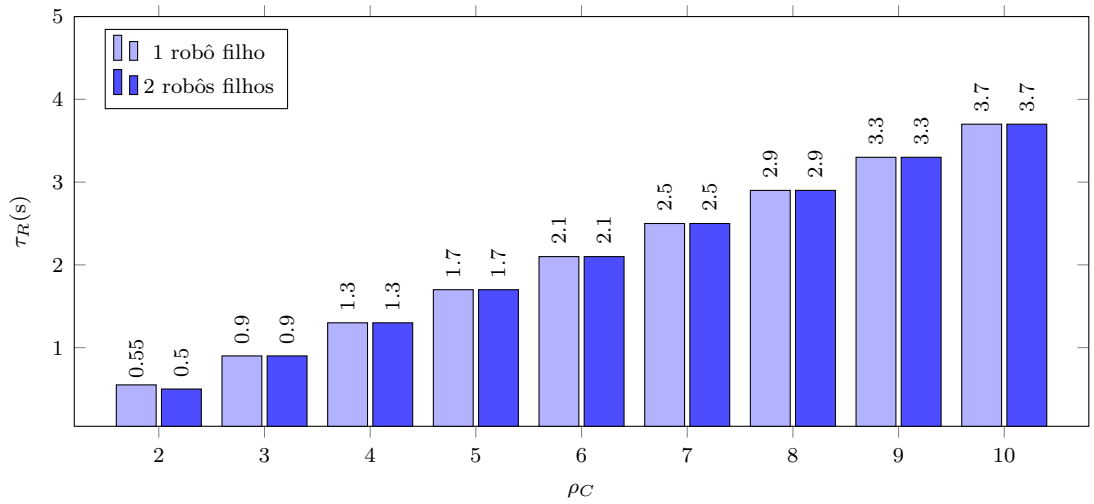
(a) Todos os robôs estão no caminho crítico.

(b) Todos os robôs no caminho crítico, exceto o último, possuem 2 filhos.

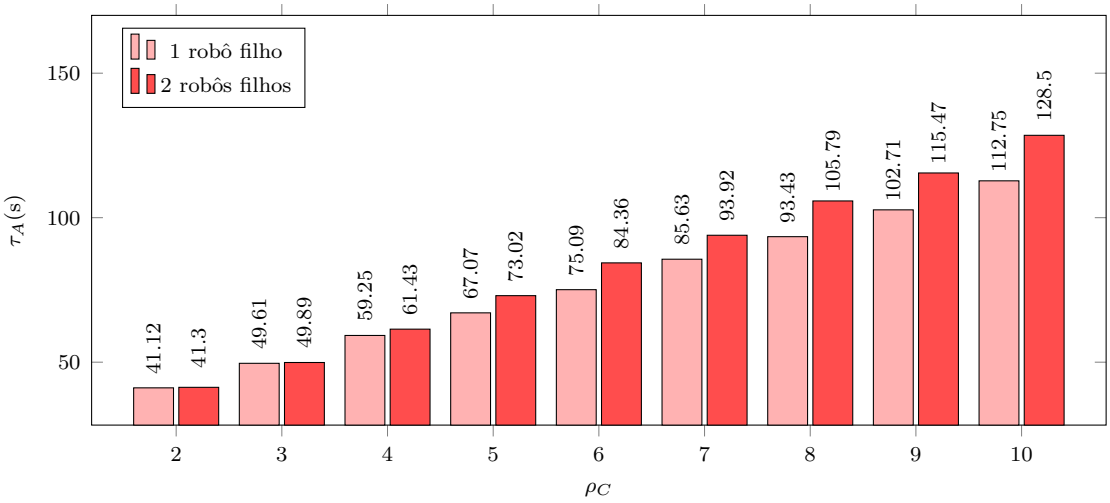
Figura 5.11: Disposição espacial dos robôs na avaliação do caminho crítico.

Uma série de simulações foi realizada para verificar o impacto do caminho crítico na propagação da onda no alinhamento. A disposição espacial dos robôs é mostrada na Figura 5.11, onde o número indica o *id* de cada robô. No primeiro conjunto de testes[4], o número de robôs no caminho crítico, ρ_C, varia 1 até 10, conforme apresentado pela Figura 5.11(a). Neste caso, cada robô intermediário no caminho crítico possui um único filho. Em um segundo conjunto de testes[5], ρ_C novamente varia entre 1 e 10. Neste caso, contudo, os robôs intermediários possuem 2 filhos cada um, e o caminho crítico é composto pelos robôs de *id* com numeração ímpar. Em

[4] `https://youtu.be/Mq4VW_WtOEY`
[5] `https://youtu.be/2ipkpalLAVU`



(a) Tempo de Recrutamento.



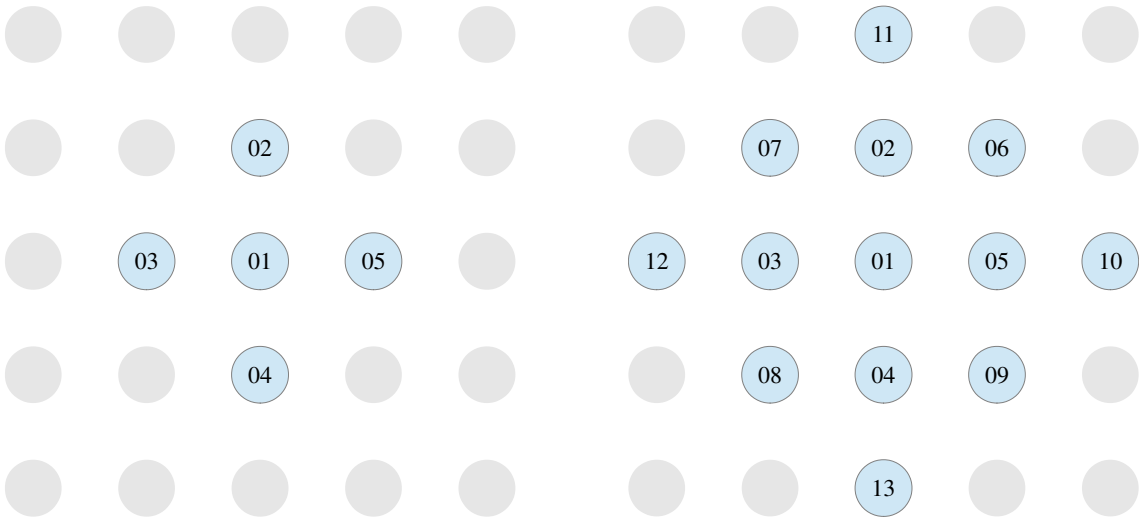
(b) Tempo de Alinhamento.

Figura 5.12: Impacto do número de robôs no caminho crítico.

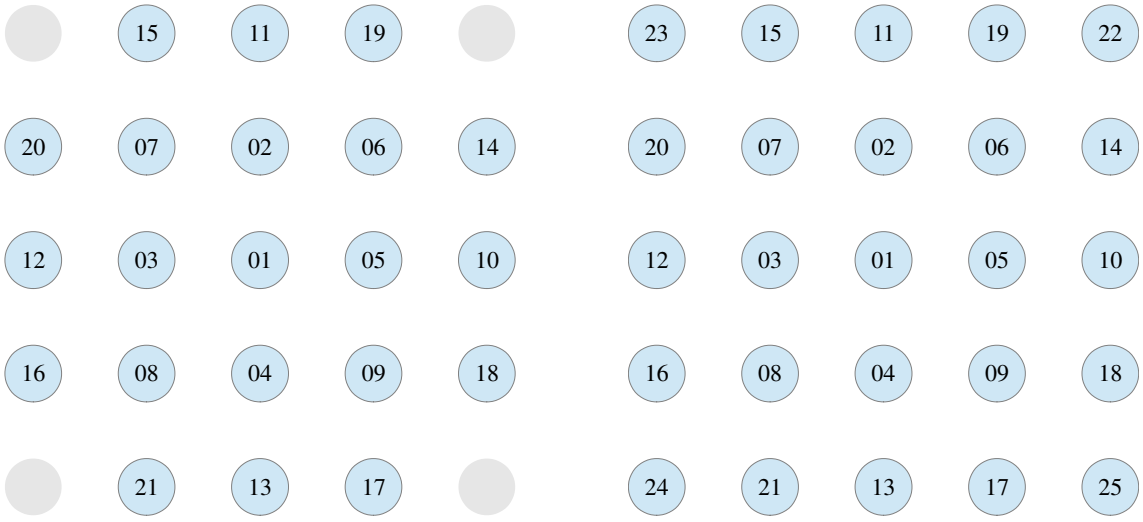
ambos os conjuntos de testes foi realizado o alinhamento com o inicializador (o robô com $id = 1$) inicialmente disposto a 180° em relação aos demais membros do enxame. Os resultados de simulação são apresentados na Figura 5.12. Como o alinhamento utiliza uma subtarefa inicial de recrutamento, o tempo total de execução é composto por τ_R, o tempo de recrutamento, e τ_A, o tempo de alinhamento. A Figura 5.12(a) mostra que τ_R cresce linearmente com o aumento do número de robôs no caminho crítico. Para este conjunto de testes, não houve uma diferença significativa entre os valores obtidos de τ_R em simulações com mesmo rho_C e diferentes quantidades de robôs filhos. Por outro lado, τ_A mostrou-se dependente tanto de ρ_C quanto do número de filhos dos robôs no caminho crítico. Com $\rho_C = 2$, o acréscimo de 1 robô filho resulta em um aumento de $0,44\%$ no valor de τ_A. Com $\rho_C = 10$, esta variação no número de filhos resulta em um aumento de $13,97\%$ no valor obtido de τ_A.

5.4.2 Resultados de Tempo de Alinhamento

Uma avaliação sistemática do tempo de alinhamento necessita considerar o aumento do número de robôs e o aumento do caminho crítico. Ao mesmo tempo, é de interesse que uma disposição espacial regular seja mantida. Assim, ao aumentar o tamanho do enxame, os robôs devem ser acrescentados em uma posição que aumente minimamente ρ_C e obedecendo a topologia de malha. É possível obter um conjunto de testes que obedeça a tais características seguindo a disposição espacial representada pela Figura 5.13.



(a) $\rho_C = 2$ (b) $\rho_C = 3$

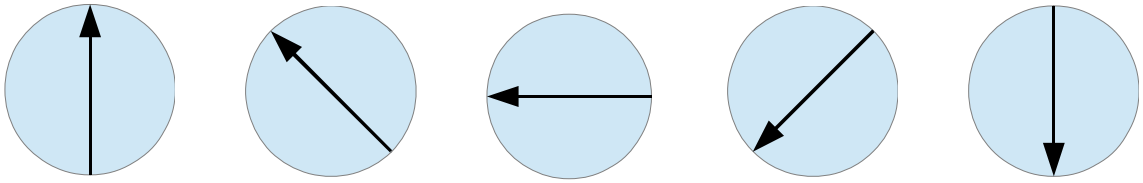


(c) $\rho_C = 4$ (d) $\rho_C = 5$

Figura 5.13: Disposição espacial dos robôs.

A posição do robô inicializador ($id = 01$) é considerada o centro do enxame. Os demais robôs são acrescentados em posições ao seu redor, de forma a aumentar minimamente o caminho crítico. Como ilustrado na Figura 5.13(a), as simulações com 2, 3, 4 e 5 robôs possuem todas $\rho_C = 2$. Seguindo esta disposição espacial, as simulações onde a quantidade de robôs varia entre 6 e 13 possuem $\rho_C = 3$ (Figura 5.13(b)), enquanto que os testes entre 14 e 21 robôs possuem $\rho_C = 4$ (Figura 5.13(c)), e entre 22 e 25 robôs, $\rho_C = 5$ (Figura 5.13(d)).

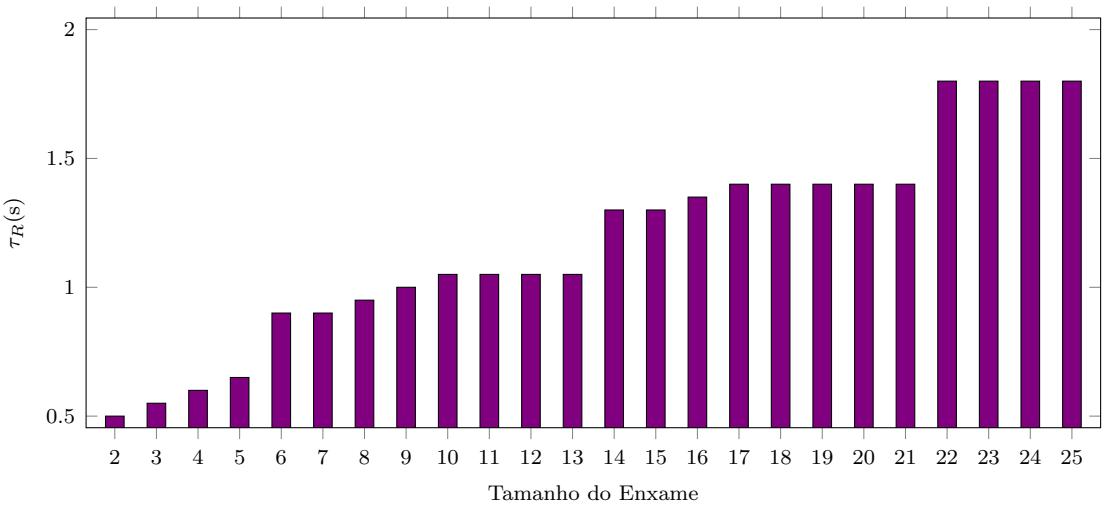
Uma outra característica verificada nas simulações é o efeito da diferença entre o direcionamento do inicializador e dos demais robôs. Para um determinado teste, o direcionamento do inicializador é considerado como referência, enquanto que os demais robôs encontram-se rotacionados de um ângulo α em relação ao inicializador. O valor de α varia em diferentes testes entre 0° (totalmente alinhados) e 180° (totalmente desalinhados). Os diferentes alinhamentos adotados são mostrados na Figura 5.14. É importante ressaltar que o movimento dos robôs durante o ajuste de direção não ocorre com velocidade constante. Na regra descrita no Algoritmo 7, um robô rotaciona com velocidade máxima quando o valor medido de α, que é a diferença de direcionamento em relação a seu robô pai, encontra-se entre 90° e 180°. Para valores de α abaixo de 90°, a velocidade de rotação é proporcional a $sen\alpha$. Desta forma, o valor da velocidade é reduzido a medida que o robô filho encontra-se mais próximo do direcionamento de seu pai. Esta redução de velocidade é necessária para evitar que o robô filho ultrapasse o direcionamento de seu pai, evitando assim oscilações no entorno de $\alpha = 0°$.



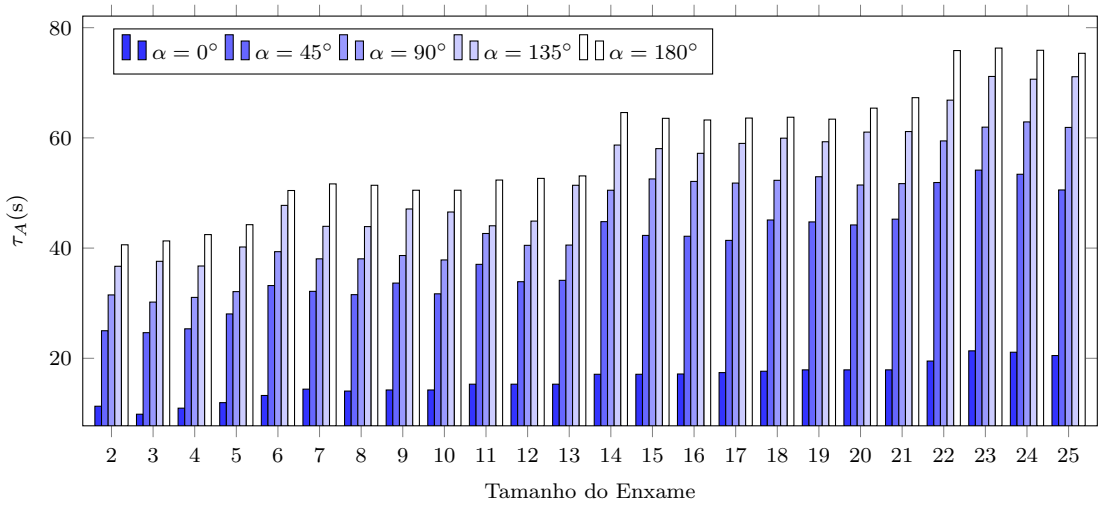
(a) $\alpha = 0°$ (b) $\alpha = 45°$ (c) $\alpha = 90°$ (d) $\alpha = 135°$ (e) $\alpha = 180°$

Figura 5.14: Direcionamento inicial dos robôs não-inicializadores.

