



Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Comportamento coletivo de exames de robôs com restrições de distâncias

Fernando Javier Mendiburu

Tese de Doutorado apresentada à Coordenadoria do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Antonio Marcus Nogueira Lima, Dr.

Orientador

Marcos Ricardo Alcântara Morais, D.Sc.

Orientador

Campina Grande, Paraíba, Brasil

©Fernando Javier Mendiburu, 22 de abril de 2019

M538c Mendiburu, Fernando Javier.
Comportamento coletivo de enxames de robôs com restrições de distâncias / Fernando Javier Mendiburu. – Campina Grande, 2019.
206 f. : il. color.

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2019.
"Orientação: Prof. Dr. Antonio Marcus Nogueira Lima, Prof. Dr. Marcos Ricardo Alcântara Morais".
Referências.


1. Robótica de Enxame. 2. Projeto Automático de Controladores. 3. Restrições de Distâncias. I. Lima, Antonio Marcus Nogueira. II. Morais, Marcos Ricardo Alcântara. Título.

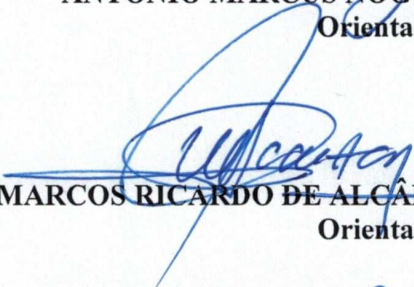
CDU 004.896(043)

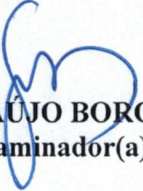
**"COMPORTAMENTO COLETIVO EM ENXAMES DE ROBÔS COM RESTRIÇÕES DE
DISTÂNCIAS "**

FERNANDO JAVIER MENDIBURU


TESE APROVADA EM 22/04/2019



ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador(a)


MARCOS RICARDO DE ALCÂNTARA MORAIS, D.Sc, UFCG
Orientador(a)


GEOVANY ARAÚJO BORGES, Dr., UNB
Examinador(a)

ANDRÉ GUSTAVO SCOLARI CONCEIÇÃO, Ph.D., UFBA
Examinador(a)


JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG
Examinador(a)


ANGELO PERKUSICH, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

Dedicatória

Dedicação feita especialmente a toda nossa grande família Mendiburu-Molina e família Oliveira-Velozo.

Aos meus pais, Elsa Haydée Molina e Luis Alberto Mendiburu, por sempre acreditarem em mim, terem me dado a possibilidade de estudar e estarem sempre presente quando preciso.

Aos meus irmãos, Mariana Alicia Mendiburu e Alejandro Luis Mendiburu pela torcida constante, mesmo na distância.

A minha amada esposa, Angélica Oliveira Velozo pela companhia e apoio durante todos estes anos juntos.

Agradecimentos

Em primeiro lugar, gostaria de agradecer a Deus por ser minha luz interior, ter me dado serenidade e coragem para enfrentar grandes desafios.

Aos meus orientadores Antonio Marcus Nogueira Lima e Marcos Ricardo Alcântara Morais pelos ensinamentos, amabilidade, disposição, conselhos e discussões em busca da qualidade e excelência do trabalho.

Agradecimento especial para o professor Edson Guedes da Costa, que me motivou e encorajou para realizar o doutorado sanduíche. Por sua vez, foi a mesma grande pessoa que me ajudou em 2008 quando mais precisava.

Ao professor Mauro Birattari, por ter me aberto as portas do seu laboratório, ter me fornecido todo suporte necessário e ter contribuído fortemente na minha pesquisa.

Aos amigos do laboratório e-robótica e da universidade, Davi Juvêncio, Felipe Nascimento, Yuri Loia, Algeir Sampaio, Dalton Valadares, Fernandes Matias, Mikael Themoteo pelas conversas acadêmicas, da vida e predisposição sempre que precisava.

Aos amigos e colegas do laboratório IRIDIA, em especial aos meus amigos David Garzón Ramos, Jonas Kuckling, Muhammad Salman e sua esposa Sarah que foram sempre prestativos. David tem contribuído fortemente no meu trabalho com suas idéias, na minha formação como pesquisador e seus conselhos pertinentes.