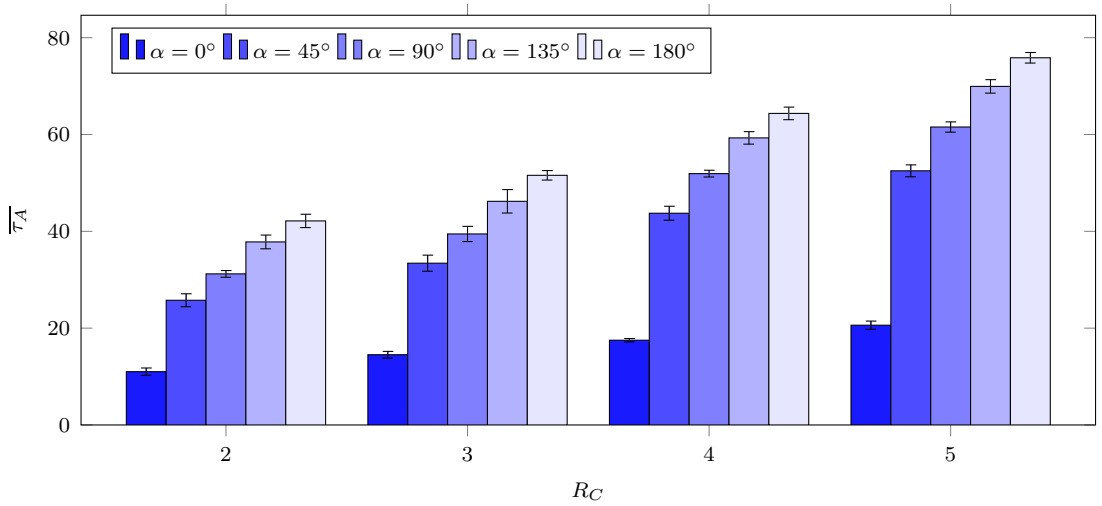
Os resultados obtidos nas simulações de alinhamento são apresentados na Figura 5.15. A subtarefa de recrutamento não é afetada pelo direcionamento inicial dos robôs. Desta forma, apenas uma série de valores é representada na Figura 5.15(a), indicando a relação de τ_R com o número total de robôs. Observa-se que τ_R de fato aumenta com o tamanho do enxame. Há ainda um forte impacto causado pelo aumento de ρ_C, como pode ser visto na variação de 5 para 6 robôs, de 13 para 14 robôs e 21 para 22 robôs.



(a) Tempo de Recrutamento.



(b) Tempo de Alinhamento.



(c) Tempo Médio de Alinhamento.

Figura 5.15: Resultados obtidos nas simulações de alinhamento.

Ao contrário do que ocorre durante o recrutamento, o alinhamento de robôs é diretamente afetado pelo direcionamento dos robôs em relação ao inicializador. Esta característica pode ser visualizada pelos valores obtidos para τ_A em diferentes configurações de simulação, representados na Figura 5.15(b). Simulações com o maior grau de desalinhamento, onde os não-inicializadores todos apontam 180° em relação ao inicializador, resultaram nos maiores valores obtidos de τ_A. Nesta situação, os degraus em τ_A com 6, 14 e 22 robôs são facilmente notados. Nos experimentos com o tamanho do enxame variando entre 2 e 25 robôs, a dependência em relação ao caminho é verificada com ρ_C variando de 1 até 4. Para um mesmo tamanho de enxame, os diferentes valores obtidos de τ_A, na Figura 5.15(b), indicam que diferentes direcionamentos necessitam de mais ou menos tempo para a conclusão do alinhamento. Estes valores não seguem uma variação linear pois a velocidade de rotação dos robôs não é constante, conforme discutido previamente na Seção 5.4.1.

É possível ainda analisar de uma outra forma o impacto dos diferentes parâmetros no tempo de alinhamento. O *Tempo Médio de Alinhamento* ($\overline{\tau_A}$) é a média aritmética dos valores obtidos de τ_A em simulações com mesmos α e ρ_C, mas com diferentes tamanhos de enxame. A Figura 5.15(c) apresenta o comportamento de $\overline{\tau_A}$. Nota-se claramente a pequena variação do tempo de alinhamento devido ao aumento do enxame em testes com mesmo α e ρ_C. Esta grandeza, contudo, aumenta de forma bem mais significativa considerando variações de α e ρ_C.

É notável o fato de que a subtarefa de alinhamento necessita de mais tempo que o recrutamento para ser concluída. Comparando os tempos de recrutamento e alinhamento para o experimento envolvendo 25 robôs e $\alpha = 0°$, onde os robôs encontram-se inicialmente alinhados, os valores obtidos de τ_R são em torno de 10 vezes menores que os de τ_A. Considerando o experimento de 25 robôs com o maior desalinhamento, isto é, com $\alpha = 180°$, o valor obtido de τ_A é em torno de 40 vezes menor que τ_R. É importante ressaltar que as duas subtarefas possuem naturezas distintas. O recrutamento é baseado tão somente na propagação de mensagens, dependendo unicamente da comunicação entre robôs para ser realizado. O alinhamento, ainda que também faça uso do conceito de onda de mensagens, é uma tarefa muito mais complexa. Os robôs precisam manter comunicação com a vizinhança e verificar repetidamente se estão alinhados em relação ao robô pai. Mesmo nos experimentos em que os robôs encontram-se inicialmente alinhados ($\alpha = 0°$), existe um τ_A necessário para a propagação da onda. Para os demais casos, existe a rotação dos robôs, que requer mais ou menos tempo dependendo do grau de alinhamento em relação ao inicializador.

5.5 Simulações de Navegação Coletiva

Nesta seção são discutidas as simulações de navegação coletiva [16, 17]. Esta é uma tarefa complexa, implementada principalmente para ilustrar a ideia de sequenciamento de subtarefas. Conforme descrito na Seção 4.4, a navegação coletiva é realizada pela sequência de três subtarefas: o recrutamento, o alinhamento e a movimentação coordenada. Tanto o recrutamento quanto o alinhamento foram discutidos em seções anteriores. Nesta seção,o objetivo é avaliar o comportamento de movimentação coordenada observado em diferentes simulações.

5.5.1 Impacto de Diferentes Cenários

Durante a movimentação coordenada, o grupo de robôs recrutado pelo inicializador se desloca pelo ambiente, de forma que nenhum membro perca conectividade com o grupo. O movimento é baseado em uma estratégia seguidor de líder, onde cada filho segue seu respectivo pai, sendo o inicializador o líder geral do grupo.

Ao contrário da metodologia adotada para as demais subtarefas, métricas baseadas no tempo de simulação não foram empregadas para o caso da navegação coletiva . A movimentação dos robôs é uma operação que depende de inúmeros fatores, tais como a velocidade dos robôs, a trajetória percorrida pelo enxame e a posição de eventuais obstáculos no ambiente. Dessa forma, o tempo de simulação é ema grandeza que pode variar imensamente de uma simulação para outra, sem que isto indique alguma característica relevante do experimento. Ao invés disso, verificou-se a capacidade do enxame de realizar deslocamentos mantendo seu agrupamento.

5.5.2 Experimentos

Dez diferentes experimentos foram realizados para a verificação do funcionamento da tarefa de navegação coletiva. Tais simulações envolvem o uso de um ou mais grupos de robôs em ambientes abertos ou com obstáculos. Os diferentes cenários simulados são sucintamente descritos na Tabela 5.2. Cada um dos experimentos é descrito com detalhes no Apêndice A. Os vídeos das simulações estão disponíveis para visualização *online* em [12]. Em todos os casos a tarefa de navegação pôde ser realizada, isto é, seguindo a sequência de três subtarefas. Além disso, a movimentação transcorreu sem que robôs se perdessem do enxame.

O esquema de movimentação coordenada desenvolvido como subtarefa na navegação coletiva possui limitações. Uma delas diz respeito à posição dos robôs pais em relação aos filhos. Cada robô precisa estar localizado em um ponto (x, y) no sistema de coordenadas de seu respectivo robô pai, tal que $y < 0$. Robôs pais precisam estar a frente de seus robôs filhos para que estes possam segui-lo. Isto é explicado

Tabela 5.2: Experimentos de navegação coletiva.

Experimento	Descrição
1	Grupo de robôs com interrupção do movimento.
2	Grupo de robôs com movimento circular.
3	Dois grupos de robôs.
4	Grupo de robôs em ambiente fechado.
5	Grupo atravessando barreira de robôs.
6	Grupo em ambiente com redução espacial.
7	Grupo de robôs contornando parede.
8	Dois grupos em ambiente com obstáculos.
9	Três grupos em ambiente com obstáculos.
10	Quatro grupos em ambiente fechado e obstáculos.

pela característica não-holonômica da movimentação dos robôs. Outra limitação observada está associada com a quantidade de robôs na vizinhança. Embora cada robô necessite de comunicação apenas com seu pai e eventuais filhos, é possível que haja a recepção de mensagens enviadas por outros vizinhos que se encontrem dentro da área de alcance de comunicação. Estas mensagens não são usadas, e atrasam o recebimento de mensagens vindas de pais e filhos. Em testes iniciais realizados como mais de 6 robôs na vizinhança, observou-se a ocorrência do efeito de *starvation*, quando algumas das mensagens nunca são transmitidas ao destinatário ao número excessivo de mensagens de outros robôs. Em testes onde estas duas condições não foram respeitadas o enxame não foi capaz de realizar a movimentação coordenada. Desta forma, nos experimentos apresentados nesta seção, evitou-se a ocorrência de tais situações.

5.5.3 Manutenção da Formação

Em todas as simulações de navegação, a disposição espacial dos robôs após um certo período de movimentação na ausência de obstáculos é muito próximo da disposição inicial ao início da subtarefa de movimentação coordenada. A medida que os obôs se deslocam, existe um erro cumulativo na posição dos robôs filhos em relação a seus pais. Este erro é principalmente causado por mudanças de direção dos robôs na presença de obstáculos. Não sendo uma formação rigorosamente fixa, os robôs possuem a flexibilidade para ajustar suas posições durante seu deslocamento. Contudo, para restaurar o padrão do início do deslocamento, é necessário que o grupo continue a movimentação sem alterações de direção do inicializador, permitindo que os robôs reorganizem suas posições. Em outras palavras, a formação inicial é restaurada quando o grupo de robôs navega na ausência de obstáculos.

A subtarefa de movimentação mostrou-se muito útil por ser capaz de restaurar a formação após uma perturbação ocasionada por obstáculos. A formação completa do

enxame não é conhecida ou registrada nos robôs. De fato, a informação da disposição espacial encontra-se distribuída pelos robôs do enxame, cada um conhecendo apenas sua posição em relação a seu robô pai.

5.6 Simulações de Redimensionamento

Esta seção apresenta a avaliação experimental do redimensionamento do enxame. O objetivo dos testes é estudar os comportamentos de agregação e dispersão dos enxames de robôs[6]. Utilizou-se um modelo baseado no robô *Kilobot*, alterado de forma a suportar a detecção de vizinhos descrita na Seção 4.5.1. Esta modificação permite aos robôs se movimentarem seguindo as regras de *flocking*.

5.6.1 Configurações da Simulação

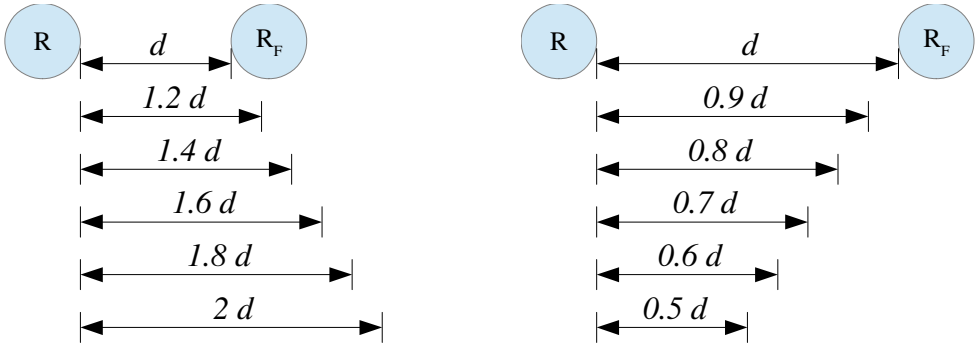
As simulações realizadas estão divididas em dois conjuntos, compostos por testes de agregação e dispersão. Em cada teste, foi feita a medição do tempo necessário para a execução do redimensionamento. Conforme o tipo de teste, este foi chamado de chamado de *Tempo de Agregação* (τ_G) ou *Tempo de Dispersão* (τ_D). As simulações foram executadas considerando diferentes quantidades de robôs, de forma a ser possível verificar a relação entre o tempo de simulação e o tamanho do enxame.

O mesmo algoritmo local é empregado tanto para a agregação quanto para a dispersão. O que diferencia os dois comportamentos são os pesos associados com as regras de *flocking*, w, e o fator de redimensionamento E. Os valores de E usados nas simulações são ilustrados pela Figura 5.16. Nos testes de agregação, tem-se $0,5 < E < 1$. Assim, a distância final entre pais e filhos é uma fração da distância inicial, sendo a menor distância adotada a metade da inicial. Por outro lado, nos testes de dispersão, tem-se $1 < E < 2$. Os robôs afastam-se de forma que a distância final seja maior que a inicial, com a maior distância adotada sendo o dobro da inicial.

Os valores de w foram ajustados de formas distintas para a agregação e para a dispersão. O uso de diferentes valores resulta em uma maior ou menor contribuição da velocidade calculada por cada regra, o que resulta em diferentes comportamentos do enxame. A Tabela 5.3 apresenta os valores empregados nas simulações. Tais valores foram escolhidos de forma que o enxame fosse capaz de executar, com os mesmos parâmetros, todas as simulações de agregação e dispersão. Vale notar que foi possível realizar todos os testes de dispersão empregando apenas a regra de separação. De fato, esta é a regra responsável pelo afastamento mútuo entre os robôs. Para a agregação, contudo, foi preciso empregar as quatro regras com um conjunto de valores bem específico. O peso atribuído à regra de coesão é superior ao

[6] `https://www.youtube.com/playlist?list=PLaeoC3b2KX62s9owqUcIcKSnjMQxzzJRz`

da separação, uma vez que os robôs precisam se aproximar. Ainda assim, a separação não é nula. Isto mostrou-se necessário devido às diferentes velocidades apresentadas por robôs em simulações considerando apenas a regra de coesão. A componente de velocidade atribuída à separação equilibra a velocidade final dos robôs, fazendo com que a agregação ocorra de forma mais uniforme. Pela Equação 4.10, os valores usados para w resultam em velocidades acima de v_{max}. Contudo, isto é necessário para acelerar a movimentação dos robôs, que em grande parte das simulações ocorre com velocidades muito abaixo de v_{max}. Caso as velocidades calculadas ultrapassem a máxima, ambas são multiplicadas por um fator normalizador, de modo que a maior velocidade, v_{cw} ou v_{ccw}, seja v_{max}.



(a) Agregação. (b) Dispersão.

Figura 5.16: Variação de distância considerando diferentes valores de E.

Tabela 5.3: Pesos usados nas simulações de redimensionamento.

Regra	Obstáculos	Coesão	Separação	Pai
Peso	w_1	w_2	w_3	w_4
Agregação	1	3	0,5	3
Dispersão	0	0	3	0

A disposição espacial dos robôs segue a organização usada nos testes de alinhamento apresentados na Seção 5.4.2. O tamanho do enxame varia de 2 a 25 robôs, conforme ilustrado previamente na Figura 5.13. As simulações permitem a avaliação do tempo de simulação em relação ao tamanho do enxame e ao caminho crítico ρ_C. A detecção na subtarefa inicial de recrutamento (que define a relação entre robôs pais e filhos) restringe a vizinhança a no máximo 4 robôs. Nos testes de agregação esta restrição não é necessária, pois os robôs encontram-se inicialmente afastados de forma que apenas 4 vizinhos possam ser detectados por cada robô. O número de robôs em cada vizinhança, contudo, aumenta a medida que o enxame se agrega. Nos testes de dispersão, os robôs encontram-se inicialmente próximos uns dos outros, permitindo vizinhanças com mais de 4 robôs. Com o afastamento mútuo dos robôs,

as vizinhanças passam a ter no máximo 4 robôs ao término da dispersão, sendo estes os mesmos robôs que constituem a vizinhança detectada ao início da tarefa.

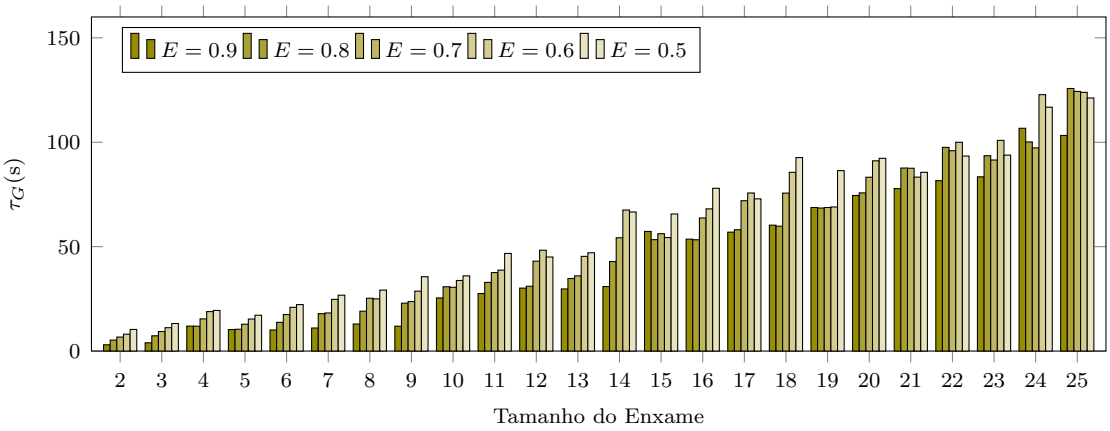
5.6.2 Resultados do Redimensionamento

Esta seção expõe os resultados obtidos nas simulações de agregação e dispersão. Os valores de τ_R referentes à subtarefa inicial de recrutamento são omitidos, pois apresentam o mesmo comportamento exibido nos experimentos de alinhamento apresentados na Seção 5.4.2. Isto se deve ao fato dos conjuntos de testes empregarem as mesmas disposições iniciais dos experimentos de alinhamento, sendo este o único fator que afeta o tempo de recrutamento.

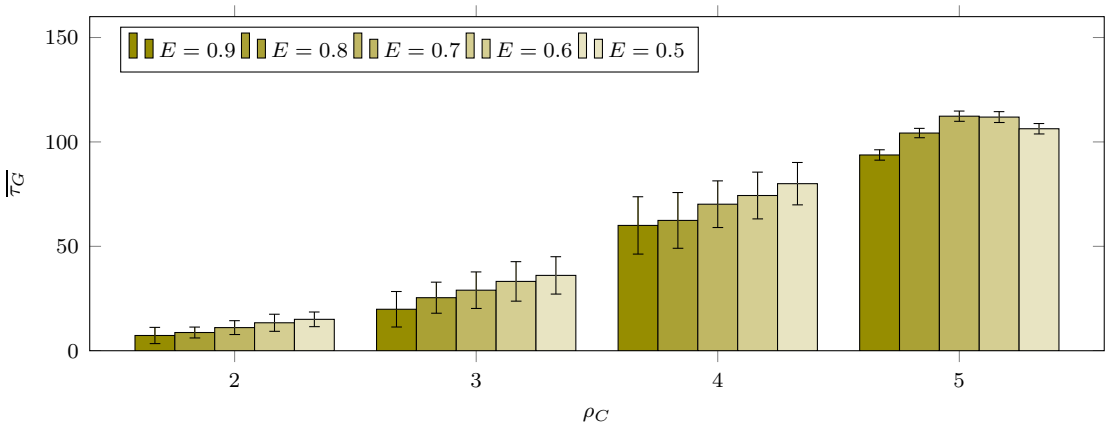
A Figura 5.17 mostra o tempo de simulação obtido na simulações de agregação do enxame. Os valores de τ_G são ordenados em função do tamanho do enxame na Figura 5.17(a). Em experimentos com mesmo tamanho do enxame, as barras indicam os valores obtidos de τ_G em simulações com diferentes valores de E. De modo geral, nota-se uma tendencia de crescimento linear de τ_G em função do tamanho do enxame. Isto difere do ocorrido nas simulações de alinhamento, onde os tempos de execução mostraram-se muito dependentes de ρ_C. Isto se deve ao movimento dos robôs. Enquanto no alinhamento o movimento é constituído puramente de rotações, na agregação os robôs efetivamente se deslocam uns em relação aos outros. O aumento do número de robôs afeta a quantidade de indivíduos que participam do *flocking*, o que consequentemente afeta a velocidade individual de cada robô. Além disso, com a aproximação mútua, mais robôs são detectados em cada vizinhança. Também é importante ressaltar a variação de τ_G em função de E em testes com mesmo tamanho de enxame. É esperado que valores mais próximos de $E = 1$ resultem em experimentos mais rápidos, pois o redimensionamento do enxame será menor, o que requer pouca movimentação dos robôs. Este comportamento é observado em grande parte dos experimentos realizados. Contudo, para os casos com 12, 14, 15, 17, 21, 22, 23, 24 e 25 robôs, foram observados valores que não obedeceram à tal expectativa. Isto pode ser explicado pela não-uniformidade nas velocidades dos robôs.

A Figura 5.17(b) mostra o *Tempo Médio de Agregação* ($\overline{\tau_G}$), constituído pela média dos valores de tempo de simulação obtidos em simulações com mesmos E e ρ_C. A variação dos valores obtidos de τ_G é notável em $\rho_C = 3$ e $\rho_C = 4$.

Os testes de dispersão têm seus resultados representados pela Figura 5.18. Embora os valores de τ_D apresentados na Figura 5.18(a) aumentem com o tamanho do enxame, este crescimento se dá de forma diferente do ocorrido na agregação. Nota-se que τ_D aumenta com o valor de E em todos os tamanhos de enxame usados. Além disso, esta variação é mais acentuada que a variação de τ_G nos testes de agregação, conforme mostra a Figura 5.17(a).



(a) Tempo de Agregação.

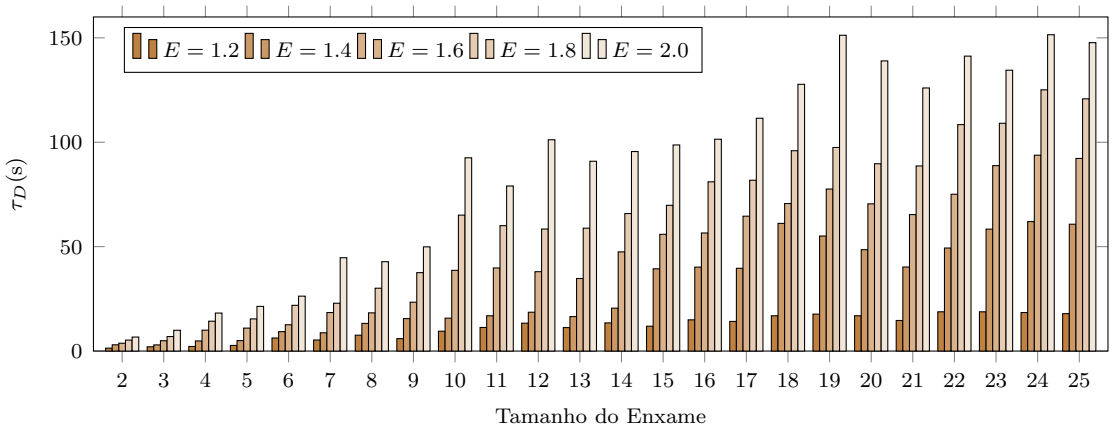


(b) Tempo Médio de Agregação.

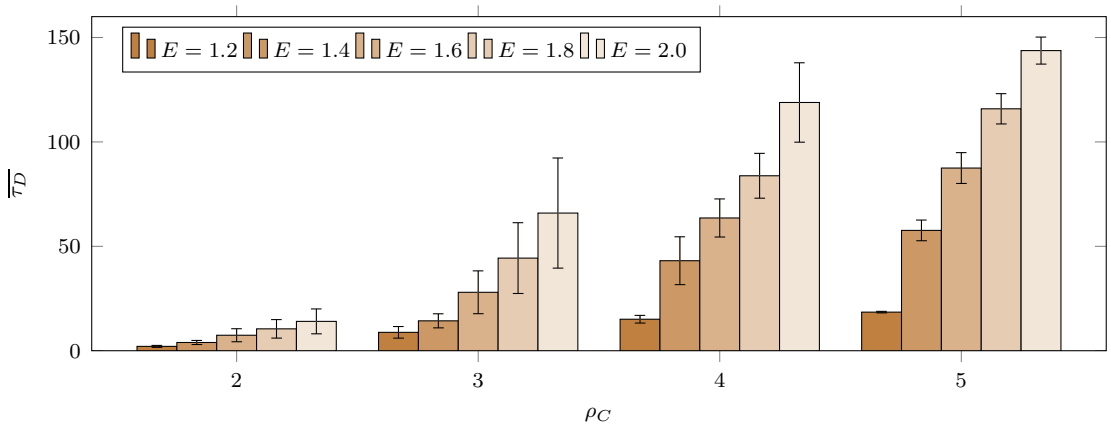
Figura 5.17: Resultados obtidos nas simulações de agregação.

Mais uma vez, o tamanho do caminho crítico não mostrou-se tão impactante na dispersão quanto observado nos testes de alinhamento. O *Tempo Médio de Dispersão* ($\overline{\tau_D}$) mostrado na Figura 5.18(b) apresenta grande variância, principalmente com $\rho_C = 3$ e $\rho_C = 4$. Isto indica a oscilação nos valores obtidos de τ_D observada na Figura 5.18(a). O gráfico de $\overline{\tau_D}$ realça também a variação do tempo de dispersão em função de E. Os valores de $\overline{\tau_D}$ com $E = 2$ para um determinado ρ_C são maiores que os valores com $E = 1.2$ para o ρ_C imediatamente superior. Isto não ocorreu nos testes de agregação, onde os valores de $\overline{\tau_G}$ para um determinado ρ_C são inferiores a todos os valores de um ρ_C imediatamente maior, independentemente de E.

O comportamento mais relevante nesta simulação de dispersão diz respeito à oscilação dos valores obtidos de τ_D. Isto é notável na Figura 5.18(a) para o caso com $E = 2$, principalmente com 10 e 21 robôs. No primeiro caso, τ_D aumenta de foma mais acentuada quando o enxame varia de 9 para 10 robôs do que de 8 para 9 robôs. No segundo, o valor de τ_D diminui com o aumento do enxame de 19 para 20 e de 20 para 21 robôs. Em ambos os casos, a variação se deve ao mesmo motivo: a simetria



(a) Tempo de Dispersão.



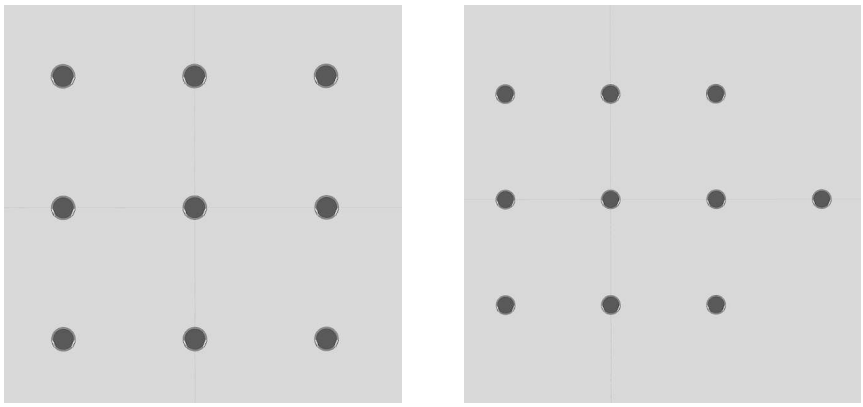
(b) Tempo Médio de Dispersão.

Figura 5.18: Resultados obtidos nas simulações de dispersão.

do enxame, ilustrada pela Figura 5.19. Com 9 robôs o enxame é simétrico, sendo composto pelo inicializador e 8 vizinhos, onde os 4 robôs filhos possuem também 1 filho cada. Com 10 robôs, um dos filhos do inicializados passará a ter 2 filhos. A presença deste robô a mais resulta em um desequilíbrio na ação da força de repulsão que rege a dispersão. Isto faz com que a condição de redimensionamento necessite de mais tempo para ser alcançada, quando em comparação com o caso com 9 robôs. Na mudança de 20 para 21 robôs ocorre o inverso: o aumento na quantidade de robôs é compensado com a simetria do enxame, o que reduz o tempo necessário para conclusão da dispersão.

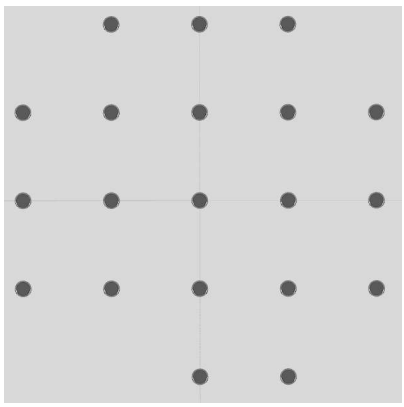
5.7 Considerações Finais do Capítulo

O recrutamento e o alinhamento de robôs fazem uso direto da propagação de informação. Para ambas as subtarefas, foram apresentados experimentos relacionando o tempo de simulação com características do enxame, como a quantidade de robôs

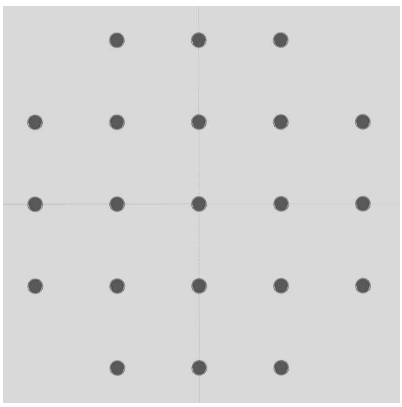


(a) 9 robôs.

(b) 10 robôs.



(c) 20 robôs.



(d) 21 robôs.

Figura 5.19: Disposições simétricas (9 e 21 robôs) ou não-simétricas (10 e 20 robôs) em torno do inicializador central.

e o caminho crítico de propagação da mensagem. Já a subtarefa de movimentação coordenada é baseada em uma combinação das regras de seguidor de líder e desvio de obstáculos. A simulação de tais comportamentos mostra a possibilidade de uso de ondas de mensagens tanto como base para implementação de tarefas específicas, como recrutamento e alinhamento, mas também para o controle da execução em sequência de subtarefas.

Simulações mostraram, tanto para a agregação quanto para a dispersão, a relação entre o tamanho do enxame e o fator de redimensionamento com o tempo de simulação. Os resultados obtidos mostram que outros fatores influenciam os resultados, tais como a disposição espacial dos robôs. O mais importante é o fato do redimensionamento usar a onda de mensagens para a verificação do término da tarefa. Isto permite que outras subtarefas possam ser iniciadas pelo enxame após o término do redimensionamento.

O capítulo seguinte conclui esta tese, apresentando os principais objetivos alcançados, as dificuldades encontradas e pontos que poderiam ser explorados em estudos posteriores.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho teve por objetivo estudar a possibilidade de implementação de tarefas baseadas na propagação de mensagens em enxames de robôs conectados, isto é, robôs capazes de interagir por meio de um sistema de comunicação sem fio. O presente capítulo conclui este trabalho, apresentando uma revisão dos objetivos alcançados, as principais dificuldades observadas neste estudo e os possíveis trabalhos futuros desta pesquisa.

6.1 Conclusões

Nesta tese, um algoritmo de ondas de mensagens foi idealizado, desenvolvido e executado em um enxame simulado de robôs. Este algoritmo serviu de base geral para a para a concepção de diferentes tipos de tarefas coletivas. O principal requisito para a realização desta operação é a presença de um sistema de comunicação por mensagens. Um robô que constitui o enxame pode enviar um sinal para outros robôs, desde que estes se encontrem dentro de uma determinada área de comunicação. A relação de vizinhança entre dois robôs deve ser recíproca: se um robô envia mensagens para um determinado vizinho, então este deve obrigatoriamente ser capaz de receber mensagens do mesmo vizinho. A capacidade de comunicação entre robôs vizinhos configura uma topologia de comunicação no enxame.

Diferentes tarefas foram executadas com o objetivo de validar a estratégia proposta. A propagação da onda de mensagens pelo é interpretada como uma tarefa de recrutamento. Um robô inicializador começa a transmissão de mensagens de propagação, que por sua vez ativa o envio pelos robôs vizinhos. A relação de vizinhança permite que todo o enxame seja inicializado em decorrência do recebimento de mensagens. A propagação de mensagens no recrutamento define a relação entre robôs pais e robôs filhos. Mensagens de realimentação são transmitidas dos filhos para os pais, de forma que ao alcançarem o robô inicializador, este pode afirmar que os demais membros do enxame participaram do recrutamento. Este fato é uma

das principais contribuições deste trabalho, e é o que torna o recrutamento tão importante. Em tarefas baseadas no recrutamento (ou que pelo menos, façam uso em algum momento da estratégia de mensagens de propagação e realimentação), pelo menos um dos robôs tem o conhecimento do término global da tarefa. Isto se deve à relação de causalidade na propagação de mensagens, uma vez que as mensagens de realimentação são enviadas apenas após o término local da tarefa em questão. Tal característica abre caminho para que os próprios membros do enxame encerrem sua operação de forma coordenada, ou mesmo, que realizem transições para tarefas seguintes, não havendo necessidade de um observador global externo ao enxame.

As ondas de mensagens foram empregadas também na tarefa de alinhamento do enxame. Nesta tarefa, todos os robôs devem apontar para uma mesma direção. Os robôs não possuem uma referência global de direcionamento, e usam a posição dos vizinhos como orientação. A emergência do alinhamento ocorre de forma hierárquica seguindo a relação entre robôs pais e filhos a partir do inicializador. Para que o alinhamento ocorra, é necessário que o recrutamento tenha sido previamente executado, de forma que os robôs conheçam a relação entre pais e filhos. Quando os robôs em um percurso desde o inicializador até um robô sem filhos encontram-se todos com um mesmo direcionamento, então mensagens de realimentação são enviadas dos filhos para os pais. Quando o inicializador recebe a realimentação de todos os filhos, então o alinhamento está concluído. Este processo segue a mesma estruturação do recrutamento, com uma propagação inicial de mensagens intercalada com a tarefa em si (as rotações para ajuste de direcionamento), terminando pelo envio de mensagens dos filhos para os pais.

A dependência causal na onda de mensagens permite o encadeamento de tarefas. Neste caso, as tarefas em sequência são chamadas de subtarefas. Isto é útil, pois cada subtarefa ocorre de forma independente, não havendo interferência entre subtarefas. Isto pode ser verificado na tarefa de navegação coletiva, que é composta por três subtarefas executadas em sequência. O alinhamento só inicia após o término do recrutamento, enquanto que a movimentação só começa após a conclusão do alinhamento. A subtarefa de movimentação empregada não faz uso direto da onda de mensagens, isto é, das mensagens de propagação e realimentação. O movimento ocorre pelo ajuste das distâncias e direcionamento de cada robô em resposta à movimentação de seu pai. Contudo, é possível acrescentar uma onda à subtarefa de movimentação para que esta seja interrompida de alguma forma, como por exemplo, a identificação de algum alvo de interesse. A forma com que a movimentação foi implementada apresentou ainda a característica útil de manutenção da formação do enxame na ausência de perturbações. Cada robô sabe apenas a posição relativa de seu respectivo pai, não necessitando conhecer a disposição geral do enxame. O estudo da estratégia de ondas de mensagens possibilitou a publicação de cinco

trabalhos acadêmicos [14–18].