Aos meus pais Elsa e Luis, pelo esforço realizado durante todos estes anos para que eu e meus irmãos tenhamos a oportunidade de estudar e crescer nas nossas vidas.

Aos meus irmãos e cunhados, Alejandro (Jazmín) e Mariana (Mauri), pelo apoio e compreensão, assim como meus pais, mesmo em minha ausência.

A minha querida esposa Angélica, que contribuiu muito para que este trabalho seja realizado. Ela foi e continuará sendo minha psicóloga, redatora e tradutora, dentre outras ajudas inestimáveis. Por seu apoio incondicional, serei eternamente grato.

Aos meus sogros Mauro e Ivoneide e meus cunhados Alisson, Andrezza, Gisele e Junior pela torcida, boas energias e preocupação comigo.

Agradeço também aos funcionários da COPELE, em especial a Pedro e Angela, pela ajuda e disposição na resolução de problemas.

Agradeço ao apoio financeiro durante meu doutorado do CNPq e da CAPES que possibilitou minha dedicação integral a esta pesquisa.

Agradeço a um conjunto de 20 robôs e-puck por estarem disponíveis sempre que precisei, não ter dado problemas mecânicos frequentes e não consumir muita bateria. Sem eles, a validação nos experimentos físicos seria impossível. Agradeço também a mais 2 robôs reservas que estavam disponíveis ante qualquer inconveniente.

Por fim, gostaria de agradecer a todos aqueles que contribuíram direta ou indiretamente neste trabalho.

Fernando Javier Mendiburu

Resumo

Nesta tese é abordado o problema de restrições de distâncias no contexto de comportamento coletivo em enxames de robôs. São propostas duas novas especializações do projeto automático de controlador individual de cada robô: uma especialização modular e outra monolítica. O objetivo destas novas especializações é projetar o controlador individual de cada robô para produzir o comportamento coletivo do enxame, no qual deve ser mantida distâncias específicas entre os robôs. Abordagens de síntese automática em enxames de robôs, anteriormente propostas na bibliografia, não consideram explicitamente estas restrições de distâncias na especialização do método. O método modular modifica o controlador de baixo nível do robô para permitir o controle de velocidade linear e angular do mesmo. Por sua vez, é incluído um novo comportamento individual, chamado *formar*. Este comportamento, projetado com a teoria de campos potenciais artificiais, permite incluir a restrição de distância num controlador modular baseado em comportamentos. O comportamento *formar* permite o ajuste das interações entre os robôs para produzir o comportamento coletivo de coesão do enxame. O método monolítico adiciona o mesmo comportamento *formar* num controlador em forma de rede neuronal sem nós ocultos. Foi realizado um estudo comparativo, em simulação, dos principais métodos automáticos apresentados na bibliografia que abordam tarefas coletivas com restrições de distâncias. Por sua vez, esses métodos foram comparados com as especializações propostas nesta tese: o método modular e o método monolítico. Os métodos foram comparados em simulação e com robôs físicos em três tarefas de robótica de enxame, utilizando 20 robôs e-puck. Dos resultados observados se conclui que os métodos de projeto automático presentes na bibliografia, assim como também o método monolítico proposto nesta tese, não conseguiram favorecer a obtenção de comportamentos coletivos relevantes de coesão entre os robôs do enxame. Porém, no método modular, se conclui que as restrições introduzidas produzem comportamentos coletivos relevantes com coesão e o desempenho dos controladores projetados é significativamente superior aos obtidos com outros métodos presentes na bibliografia.

Palavras-chave: Robótica de enxame. Projeto automático de controladores. Restrições de distâncias.