O redimensionamento foi a última classe de tarefas abordada neste trabalho, compreendendo a agregação e a dispersão do enxame. Assim como o alinhamento, o redimensionamento também necessita re uma etapa inicial de recrutamento para que a relação entre robôs pais e filhos. A movimentação dos robôs nesta tarefa é coordenada por regras de *flocking*, que definem a atração e repulsão em relação à vizinhança. A onda de mensagens é usada com base no sistema de comunicação do enxame, enquanto que a movimentação baseada em *flocking* usa o sistema de detecção. A interação entre os dois sistemas permitiu o controle do grau de redimensionamento.

A implementação de tarefas com características distintas, mas baseadas no algoritmo de ondas de mensagens, comprova que a estratégia proposta é geral. É possível concluir que esta abordagem pode ser empregada em outros tipos de tarefas, em operações que requerem a propagação de informação, ou mesmo em situações onde é necessário definir o início e o fim de uma ação coletiva.

6.2 Trabalhos Futuros

O principal desafio encontrado neste trabalho foi a adaptação de tarefas à estratégia proposta. Embora muitas aplicações tenham sido descritas no Capítulo 2, apenas algumas foram executadas. Dentre as tarefas implementadas, exitem as relacionadas com a organização espacial (redimensionamento e alinhamento), com a navegação (movimentação coordenada) e com a tomada de decisão (recrutamento). Segundo Brambilla *et al.* [4], estas são consideradas as três principais classes de tarefas estudadas em robótica de enxame. Desta forma, é de se esperar que outras tarefas pertencentes a estas três classes também possam ser adaptadas à estratégia de ondas de mensagens. Contudo, isto não é algo simples. Não há na literatura consenso sobre vários aspectos referentes às tarefas de enxame. Por exemplo, não há especificações únicas sobre o funcionamento de tarefas, cabendo ao pesquisador interpretar como cada tarefa deve ser realizada a partir de uma descrição em um alto nível de abstração. Além disso, não há definições sobre quais tarefas (dentre as inúmeras encontradas na literatura) são as mais relevantes de serem estudadas. Uma exceção é a tarefa de forrageamento, que devido a sua complexidade e o potencial uso em aplicações de busca, é considerada uma tarefa de referência. Um segundo desafio, ainda relacionado com as aplicações de enxame, é a composição das tarefas por subtarefas. Este trabalho expôs a possibilidade de encadeamento de subtarefas com base na propagação de ondas de mensagens. Neste caso, mais uma vez fica a cargo do pesquisador entender como uma determinada tarefa pode ser descrita como uma sucessão de operações simples.

Outra dificuldade observada neste estudo de robótica de enxame é a relação entre os recursos de um determinado modelo de robô móvel e as operações que este pode realizar. Por exemplo, o robô *Kilobot* foi escolhido como plataforma experimental para este trabalho devido a seu sistema de comunicação, baseado justamente em um esquema de troca de mensagens. Contudo, tal robô possui limitações no que diz respeito à movimentação e detecção. Esta é uma característica observada na literatura, com robôs modulares comerciais sendo usados apenas em tarefas adequadas a sua arquitetura. Este tipo de limitação explica por que certos trabalhos, como o *Swarm-bots* [45], incluem também o projeto físico e simulado dos robôs, de forma que estes atendam as necessidades das tarefas de interesse.

No presente trabalho, o uso do ambiente de simulação *V-REP* foi adotado devido a sua capacidade de modificação dos modelos de robôs. Desta forma, foi possível realizar modificações no modelo simulado do *Kilobot*, acrescentando a capacidade de detecção de vizinhança para a realização de tarefas além do recrutamento. Este fato também levanta a discussão sobre a comparação entre estudos realizados envolvendo apenas simulações e estudos com robôs físicos. Embora muitos fatores físicos possam ser incluídos em um ambiente simulado, sempre haverá o chamado *reality gap* [126]. Por outro lado, os atuais modelos comerciais de robôs móveis são indicados para uso apenas em ambientes controlados. Este fato, de certa forma, também representa um *reality gap* em relação a eventuais usos de robótica de enxame em aplicações do mundo real. De forma geral, os estudos de robótica de enxame apresentam vários níveis de implementação: simulações pontuais de agentes, experimentos em ambientes físicos simulados e o uso em robôs reais. Uma determinada implementação está associada com o nível de abstração do estudo em questão. Trabalhos focados no desempenho de tarefas necessitam de implementações mais realísticas para validar a robustez do controlador. Trabalhos focados em metodologias de desenvolvimento, ou que requerem uma análise macroscópica do enxame, não necessitam do detalhamento dos robôs reais, e geralmente fazem uso de modelos de múltiplos agentes. Neste trabalho, a estratégia proposta apresenta-se em um nível intermediário, com a onda de mensagens sendo um conceito macroscópico do enxame, mas as tarefas sendo algo mais próximo da realidade.

Uma possível extensão deste trabalho é o estudo mais aprofundado das tarefas já implementadas. Por exemplo, pode-se verificar o impacto no tempo de execução causado por fatores não listados, como o número de robôs vizinhos. Nos testes de redimensionamento os robôs foram dispostos espacialmente em uma malha, de forma que cada robô tivesse no máximo 4 vizinhos. Contudo, verificou-se que a tarefa pode ser executada com outras disposições. É de interesse verificar o efeito do uso de outras topologias regulares, como com 3 ou 6 vizinhos. Também é necessário um maior estudo na tarefa de navegação, principalmente no que diz respeito à ma-

nutenção da formação dos robôs. Este foi um comportamento observado durante a movimentação do enxame, mas não chegou-se a analisar a forma com que este ocorre. Uma estudo possível é a avaliação do erro nas posições relativas dos robôs, isto é, a variação entre a posição esperada e a posição real durante o movimento.

Um segundo direcionamento deste trabalho é a modificação das tarefas já implementadas e o uso da estratégia em outras tarefas. Uma limitação da atual implementação do esquema de onda de mensagens é a incapacidade de lidar com falhas. Embora os robôs possam mover-se, evitando assim a fragmentação da topologia de comunicação, não há segurança quanto a eventuais falhas no funcionamento de algum dos robôs. Talvez uma possível solução para esta limitação seja a reexecução da tarefa de recrutamento, fazendo com que a relação entre robôs pais e filhos seja reestabelecida ignorando a presença de robôs defeituosos. Outra vertente deste estudo é a implementação de outros tipos de tarefas usando esta mesma estratégia. Uma possível aplicação é o forrageamento coletivo, onde os robôs precisam encontrar e transportar algum objeto de interesse. A descrição desta tarefa mostra operações sendo realizadas em sequência, o que é adequado à estratégia proposta.

Por fim, outro prosseguimento deste trabalho é o emprego da onda de mensagens em outras plataformas de enxame. O simulador *V-REP* possui uma biblioteca contendo inúmeros modelos de robôs móveis, sendo muitos destes baseados em robôs reais. A simulação de tarefas em outros modelos de robôs não apenas mostraria a abrangência da metodologia proposta, mas também possibilitaria a realização de outros tipos de tarefas por robôs com diferentes características de *hardware*. Um exemplo são os robôs com capacidade de manipulação de objetos.

Referências Bibliográficas

[1] BENI, G. "The concept of cellular robotic system". In: *Proceedings of the 1988 IEEE International Symposium on Intelligent Control*, pp. 57–62. IEEE, 1988.

[2] BONABEAU, E., DORIGO, M., THERAULAZ, G. *Swarm intelligence: from natural to artificial systems*. New York, USA, Oxford University Press, Inc., 1999.

[3] BAYINDIR, L. "A review of swarm robotics tasks", *Neurocomputing*, v. 172, pp. 292–321, 2016.

[4] BRAMBILLA, M., FERRANTE, E., BIRATTARI, M., et al. "Swarm Robotics: a review from the swarm engineering perspective", *Swarm Intelligence*, v. 7, n. 1, pp. 1–41, 2013.

[5] DONCIEUX, S., BREDECHE, N., MOURET, J.-B., et al. "Evolutionary robotics: what, why, and where to", *Frontiers in Robotics and AI*, v. 2, n. 4, 2015.

[6] DUARTE, M., COSTA, V., GOMES, J., et al. "Evolution of collective behaviors for a real swarm of aquatic surface robots", *PloS one*, v. 11, n. 3, pp. e0151834, 2016.

[7] ZIPARO, V., IOCCHI, L., LIMA, P., et al. "Petri net plans", *Autonomous Agents and Multi-Agent Systems*, v. 23, n. 3, pp. 344–383, 2011.

[8] MERMOUD, G., UPADHYAY, U., EVANS, W., et al. "Top-down vs. bottom-up model-based methodologies for distributed control: a comparative experimental study". In: *Experimental Robotics*, pp. 615–629. Springer, 2014.

[9] SEGALL, A. "Distributed network protocols", *IEEE Transactions on Information Theory*, v. 29, n. 1, pp. 23–35, 1983.

[10] ROHMER, E., SINGH, S., FREESE, M. "V-REP: a versatile and scalable robot simulation framework". In: *Proceedings of the 2013 IEEE/RSJ In-*

ternational Conference on Intelligent Robots and Systems, pp. 1321–1326. IEEE, 2013.

[11] RUBENSTEIN, M., AHLER, C., HOFF, N., et al. "Kilobot: a low cost robot with scalable operations designed for collective behaviors", *Robotics and Autonomous Systems*, v. 62, n. 7, pp. 966–975, 2014.

[12] SILVA JUNIOR, L. "Simulações Wave-Swarm". 2017. Disponível em `https://www.youtube.com/channel/UCjMHBBznDeXfL6kB-UcQNFg`. Acesso em 01 de Maio de 2017.

[13] SILVA JUNIOR, L., NEDJAH, N. "Review of methodologies and tasks in swarm robotics towards standardization". 2017. Submetido em 12/2016. Revisado em 05/2017. Aguardando reposta.

[14] SILVA JUNIOR, L., NEDJAH, N. "Wave algorithm for recruitment in swarm robotics". In: *Proceedings of the 2015 International Conference of Computational Science and Its Applications*, Springer, pp. 3–13, Cham, Switzerland, 2015.

[15] SILVA JUNIOR, L., NEDJAH, N. "Distributed strategy for robots recruitment in swarm-based systems", *International Journal of Bio-Inspired Computation*, v. 8, n. 2, pp. 99–108, 2016.

[16] SILVA JUNIOR, L., NEDJAH, N. "Efficient strategy for collective navigation control in swarm robotics", *Procedia Computer Science*, v. 80, pp. 814–823, 2016. Proceedings of the 2016 International Conference on Computational Science.

[17] SILVA JUNIOR, L., NEDJAH, N. "Wave algorithm applied to collective navigation of robotic swarms", *Applied Soft Computing*, v. 57, pp. 698–707, 2017.

[18] SILVA JUNIOR, L., NEDJAH, N. "Distributed algorithms for recruitment and coordinated motion in swarm robotic systems". In: *Artificial Intelligence: Concepts, Methodologies, Tools, and Applications*, IGI Global, pp. 671–693, Hershey, USA, 2017.

[19] GUZZONI, D., CHEYER, A., JULIA, L., et al. "Many robots make short work: report of the SRI International mobile robot team", *AI Magazine*, v. 18, n. 1, pp. 55–64, 1997.

[20] DENEUBOURG, J.-L., THERAULAZ, G., BECKERS, R., et al. "Swarm made architectures". In: *1st European Conference on Artificial Life*, pp. 123–133. MIT Press, 1992.

[21] THERAULAZ, G., DENEUBOURG, J.-L. "Swarm Intelligence in social insects and the emergence of cultural swarm patterns", *The Ethological roots of Culture*, pp. 1–19, 1994.

[22] DORIGO, M., MANIEZZO, V., COLORNI, A. "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics*, v. 26, n. 1, pp. 29–41, 1996.

[23] KENNEDY, J., EBERHART, R. "Particle swarm optimization". In: *Proceedings of the 1995 IEEE International Conference on Neural Networks*, v. 4, pp. 1942–1948, 1995.

[24] YANG, X.-S. *Nature-inspired metaheuristic algorithms*. Cambridge, UK, Luniver Press, 2010.

[25] ŞAHIN, E. "Swarm robotics: from sources of inspiration to domains of application". In: *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*, pp. 10–20, Berlin, Germany, Springer, 2005.

[26] WINFIELD, A., NEMBRINI, J. "Safety in numbers: fault-tolerance in robot swarms", *International Journal of Modelling, Identification and Control*, v. 1, n. 1, pp. 30–37, 2006.

[27] RUBENSTEIN, M., CORNEJO, A., NAGPAL, R. "Programmable self-assembly in a thousand-robot swarm", *Science*, v. 345, n. 6198, pp. 795–799, 2014.

[28] WERFEL, J., PETERSEN, K., NAGPAL, R. "Designing collective behavior in a termite-inspired robot construction team", *Science*, v. 343, n. 6172, pp. 754–758, 2014.

[29] FUKUDA, T. *Soft computing for intelligent robotic systems*, v. 21. Berlin, Germany, Springer, 2013.

[30] SCHMICKL, T., HAMANN, H. "BEECLUST: a swarm algorithm derived from honeybees", *Bio-inspired Computing and Communication Networks*, 2011.

[31] MENDONÇA, R., NEDJAH, N., MOURELLE, L. "Efficient distributed algorithm of dynamic task assignment for swarm robotics", *Neurocomputing*, v. 172, pp. 345–355, 2016.

[32] SHAPIRO, J. "Thinking about bacterial populations as multicellular organisms", *Annual Reviews in Microbiology*, v. 52, n. 1, pp. 81–104, 1998.

[33] BONABEAU, E., THERAULAZ, G., DENEUBOURG, J.-L., et al. "Self-organization in social insects", *Trends in Ecology & Evolution*, v. 12, n. 5, pp. 188–193, 1997.

[34] PITCHER, T., MAGURRAN, A., WINFIELD, I. "Fish in larger shoals find food faster", *Behavioral Ecology and Sociobiology*, v. 10, n. 2, pp. 149–151, 1982.

[35] BENI, G. "From swarm intelligence to swarm robotics". In: *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*, pp. 1–9, Berlin, Germany, Springer, 2005.

[36] EIBEN, A., SMITH, J. *Introduction to Evolutionary Computing*. Berlin, Germany, Springer, 2003.

[37] HOLLAND, J. "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence", *Ann Arbor: University of Michigan Press, 1975*, v. 1, 1975.

[38] SALCEDO-SANZ, S. "Modern meta-heuristics based on nonlinear physics processes: a review of models and design procedures", *Physics Reports*, v. 655, pp. 1–70, 2016.

[39] CLERC, M. "TRIBES-un exemple d'optimisation par essaim particulaire sans parametres de contrôle", *Optimisation par Essaim Particulaire (OEP 2003), Paris, France*, v. 64, 2003.

[40] KARABOGA, D., BASTURK, B. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *Journal of Global Optimization*, v. 39, n. 3, pp. 459–471, 2007.

[41] ŁUKASIK, S., ŻAK, S. "Firefly algorithm for continuous constrained optimization tasks". In: *Proceedings of the 2009 International Conference on Computational Collective Intelligence*, pp. 97–106. Springer, 2009.

[42] YANG, X.-S., DEB, S. "Cuckoo search via Lévy flights". In: *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210–214. IEEE, 2009.

[43] CIVICIOGLU, P. "Backtracking search optimization algorithm for numerical optimization problems", *Applied Mathematics and Computation*, v. 219, n. 15, pp. 8121–8144, 2013.

[44] WOLPERT, D. H., MACREADY, W. G. "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, pp. 67–82, 1997.

[45] DORIGO, M., TUCI, E., GROSS, R., et al. "The swarm-bots project". In: *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*, pp. 31–44, Berlin, Germany, Springer, 2005.

[46] DORIGO, M., FLOREANO, D., GAMBARDELLA, L., et al. "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms", *Robotics & Automation Magazine, IEEE*, v. 20, n. 4, pp. 60–71, 2013.

[47] MCLURKIN, J., SMITH, J. "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots". In: *Distributed Autonomous Robotic Systems*, pp. 399–408, Tokyo, Japan, Springer, 2007.

[48] MCLURKIN, J., LYNCH, A. J., RIXNER, S., et al. "A low-cost multi-robot system for research, teaching, and outreach". In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*, pp. 597–609, Berlin, Germany, Springer, 2013.

[49] MCLURKIN, J., MCMULLEN, A., ROBBINS, N., et al. "A robot system design for low-cost multi-robot manipulation". In: *Proceedings of the 2014 IEEE/RSJ International Conference onIntelligent Robots and Systems*, pp. 912–918. IEEE, 2014.

[50] MONDADA, F., FRANZI, E., GUIGNARD, A. "The development of Khepera". In: *Proceedings of the 1999 International Khepera Workshop: Experiments with the Mini-Robot Khepera*, n. LSRO-CONF-2006-060, HNI-Verlagsschriftenreihe, Heinz Nixdorf Institut 64, pp. 7–14, 1999.

[51] SOARES, J., NAVARRO, I., MARTINOLI, A. "The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox". In: *Robot 2015: Second Iberian Robotics Conference*, pp. 767–781. Springer, 2016.

[52] CHALLINGER, J. "Efficient use of robots in the undergraduate curriculum". In: *ACM SIGCSE Bulletin*, v. 37, pp. 436–440. ACM, 2005.

[53] PUGH, J., RAEMY, X., FAVRE, C., et al. "A fast onboard relative positioning module for multirobot systems", *IEEE/ASME Transactions on Mechatronics*, v. 14, n. 2, pp. 151–162, 2009.

[54] MONDADA, F., BONANI, M., RAEMY, X., et al. "The e-puck, a robot designed for education in engineering". In: *Proceedings of the 2009 Conference on Autonomous Robot Systems and Competitions*, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[55] BETTHAUSER, J., BENAVIDES, D., SCHORNICK, J., et al. "WolfBot: a distributed mobile sensing platform for research and education". In: *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, pp. 1–8. IEEE, 2014.

[56] WILSON, S., GAMEROS, R., SHEELY, M., et al. "Pheeno, a versatile swarm robotic research and education platform", *IEEE Robotics and Automation Letters*, v. 1, n. 2, pp. 884–891, 2016.

[57] GERKEY, B., VAUGHAN, R., HOWARD, A., et al. "The player/stage project". 2003. Disponível em `http://playerstage.sourceforge.net`. Acesso em 01 de Maio de 2017.

[58] KOENIG, N., HOWARD, A. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, v. 3, pp. 2149–2154. IEEE, 2004.

[59] MAGNENAT, S., WAIBEL, M., BEYELER, A. "Enki: The fast 2D robot simulator". 2007. Disponível em `http://home.gna.org/enki/`. Acesso em 01 de Maio de 2017.

[60] MICHEL, O. "Webots: professional mobile robot simulation", *International Journal of Advanced Robotic Systems*, v. 1, n. 1, pp. 5, 2004.

[61] PINCIROLI, C., TRIANNI, V., O'GRADY, R., et al. "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems", *Swarm intelligence*, v. 6, n. 4, pp. 271–295, 2012.

[62] CARPIN, S., LEWIS, M., WANG, J., et al. "USARSim: a robot simulator for research and education". In: *Proceedings of the 2007 IEEE International Conference on Robotics and Automation,*, pp. 1400–1405. IEEE, 2007.

[63] KITANO, H., ASADA, M., KUNIYOSHI, Y., et al. "Robocup: the robot world cup initiative". In: *Proceedings of the 1997 International Conference on Autonomous Agents*, pp. 340–347. ACM, 1997.

[64] BREDECHE, N., MONTANIER, J.-M., WEEL, B., et al. "Roborobo! a fast robot simulator for swarm and collective robotics", *arXiv preprint arXiv:1304.2888*, 2013.

[65] GHOMMAM, J., MEHRJERDI, H., SAAD, M. "Leader-follower formation control of nonholonomic robots with fuzzy logic based approach for obstacle avoidance". In: *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2340–2345. IEEE, 2011.

[66] BEZZO, N., MEHTA, A., ONAL, C., et al. "Robot makers: the future of digital rapid design and fabrication of robots", *IEEE Robotics & Automation Magazine*, v. 22, n. 4, pp. 27–36, 2015.

[67] BERMAN, B. "3-D printing: the new industrial revolution", *Business horizons*, v. 55, n. 2, pp. 155–162, 2012.

[68] YU, J., HAN, S. D., TANG, W. N., et al. "A portable, 3D-printing enabled multi-vehicle platform for robotics research and education". In: *Proceedings of the 2017 IEEE International Conference on Robotics and Automation*, pp. 1475–1480, 2017.

[69] PAN, S., GUO, S., SHI, L., et al. "A spherical robot based on all programmable SoC and 3-D printing". In: *Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation*, pp. 150–155, 2014.

[70] BEZZO, N., PARK, J., KING, A., et al. "Demo abstract: ROSLab; a modular programming environment for robotic applications". In: *Proceedings of the 2014 ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 214–214, 2014.

[71] MEHTA, A., BEZZO, N., GEBHARD, P., et al. "A design environment for the rapid specification and fabrication of printable robots". In: *The 14th International Symposium on Experimental Robotics*, pp. 435–449, Cham, Switzerland, Springer, 2016.

[72] BEZZO, N., GEBHARD, P., LEE, I., et al. "Rapid co-design of electromechanical specifications for robotic systems". In: *Proceedings of the 2015 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V009T07A009–V009T07A009. American Society of Mechanical Engineers, 2015.

[73] DUDEK, G., JENKIN, M., MILIOS, E., et al. "A taxonomy for swarm robots". In: *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, v. 1, pp. 441–447, 1993.

[74] CAO, Y. U., FUKUNAGA, A., KAHNG, A., et al. "Cooperative mobile robotics: antecedents and directions". In: *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, v. 1, pp. 226–234, 1995.

[75] IOCCHI, L., NARDI, D., SALERNO, M. "Reactivity and deliberation: a survey on multi-robot systems". In: *Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From RoboCup to Real-World Applications*, pp. 9–32, Berlin, Germany, Springer, 2001.

[76] BAYINDIR, L., ŞAHIN, E. "A review of studies in swarm robotics", *Turkish Journal of Electrical Engineering*, v. 15, n. 2, pp. 115–147, 2007.

[77] DUDEK, G., JENKIN, M., MILIOS, E., et al. "A taxonomy for multi-agent robotics", *Autonomous Robots*, v. 3, n. 4, pp. 375–397, 1996.

[78] CAO, Y. U., FUKUNAGA, A., KAHNG, A. "Cooperative mobile robotics: antecedents and directions", *Autonomous Robots*, v. 4, n. 1, pp. 7–27, 1997.

[79] KUBE, C., BONABEAU, E. "Cooperative transport by ants and robots", *Robotics and Autonomous Systems*, v. 30, n. 1, pp. 85–101, 2000.

[80] SPEARS, W., SPEARS, D., HAMANN, J., et al. "Distributed, physics-based control of swarms of vehicles", *Autonomous Robots*, v. 17, n. 2-3, pp. 137–162, 2004.

[81] GARNIER, S., GAUTRAIS, J., ASADPOUR, M., et al. "Self-organized aggregation triggers collective decision making in a group of cockroach-like robots", *Adaptive Behavior*, v. 17, n. 2, pp. 109–133, 2009.

[82] LUDWIG, L., GINI, M. "Robotic swarm dispersion using wireless intensity signals". In: *Distributed Autonomous Robotic Systems*, pp. 135–144, Tokyo, Japan, Springer, 2006.

[83] UGUR, E., TURGUT, A., ŞAHIN, E. "Dispersion of a swarm of robots based on realistic wireless intensity signals". In: *Proceedings of the 2007 International Symposium on Computer and Information Sciences*, pp. 1–6. IEEE, 2007.

[84] NECSULESCU, P., SCHILLING, K. "Automation of a multiple robot self-organizing multi-hop mobile ad-hoc network (MANET) using signal strength". In: *Proceedings of the 2015 IEEE International Instrumentation and Measurement Technology Conference*, pp. 505–510. IEEE, 2015.

[85] RANJBAR-SAHRAEI, B., WEISS, G., NAKISAEE, A. "A multi-robot coverage approach based on stigmergic communication". In: *Multiagent System Technologies*, pp. 126–138, Berlin, Germany, Springer, 2012.

[86] CALISKANELLI, I., BROECKER, B., TUYLS, K. "Multi-robot coverage: a bee pheromone signalling approach". In: *Artificial Life and Intelligent Agents*, pp. 124–140, Cham, Switzerland, Springer, 2014.

[87] BROECKER, B., CALISKANELLI, I., TUYLS, K., et al. "Social insect-inspired multi-robot coverage". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1775–1776. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[88] HOWARD, A., MATARIĆ, M., SUKHATME, G. "Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem". In: *Distributed Autonomous Robotic Systems*, Springer, pp. 299–308, Tokyo, Japan, 2002.

[89] REIF, J., WANG, H. "Social potential fields: a distributed behavioral control for autonomous robots", *Robotics and Autonomous Systems*, v. 27, n. 3, pp. 171–194, 1999.

[90] CAMAZINE, S., SNEYD, J. "A model of collective nectar source selection by honey bees: self-organization through simple rules", *Journal of Theoretical Biology*, v. 149, n. 4, pp. 547–571, 1991.

[91] BEN-JACOB, E. "Bacterial self-organization: co-enhancement of complexification and adaptability in a dynamic environment", *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, v. 361, n. 1807, pp. 1283–1312, 2003.

[92] SPRINGHOLZ, G., HOLY, V., PINCZOLITS, M., et al. "Self-organized growth of three-dimensional quantum-dot crystals with fcc-like stacking and a tunable lattice constant", *Science*, v. 282, n. 5389, pp. 734–737, 1998.

[93] FLOCCHINI, P., PRENCIPE, G., SANTORO, N., et al. "Arbitrary pattern formation by asynchronous, anonymous, oblivious robots", *Theoretical Computer Science*, v. 407, n. 1, pp. 412–447, 2008.

[94] LEVI, P., KERNBACH, S. *Symbiotic multi-robot organisms: reliability, adaptability, evolution*, v. 7. Berlin, Germany, Springer, 2010.

[95] LIU, W., WINFIELD, A. "Self-assembly in heterogeneous modular robots". In: *Distributed Autonomous Robotic Systems*, pp. 219–232, Berlin, Germany, Springer, 2014.

[96] AULINAS, J., PETILLOT, Y., SALVI, J., et al. "The SLAM problem: a survey". In: *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development*, pp. 363–371, Amsterdam, Netherlands, 2008. IOS Press.

[97] NIETO-GRANDA, C., ROGERS, J., CHRISTENSEN, H. "Coordination strategies for multi-robot exploration and mapping", *The International Journal of Robotics Research*, v. 33, n. 4, pp. 519–533, 2014.

[98] FAIGL, J., KULICH, M. "On benchmarking of frontier-based multi-robot exploration strategies". In: *Proceedings of the 2015 European Conference on Mobile Robots*, pp. 1–8. IEEE, 2015.

[99] RUSSELL, K., SCHADER, M., ANDREA, K., et al. "Swarm robot foraging with wireless sensor motes". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 287–295. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[100] LIEMHETCHARAT, S., YAN, R., TEE, K. P., et al. "Multi-robot item delivery and foraging: two sides of a coin", *Robotics*, v. 4, n. 3, pp. 365–397, 2015.

[101] PITONAKOVA, L., CROWDER, R., BULLOCK, S. "Information flow principles for plasticity in foraging robot swarms", *Swarm Intelligence*, v. 10, n. 1, pp. 33–63, 2016.

[102] WINFIELD, A. "Foraging robots". In: *Encyclopedia of Complexity and Systems Science*, pp. 3682–3700, New York, USA, Springer, 2009.

[103] BALCH, T., ARKIN, R. "Behavior-based formation control for multirobot teams", *IEEE Transactions on Robotics and Automation*, v. 14, n. 6, pp. 926–939, 1998.

[104] NAVARRO, I., MATÍA, F. "A survey of collective movement of mobile robots", *International Journal of Advanced Robotic Systems*, v. 10, n. 73, 2013.

[105] REYNOLDS, C. "Flocks, herds and schools: a distributed behavioral model". In: *ACM SIGGRAPH Computer Graphics*, v. 21, pp. 25–34. ACM, 1987.

[106] TURGUT, A., ÇELIKKANAT, H., GÖKÇE, F., et al. "Self-organized flocking in mobile robot swarms", *Swarm Intelligence*, v. 2, n. 2-4, pp. 97–120, 2008.

[107] ERFIANTO, B., TRILAKSONO, B. "Simulation of swarm robot flocking assisted by explicit communication", *Journal of Unmanned System Technology*, v. 1, n. 2, pp. 49–57, 2013.

[108] YASUDA, T., ADACHI, A., OHKURA, K. "Self-organized flocking of a mobile robot swarm by topological distance-based interactions". In: *Proceedings of the 2014 IEEE/SICE International Symposium on System Integration*, pp. 106–111. IEEE, 2014.

[109] WANG, Z., KUMAR, V. "Object closure and manipulation by multiple cooperating mobile robots". In: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, v. 1, pp. 394–399. IEEE, 2002.

[110] PEREIRA, G., KUMAR, V., SPLETZER, J., et al. "Cooperative transport of planar objects by multiple mobile robots using object closure". In: *Experimental Robotics*, pp. 287–296, Berlin, Germany, Springer, 2003.

[111] CHEN, J., GAUCI, M., LI, W., et al. "Occlusion-based cooperative transport with a swarm of miniature mobile robots", *IEEE Transactions on Robotics*, v. 31, n. 2, pp. 307–321, 2015.

[112] RUBENSTEIN, M., CABRERA, A., WERFEL, J., et al. "Collective transport of complex objects by simple robots: theory and experiments". In: *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 47–54. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[113] VALENTINI, G., HAMANN, H., DORIGO, M. "Efficient decision-making in a self-organizing robot swarm: on the speed versus accuracy trade-off". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1305–1314. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[114] VALENTINI, G., HAMANN, H., DORIGO, M. "Self-organized collective decision-making in a 100-robot swarm". In: *Proceedings of the 2015 AAAI Conference on Artificial Intelligence*, 2015.

[115] KRIEGER, M., BILLETER, J.-B. "The call of duty: self-organised task allocation in a population of up to twelve mobile robots", *Robotics and Autonomous Systems*, v. 30, n. 1, pp. 65–84, 2000.

[116] JONES, C., MATARIĆ, M. "Adaptive division of labor in large-scale minimalist multi-robot systems". In: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, v. 2, pp. 1969–1974, 2003.

[117] CAPRARI, G. "Elisa-3 Robot". 2016. Disponível em `http://www.gctronic.com/doc/index.php/Elisa-3`. Acesso em 01 de Maio de 2017.

[118] NEDJAH, N., MENDONÇA, R., MOURELLE, L. "PSO-based distributed algorithm for dynamic task allocation in a robotic swarm", *Procedia Computer Science*, v. 51, pp. 326–335, 2015. Proceedings of the 2015 International Conference On Computational Science.

[119] RASHID, A., FRASCA, M., ALI, A., et al. "Multi-robot localization and orientation estimation using robotic cluster matching algorithm", *Robotics and Autonomous Systems*, v. 63, pp. 108–121, 2015.

[120] SÁ, A., NEDJAH, N., MOURELLE, L. "Distributed efficient localization in swarm robotic systems using swarm intelligence algorithms", *Neurocomputing*, v. 172, pp. 322–336, 2016.

[121] SÁ, A., NEDJAH, N., MOURELLE, L. "Distributed efficient localization in swarm robotics using Min–Max and Particle Swarm Optimization", *Expert Systems with Applications*, v. 50, pp. 55–65, 2016.

[122] TEL, G. *Introduction to Distributed Algorithms*. Cambridge, UK, Cambridge University Press, 2000.

[123] LYNCH, N. *Distributed Algorithms*. San Francisco, USA, Morgan Kaufmann Publishers Inc., 1996.

[124] BARBOSA, V. *An Introduction to Distributed Algorithms*. Cambridge, USA, MIT Press, 1996.

[125] FINN, S. "Resynch procedures and a fail-safe network protocol", *IEEE Transactions on Communications*, v. 27, n. 6, pp. 840–845, 1979.

[126] MOURET, J.-B., CHATZILYGEROUDIS, K. "Twenty years of reality gap: a few thoughts about simulators in evolutionary robotics". In: *Proceedings of the 2017 Genetic and Evolutionary Computation Conference Companion*, pp. 1121–1124, 2017.

Apêndice A

Resultados das Simulações de Navegação Coletiva

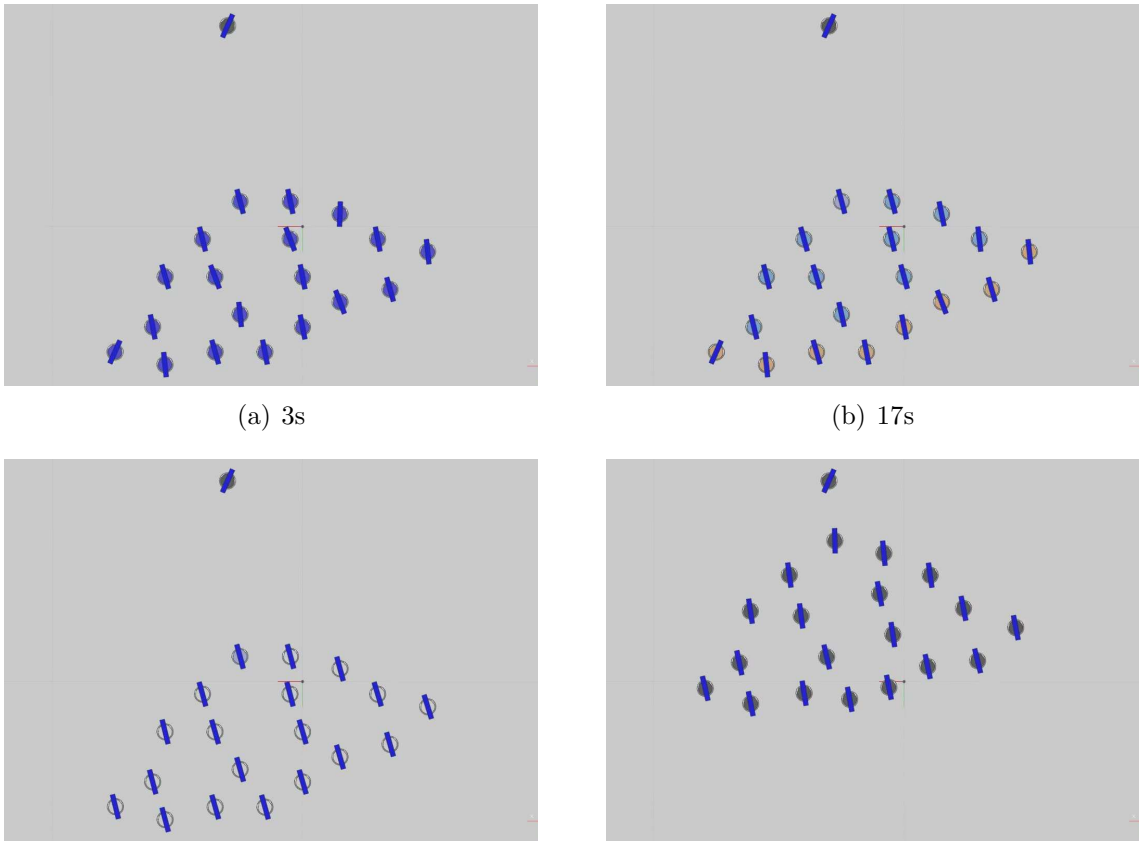
Este apêndice apresenta os resultados de simulação dos experimentos de navegação coletiva. Conforme discutido na Seção 5.5, foram realizadas simulações do enxame operando em 10 cenários distintos, com cada cenário possuindo diferentes características. Um dos objetivos deste conjunto de simulações é explorar a capacidade de realização da subtarefa de movimentação coordenada.