Abstract

In this thesis is addressed the problem of spatially organize behaviors in the context of automatic design of software controllers for swarm robots. Two new specializations are proposed for the automatic design of software controller of each robot: modular and monolithic specialization. The purpose of these new specializations is designing the individual controller of each robot to produce the swarm collective behavior in which specific distances among robots must be maintained. Automatic design methods for software controllers in swarms of robots, previously proposed in the bibliography, do not explicitly consider these distance restrictions in the specialization of the method. The modular method modifies the robot's low-level controller to allow linear and angular velocity control of the robot. Also, a new individual behavior is included in each robot, called *formation*. This behavior, designed using the artificial potential theory, allows to include distance constraints in a behavior-based controller. The *formation* behavior allows the adjustment of the interactions among robots to produce the collective behavior of the swarm. The monolithic method adds the same *formation* behavior in a neural network architecture without hidden nodes. We carry out experiments in simulation of the main automatic design methods presented in the bibliography that deal with collective tasks and distance restrictions. Also, these methods were compared with the specializations proposed in this thesis: the modular method and the monolithic method. These methods were compared in simulation and real robots in three swarm robotic tasks, using 20 e-puck robots. From the observed results, it can be concluded that the automatic design methods presented in the bibliography, as well as the monolithic method proposed in this thesis, failed to produce relevant collective behaviors and cohesion among robots. However, in the modular method, it is concluded that the introduced constraints produce cohesion and relevant collective behaviors and the performance of the designed controllers is significantly better than those obtained with methods present in the bibliography.

Keywords: Swarm robotics. Automatic design of controllers. Distance constraints.

Sumário

1	Introdução	1
1.1	Descrição do problema	7
1.2	Hipótese	8
1.3	Contribuição	9
1.4	Justificativa	9
1.5	Objetivos	10
2	Revisão bibliográfica	12
2.1	Projeto manual de controladores para robô de enxames.	13
2.1.1	Teoria de controle supervisório	14
2.1.2	Abordagem baseadas em comportamento	15
2.1.3	Abordagem de campos potenciais artificiais	16
2.2	Projeto automático de controladores para robô de enxames.	19
2.2.1	Arquiteturas modulares	20
2.2.2	Arquiteturas monolíticas	26
2.3	Considerações finais	28
3	Fundamentação Teórica	31
3.1	Sistemas multi-robô	32
3.2	Enxames de robôs	36
3.2.1	Principais características	36
3.3	Definição de comportamento	38
3.3.1	Coordenação de comportamentos	41
3.4	Comportamento coletivo	44

3.4.1	Tipos de comportamentos coletivos	45
3.5	Robôs utilizados em enxames de robôs	50
3.5.1	Robô e-puck	51
3.5.2	Robô Foot-bot	52
3.6	Projeto de controle para enxames de robôs	53
3.6.1	Tipos de controladores	54
3.6.2	Síntese de controladores	60
3.7	AUTOMODE e EVOSTICK	61
3.7.1	O poder de representação	61
3.7.2	AUTOMODE	63
3.7.3	EVOSTICK	66
3.8	Especialização de AUTOMODE e EVOSTICK	68
3.8.1	A plataforma robótica	68
3.8.2	O espaço de busca do controlador	70
3.8.3	O simulador computacional	74
3.8.4	A função objetivo	78
3.8.5	Condições experimentais	84
3.8.6	O algoritmo de otimização	86
3.9	Considerações finais	94
4	AutoMoDe-Chocolate e EvoStick	95
4.1	Modelo de robô e sensores	97
4.1.1	Modelo cinemático do robô	97
4.1.2	Modelo de sensores e atuadores do robô	98
4.2	Especialização dos métodos de projeto	102
4.2.1	Controlador de baixo nível do robô	103
4.3	Metodologia	105
4.4	Resultados experimentais	109
4.4.1	Comportamento de agrupação de objetos	109
4.4.2	Comportamento de agregação	111
4.4.3	Comportamento de formação de padrões	119