Verificou-se que em todos os experimentos o enxame conseguiu se deslocar pelo ambiente, evitando colisões e mantendo a conectividade entre os robôs. A movimentação, por sua vez, é uma das subtarefas que constitui a tarefa complexa de navegação coletiva. Desta forma, os experimentos também objetivam validar a sequenciação de subtarefas por meio de ondas de mensagens, que é a principal contribuição apresentada por este trabalho.

Em todas as simulações foram selecionados quatro instantes de tempo para exibição. As Figuras A.1 até A.11 são capturas de tela do simulador *V-REP* ilustrando o transcorrer de cada experimento. Os vídeos contendo o registro completo das simulações estão disponíveis para acesso *online* [12].

A.1 Experimento 1

O Experimento 1 exemplifica o movimento retilíneo do grupo de robôs em um ambiente sem obstáculos. Este experimento é composto por um grupo de robôs e por um robô alvo que encontra-se isolado dos demais, conforme exposto na Figura A.1. O inicializador recruta todos os demais robôs exceto o robô isolado, uma vez que este se encontra além de seu raio de comunicação. Os robôs que formam o grupo alinham-se em relação ao inicializador. Após o alinhamento, o inicializador inicia seu deslocamento em linha reta em direção ao robô isolado. Os demais robôs seguem o inicializador, tentando manter a distância medida durante o alinhamento para seus respectivos pais. Quando o robô alvo encontra-se no raio de medição do inicializador, este interrompe seu movimento e propaga uma onda de mensagens, indicando aos demais robôs que o deslocamento foi concluído. Dentre as simulações apresentadas neste apêndice, esta é a única em que o inicializador faz uso de uma onda de mensagens para encerrar a tarefa. Para os demais experimentos, é considerado que os robôs continuam seu deslocamento pelo ambiente. Um vídeo com o registro do Experimento 1 está disponível para acesso *online* [1].



(a) 3s (b) 17s

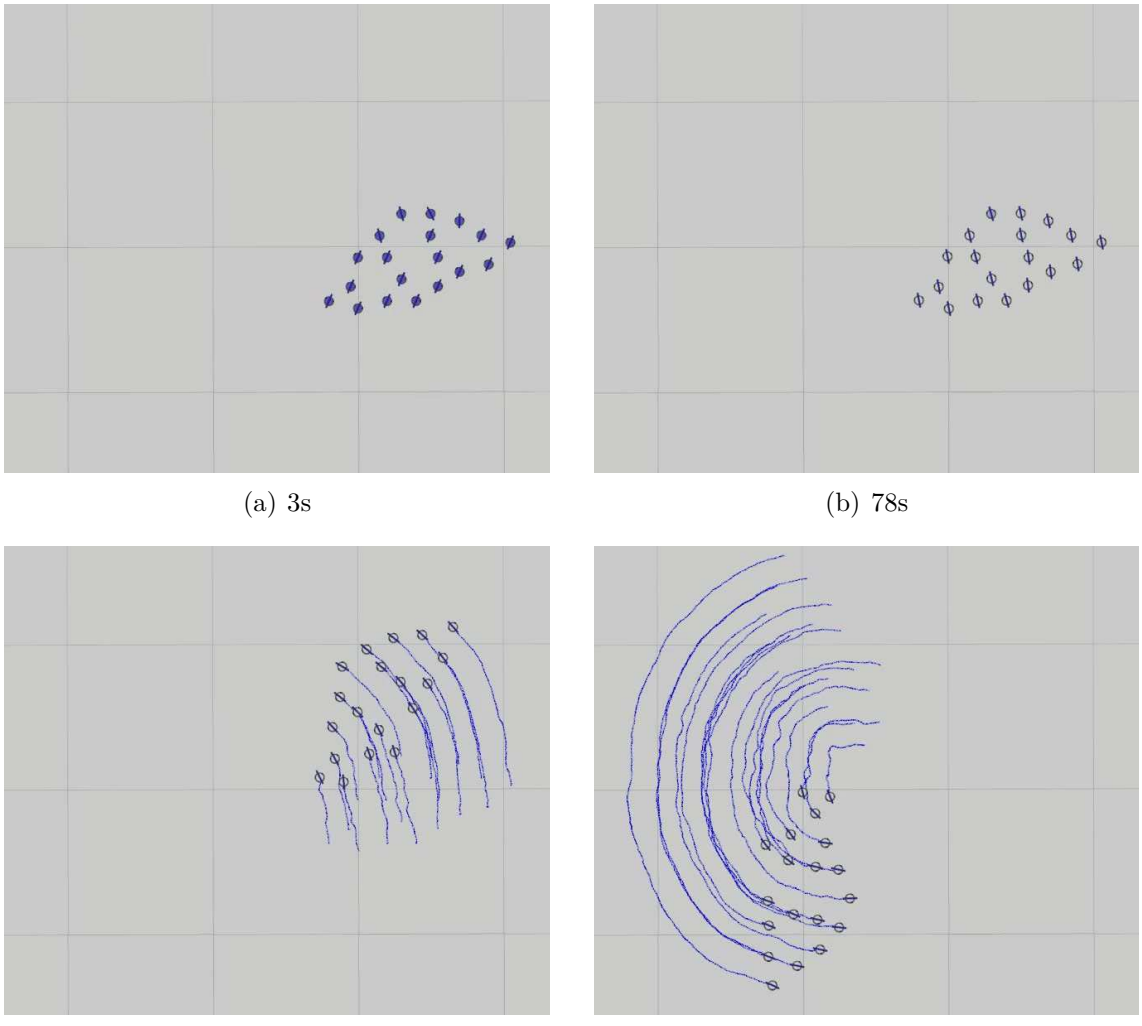
(c) 53s (d) 95s

Figura A.1: Experimento 1 de movimentação em diferentes instantes de tempo.

[1] `https://youtu.be/8gv4lMy_jDA`

A.2 Experimento 2

O Experimento 2 apresenta um enxame descrevendo uma trajetória circular. Na Figura A.2 são exibidos diferentes instantes desta simulação. O robô inicializador movimenta-se com velocidades v_{cw} e v_{ccw} constantes. Foi definido intencionalmente que $v_{ccw} > v_{cw}$, de forma que o inicializador realize um curva no sentido anti-horário. Os demais membros do enxame movimentam-se para compensar o deslocamento do inicializador. Os rastros na Figura A.2 indicam o percurso de cada robô. Neste exemplo, é possível verificar que a velocidade do robôs não é a mesma. Os obôs mais próximos do centro possuem velocidade menor que a dos mais próximos da borda. Um vídeo com o registro do Experimento 2 está disponível para acesso *online* [2].



(a) 3s (b) 78s

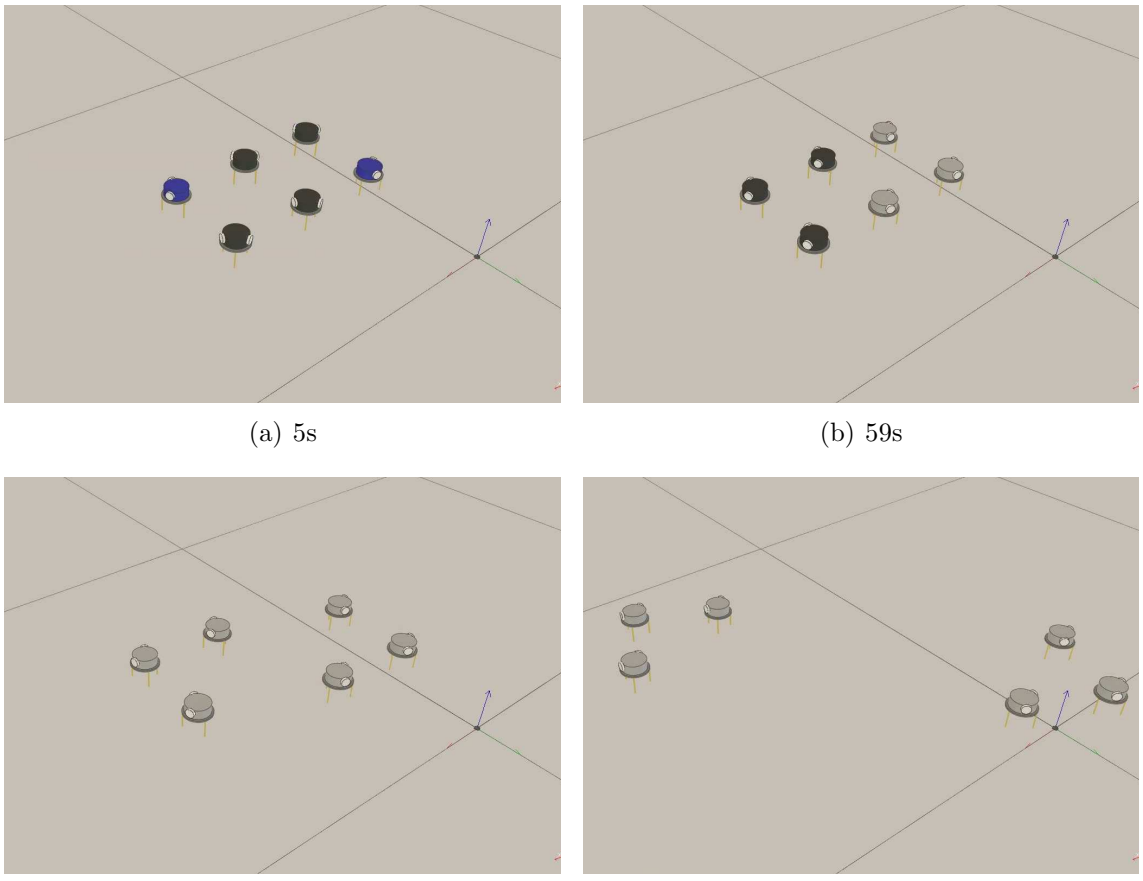
(c) 250s (d) 1000s

Figura A.2: Experimento 2 de movimentação em diferentes instantes de tempo.

[2] `https://youtu.be/8GqfjmAaHI`

A.3 Experimento 3

O Experimento 3 apresenta a movimentação de dois grupos de robôs. Neste exemplo ilustrado pela Figura A.3, o enxame é composto por 6 robôs, sendo 2 inicializadores. A execução simultânea do recrutamento por ambos os inicializadores resulta na formação de 2 grupos. Os inicializadores estão intencionalmente direcionados para lados opostos, de forma que a movimentação resulte na divisão do enxame original. Um vídeo com o registro do Experimento 3 está disponível para acesso *online* [3].



(a) 5s (b) 59s

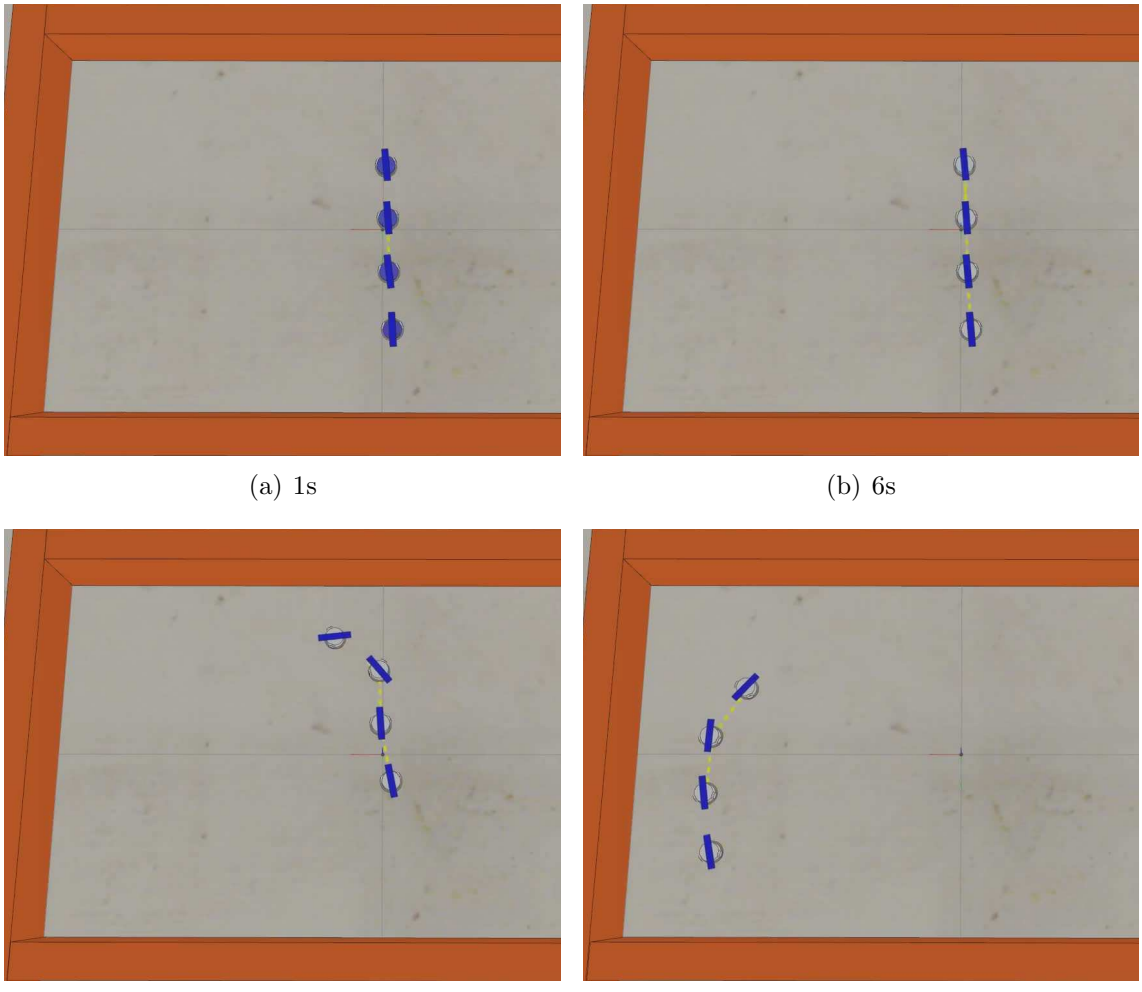
(c) 100s (d) 250s

Figura A.3: Experimento 3 de movimentação em diferentes instantes de tempo.

[3] `https://youtu.be/H3XZT_YLaoI`

A.4 Experimento 4

O Experimento 4 apresenta dois casos de movimentação em um ambiente com obstáculos. Tais obstáculos são constituídos de paredes ou muros que impedem os robôs de prosseguirem em sua trajetória. No primeiro caso, ilustrado pela Figura A.4, o enxame constitui uma linha de robôs que se movimentam pelo ambiente e evitam a colisão com as paredes. Neste exemplo, cada robô possui no máximo dois vizinhos. O primeiro vídeo com o registro do Experimento 4 está disponível para acesso *online* [4].



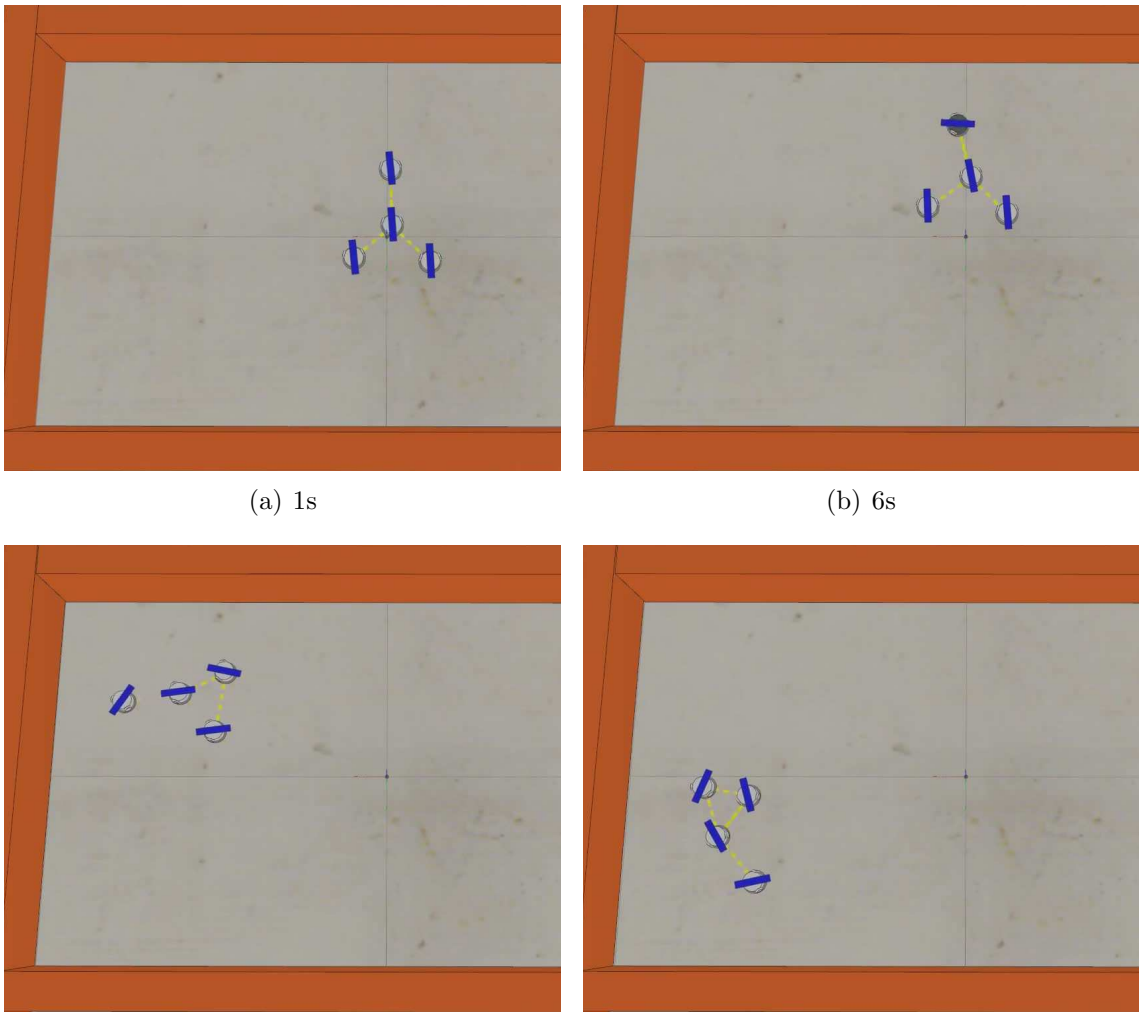
(a) 1s (b) 6s

(c) 25s (d) 100s

Figura A.4: Experimento 4 de movimentação em diferentes instantes de tempo.

[4] `https://youtu.be/WCBJ_R_H620`

No segundo caso do Experimento 4 (Figura A.5), os robôs apresentam outra organização espacial. O robô intermediário (ao centro) possui dois robôs filhos além de seu robô pai (o inicializador), totalizando assim 3 vizinhos. Assim como no caso anterior, os robôs se deslocam evitando colisões. O segundo vídeo com o registro do Experimento 4 está disponível para acesso *online* [5].



(a) 1s (b) 6s

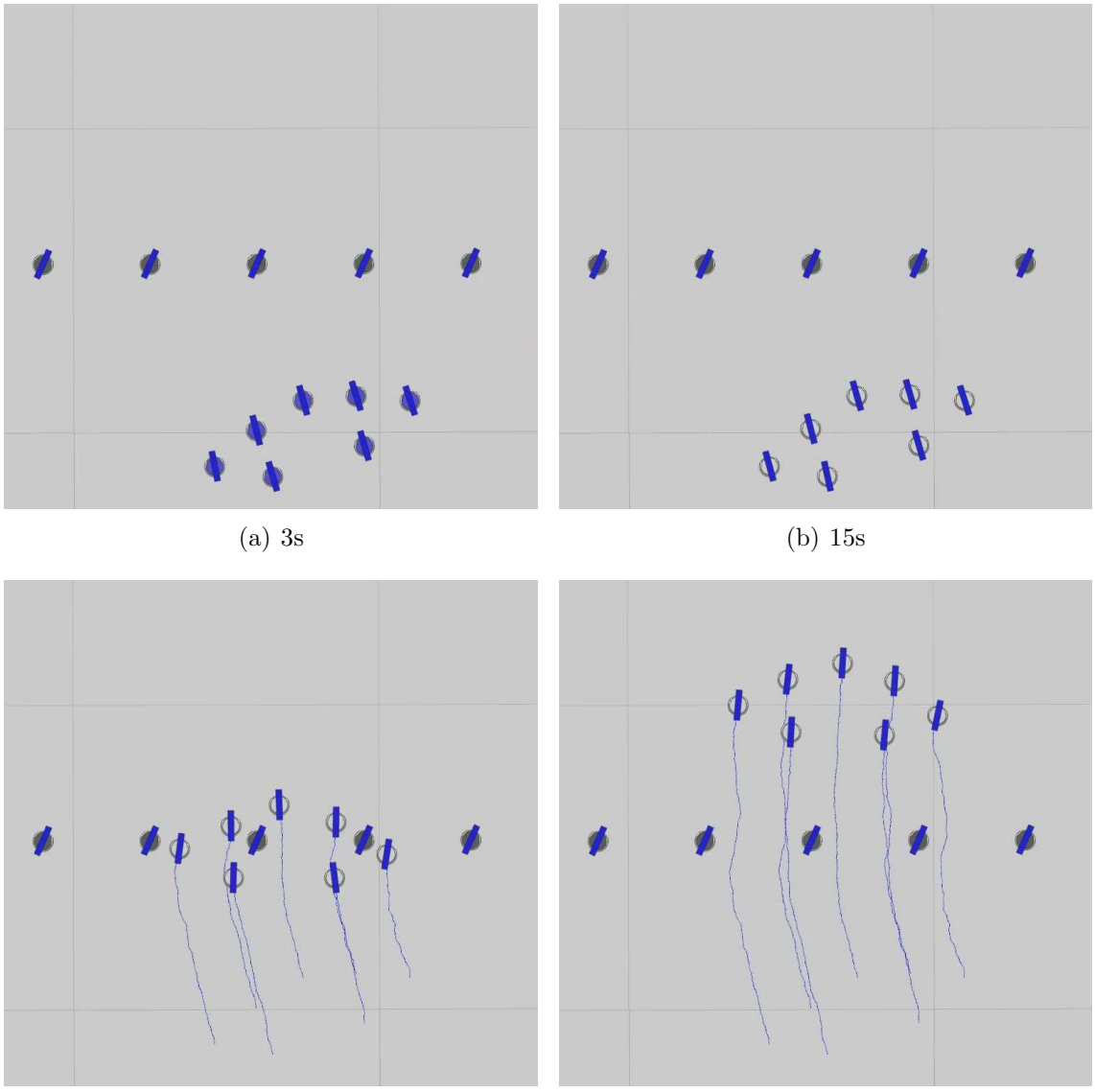
(c) 25s (d) 100s

Figura A.5: Experimento 4 de movimentação em diferentes instantes de tempo.

[5] `https://youtu.be/kFyNo0Oy_nk`

A.5 Experimento 5

O Experimento 5 apresenta dois casos de movimentação através de obstáculos. No primeiro caso, ilustrado pela Figura A.6, um enxame de 7 robôs inicia sua movimentação pelo ambiente. Robôs estáticos, não pertencentes ao enxame, são dispostos no ambiente formando uma barreira no percurso dos robôs em movimento. Neste exemplo, os robôs estáticos apenas atuam como obstáculos. Contudo, diferente da parede no Experimento 4, esta barreira pode ser atravessada. Os robôs do enxame precisam apenas desviar de cada robô da barreira, ao mesmo tempo que seguem os demais membros do enxame. O primeiro vídeo com o registro do Experimento 5 está disponível para acesso *online* [6].



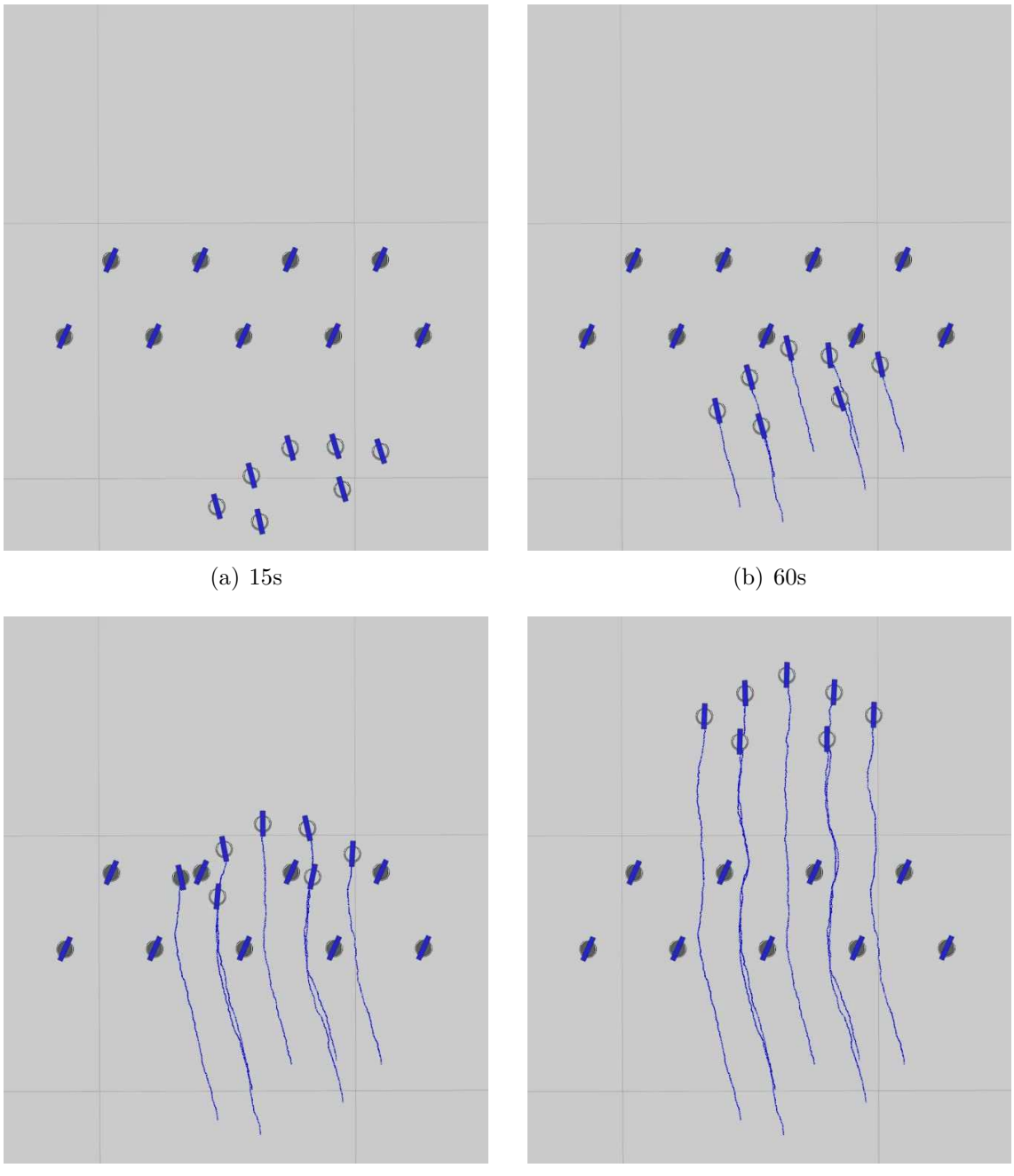
(a) 3s (b) 15s

(c) 78s (d) 130s

Figura A.6: Experimento 5 de movimentação em diferentes instantes de tempo.

[6] `https://youtu.be/xAcTjXO9d6c`

O segundo caso do Experimento 5 é exposto na Figura A.7. Este exemplo difere do anterior pela presença de duas barreiras de robôs estáticos. Também neste caso o enxame em movimento mostrou-se capaz de desviar dos obstáculos e manter a formação do grupo. O segundo vídeo com o registro do Experimento 5 está disponível para acesso *online* [7].



(a) 15s (b) 60s

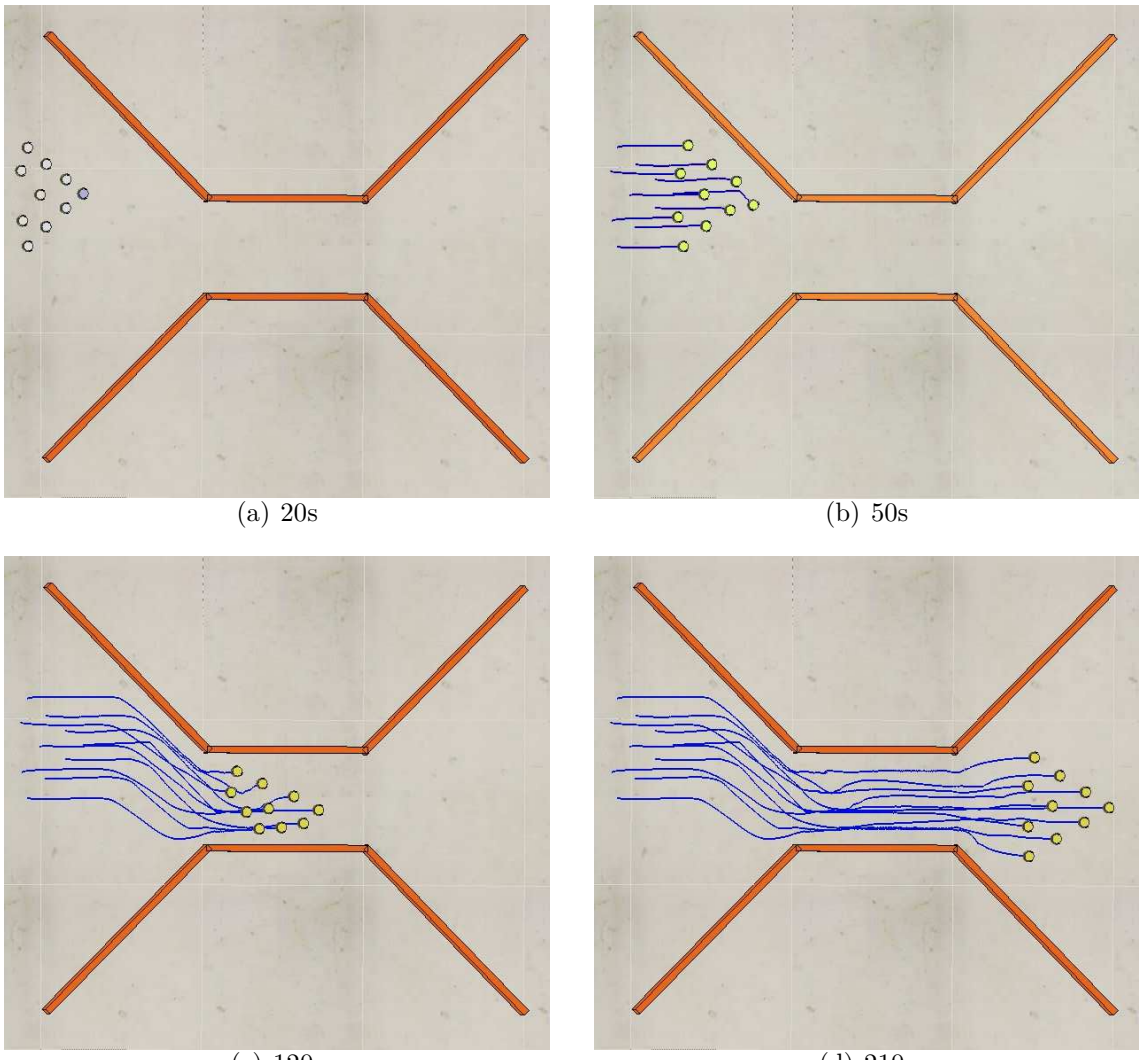
(c) 120s (d) 180s

Figura A.7: Experimento 5 de movimentação em diferentes instantes de tempo.

[7] `https://youtu.be/H_WTcmMSg-o`

A.6 Experimento 6

O Experimento 6 apresenta a navegação de um grupo de robôs em um ambiente com obstáculos (paredes). Esta operação é exposta na Figura A.8. Os robôs encontram em seu percurso um conjunto de paredes que altera a direção do trajeto do grupo como um todo. Neste experimento, as paredes foram dispostas intencionalmente de forma a guiar os robôs em direção a um corredor estreito. A diminuição do espaço no trajeto do enxame faz com que os robôs modifiquem o distanciamento mútuo existente ao início da movimentação, de forma a evitar colisões com outros robôs e com as paredes. Após saírem do corredor e da região de influência das paredes, os robôs reajustam suas distâncias, restaurando a formação inicial. Um vídeo com o registro do Experimento 6 está disponível para acesso *online* [8].



(a) 20s (b) 50s

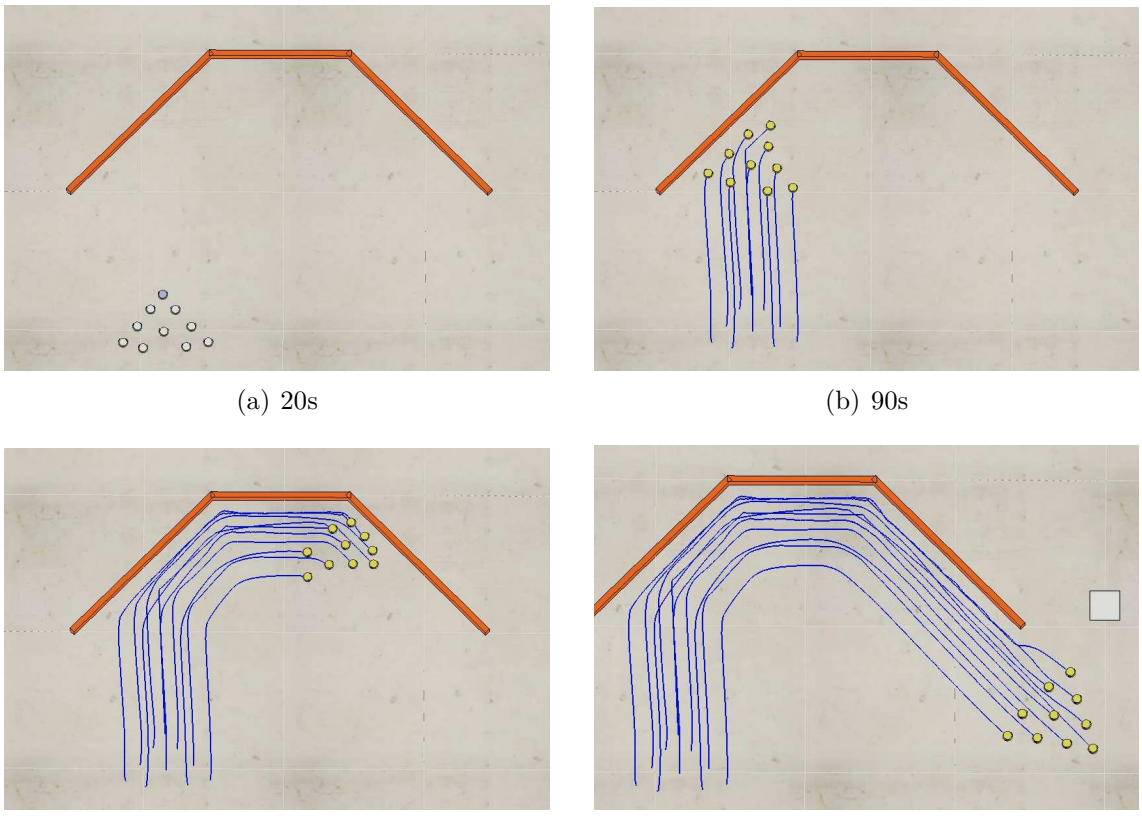
(c) 120s (d) 210s

Figura A.8: Experimento 6 de movimentação em diferentes instantes de tempo.

[8] `https://youtu.be/xVw9IyxyGhE`

A.7 Experimento 7

O Experimento 7 apresenta a navegação de um grupo de robôs em um ambiente com obstáculo côncavo. A Figura A.9 mostra a presença de paredes formando uma barreira côncava. Os robôs navegam por este ambiente, tendo sua trajetória afetada pela presença da parede. Como ocorrido no Experimento 6, os robôs modificam o distanciamento em relação aos demais robôs para evitar a ocorrência de colisões. Mais uma vez, ao saírem da área de influência das paredes, a formação inicial é restaurada. Um vídeo com o registro do Experimento 7 está disponível para acesso *online* [9].



(a) 20s (b) 90s

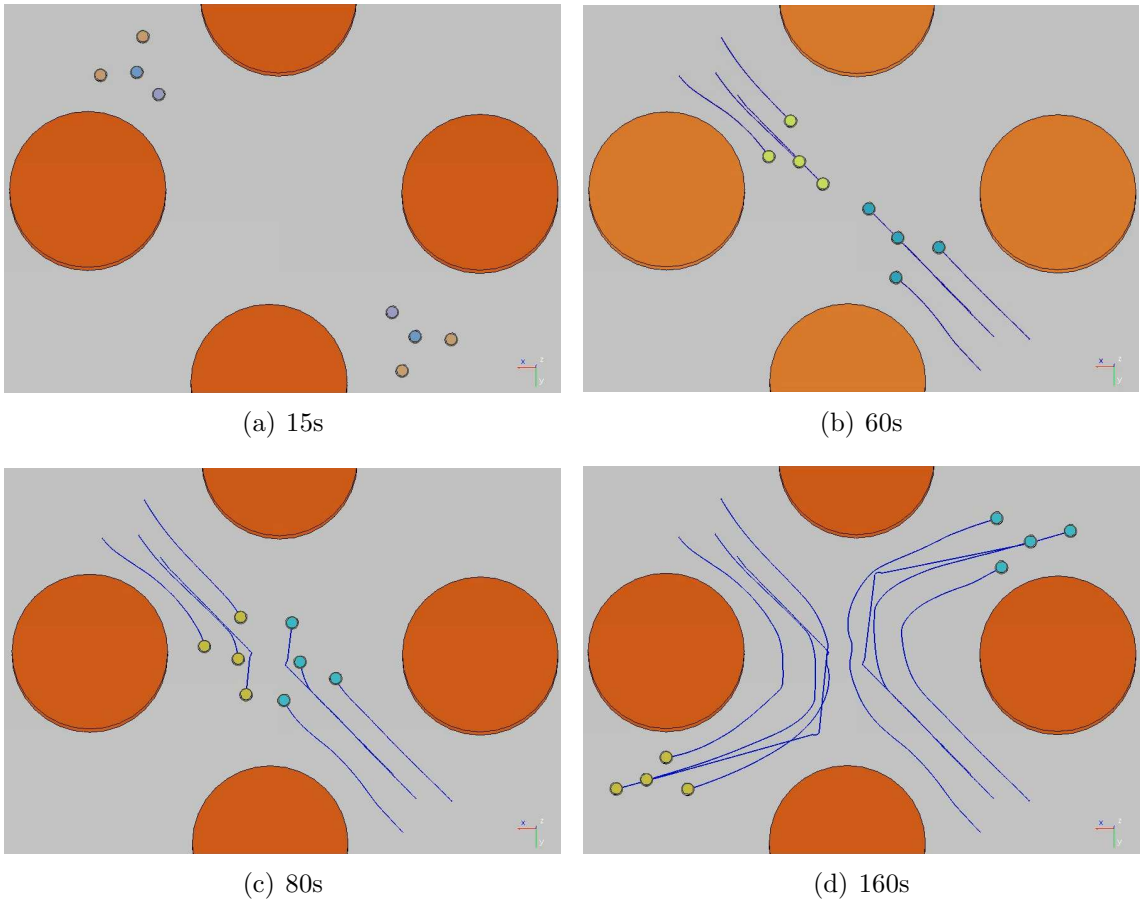
(c) 210s (d) 330s

Figura A.9: Experimento 7 de movimentação em diferentes instantes de tempo.

[9] `https://youtu.be/QgGXA8iDS94`

A.8 Experimento 8

O Experimento 8 apresenta a navegação de dois grupos de robôs em um ambiente com obstáculos. A Figura A.10 mostra um ambiente aberto contendo quatro obstáculos circulares onde os robô podem se movimentar. Neste experimento são empregados dois grupos independentes, cada qual composto por quatro robôs. Os dois inicializadores foram intencionalmente direcionados um contra o outro. Os dois grupos têm sua movimentação redirecionada para evitar colisões com robôs do grupo oposto. Além disso, os robôs de ambos os grupos também desviam dos obstáculos circulares. Um vídeo com o registro do Experimento 8 está disponível para acesso *online* [10].



(a) 15s (b) 60s

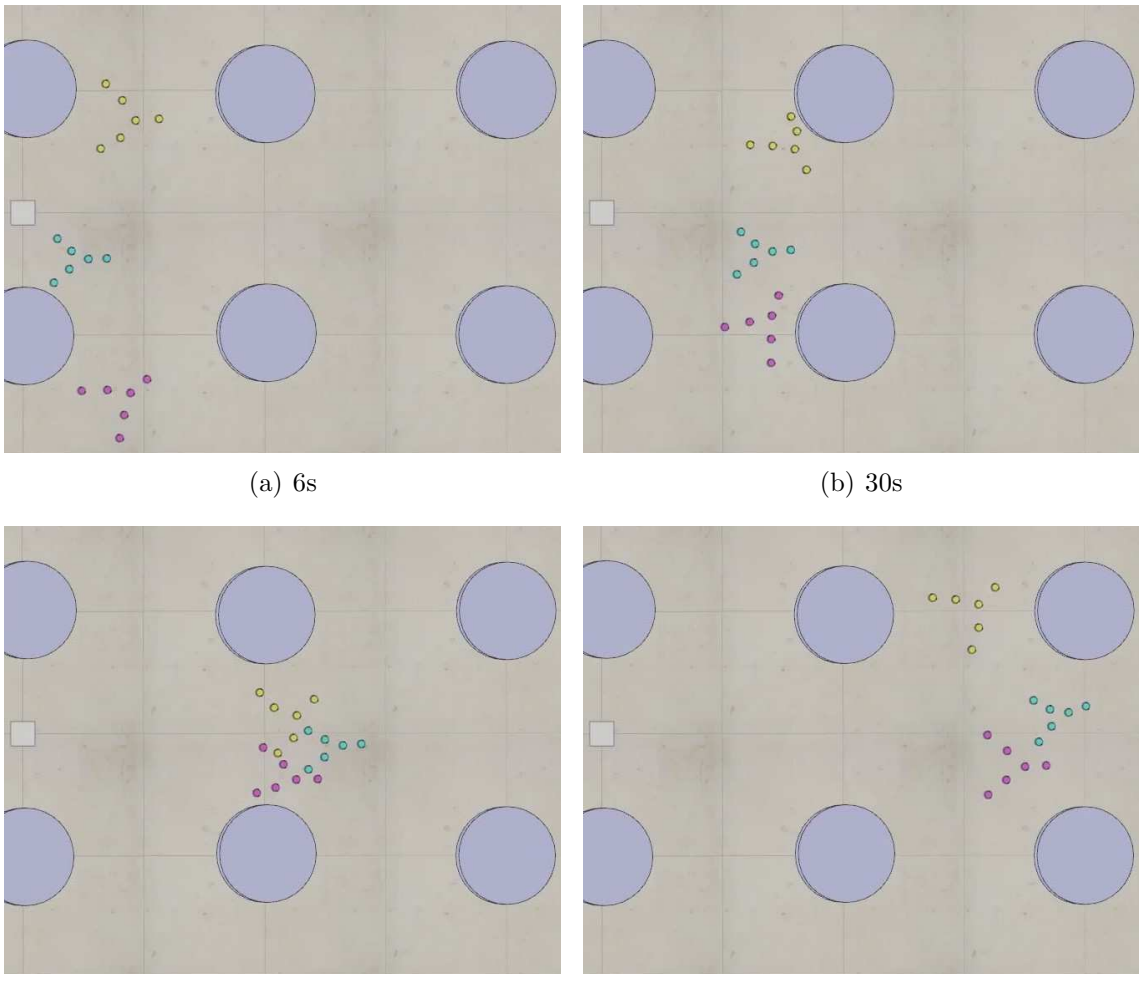
(c) 80s (d) 160s

Figura A.10: Experimento 8 de movimentação em diferentes instantes de tempo.

[10] `https://youtu.be/PTVe93EfopU`

A.9 Experimento 9

O Experimento 9 apresenta a navegação de três grupos de robôs. Neste experimento, representado pela Figura A.11, grupos independentes de robôs se deslocam em um ambiente contendo seis obstáculos circulares. A presença dos obstáculos faz com que os três grupos sejam direcionados um contra o outro. Os robôs mostram ser capazes de evitar colisões ao mesmo tempo que mantém a movimentação coordenada de cada grupo. Um vídeo com o registro do Experimento 9 está disponível para acesso *online* [11].



(a) 6s (b) 30s

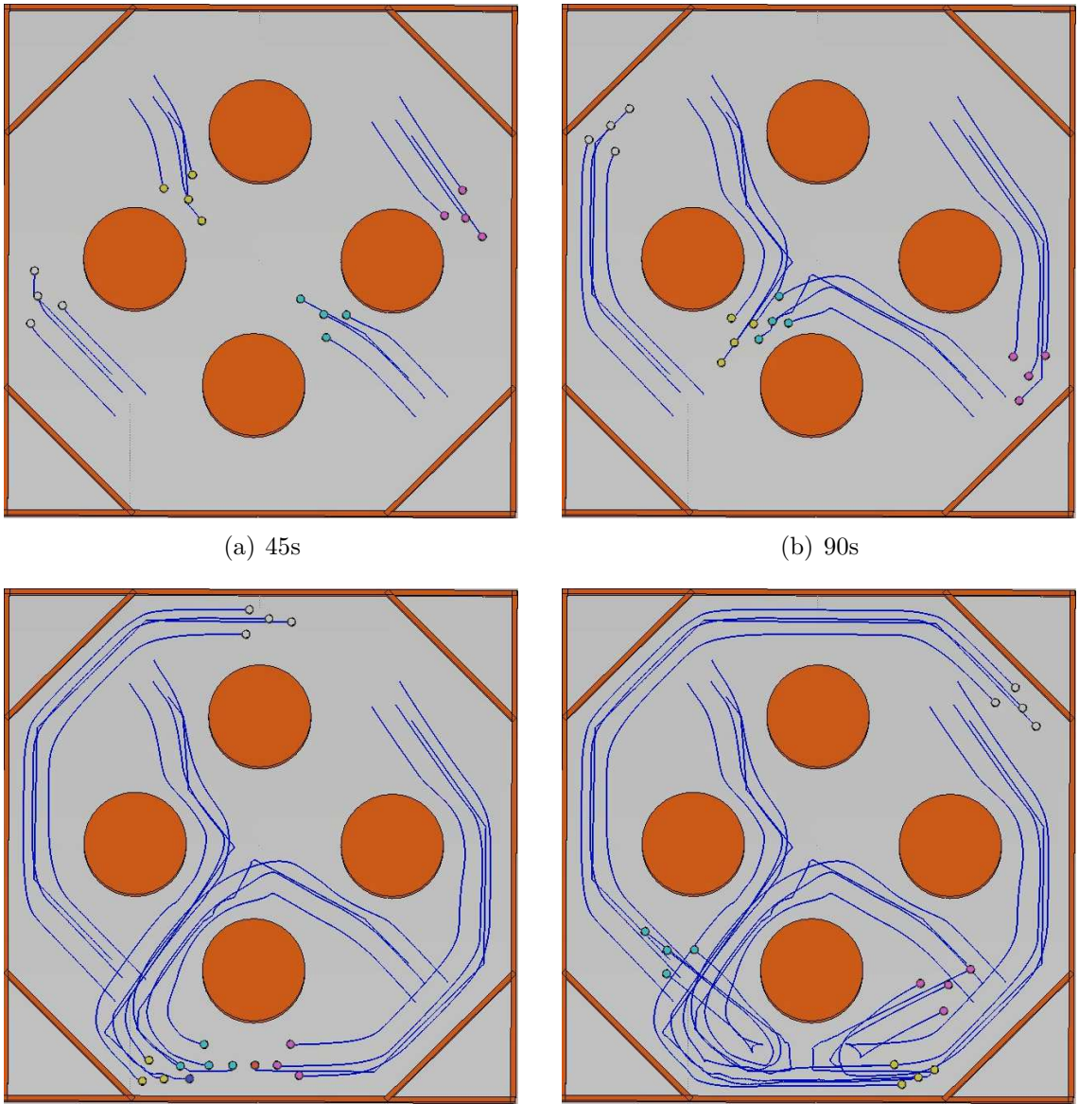
(c) 65s (d) 100s

Figura A.11: Experimento 9 de movimentação em diferentes instantes de tempo.

[11] `https://youtu.be/Jj1bjLWv2VI`

A.10 Experimento 10

O Experimento 10 apresenta a movimentação de quatro grupos de robôs em um ambiente fechado e com obstáculos. A Figura A.12 mostra que o ambiente foi montado contendo características de experimentos anteriores, como obstáculos circulares e paredes cercando o ambiente. Quatro grupos de robôs independentes movimentam-se pelo ambiente. Os rastros indicam o histórico do deslocamento de cada robô. Devido à área finita do ambiente, os grupos necessitam por mais de uma vez alterar a direção de seu trajeto a fim de evitar colisões. Um vídeo com o registro do Experimento 10 está disponível para acesso *online* [12].



(a) 45s (b) 90s

(c) 150s (d) 210s

Figura A.12: Experimento 10 de movimentação em diferentes instantes de tempo.

[12] `https://youtu.be/iAbRZqZ906Q`

Apêndice B

Lista de Publicações

B.1 Publicações em Periódicos

1. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Distributed strategy for robots recruitment in swarm-based systems."*, International Journal of Bio-Inspired Computation, Inderscience Publishers, ISSN 1758-0366, 8.2 (2016): 99-108, http://www.indersciencеonline.com/doi/abs/10.1504/IJBIC.2016.076336, Qualis-CC: B1.

2. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Wave algorithm applied to collective navigation of robotic swarms."*, Applied Soft Computing, (2016), Elsevier, ISSN 1568-4946, 57, 698-707, https://doi.org/10.1016/j.asoc.2016.06.004, Qualis-CC: A1.

3. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Review of Methodologies and Tasks in Swarm Robotics Towards Standardization"*, Swarm and Evolutionary Computation, (2017), Elsevier, Submetido em 12/2016. Revisado em 05/2017. Aguardando resposta do jornal.

B.2 Publicações em Conferências

1. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Wave algorithm for recruitment in swarm robotics"*. International Conference on Computational Science and Its Applications 2015, ICCSA 2015 . Springer International Publishing, 2015, Lecture Notes in Computer Science, ISSN 0302-9743, vol 9156, p. 3-13, https://link.springer.com/chapter/10.1007/978-3-319-21407-8_1, Qualis-CC: B1.

2. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Efficient Strategy for Collective Navigation Control in Swarm Robotics."*, International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA, Procedia Computer Science, ISSN 1877-0509, 80 (2016): 814-823, http://www.sciencedirect.com/science/article/pii/S1877050916308468, Qualis-CC: A1.

B.3 Capítulo de Livro

1. Silva Junior, L. D. R. S.; Nedjah, Nadia. *"Distributed Algorithms for Recruitment and Coordinated Motion in Swarm Robotic Systems."*, Artificial Intelligence: Concepts, Methodologies, Tools, and Applications. IGI Global, 2017, ISBN13 9781466695726, 671-693. http://www.igi-global.com/chapter/distributed-algorithms-for-recruitment-and-coordinated-motion-in-swarmrobotic-systems/173357