4.5	Considerações finais	127
5	Comportamentos com restrições de distâncias	132
5.1	Modelo de robô e sensores	133
5.1.1	Modelo cinemático do robô	133
5.1.2	Modelo de sensores e atuadores do robô	133
5.2	Arquitetura do comportamento <i>formar</i>	135
5.2.1	Escolha da lei de controle	137
5.2.2	Limitações da abordagem	141
5.3	Sistema de controle do robô	143
5.3.1	Controlador supervisor	145
5.3.2	Controlador de baixo nível	147
5.4	Resultados experimentais	148
5.4.1	Comportamento <i>formar</i> no cenário sem obstáculos	150
5.4.2	Comportamento <i>direcionar</i> e <i>dispersar</i> no cenário sem obstáculos	151
5.4.3	Movimento coletivo com coesão: Comportamento <i>reunir</i>	153
5.4.4	Grau de acurácia $\delta(t)$ no movimento coletivo	156
5.5	Considerações finais	159
6	Extensões com restrições de distâncias	160
6.1	Especialização de AUTOMODE-MATE e EVOSPACE	162
6.2	Metodologia	167
6.3	Resultados experimentais	169
6.3.1	Cobertura simples	169
6.3.2	Cobertura em rede	176
6.3.3	Cobertura condicional	182
6.4	Considerações finais	188
7	Conclusão	191
7.1	Perspectivas de trabalhos futuros	193
	Referências bibliográficas	195

Lista de símbolos e abreviaturas

$v_{i_{cam}}$	Variável de leitura da distância relativa ao LED i do robô vizinho
α	Pendente da curva de força na zona de operação
$\delta(t)$	Grau de acurácia do enxame de robôs
ϵ	Profundidade do campo potencial
\hat{a}	Vetor orientação unitário expressado no sistema de coordenadas O_c
ω	Velocidade angular do robô, $\omega \in \mathbb{R} : [-\omega_{max}, \omega_{max}]$
ω_{max}	velocidade angular máxima permitida no robô
$\phi_{i_{cam}}$	Variável de leitura do ângulo relativo ao LED i do robô vizinho
$\phi_{i_{luz}}$	Variável de leitura do ângulo do sensor de luminosidade i
$\phi_{i_{prox}}$	Variável de leitura do ângulo do sensor de proximidade i
$\phi_{i_{R\&B}}$	Variável de leitura do ângulo relativo ao robô vizinho i
ϕ_{ij}	Ângulo entre dois robôs medido desde robô i , $i \neq j$
$\rho_g(t)$	Grau de orientação entre os robôs do enxame
Θ^{elite}	Controladores candidatos sobreviventes da iteração j
Θ^{nova}	Controladores candidatos novos para a iteração $j++$
θ_d	Ângulo desejado de direção em função do sistema de coordenadas O_c
Θ_j	Espaço de busca (controladores candidatos) na iteração j
B_j	Quantidade máxima de experimentos permitidas na iteração j
B_{usado}	Quantidade de experimentos realizadas num determinado momento

$C(N_{ins}, \Theta)$	Matriz função de custo de N_{ins} fileiras e $ \Theta $ colunas
$C(\theta)$	Função objetivo dependente do controlador θ instanciado nos robôs
C_w	Cenário de trabalho onde os robôs desenvolvem as tarefas
d_{ij}	Distância entre dois robôs, $i \neq j$
d_{ij}^*	Distância desejada entre dois robôs, $i \neq j$
$e_i(p, \lambda)$	Estímulo i proveniente do ambiente da classe p e módulo λ
F_{ij}	Força virtual aplicada no robô i produzida pelo robô j , $i \neq j$
gnd_i	Variável de leitura da cor do chão do sensor de chão i
$I_{n_{ins}}$	Instância amostrada número n_{ins}
k_ω	Ganho do controlador proporcional de velocidade angular do robô
k_p	Ganho do controlador proporcional
k_v	Ganho do controlador proporcional de velocidade linear do robô
L	Distância do eixo que conecta as rodas do robô diferencial
led_i	Variável de escrita do estado do LED i , $led_i \in \mathbb{N} : [0,1]$
N^{iter}	Quantidade de iterações do algoritmo de otimização
N^{param}	Quantidade de parâmetros $x \in X$ a serem ajustados
N_{ins}	Quantidade máxima de instâncias
n_{ins}	Quantidade de instâncias amostradas
$n_{R\&B}$	Variável de leitura da quantidade de robôs vizinhos
O_c	Sistema de coordenadas fixo no robô
o_i	Vetor orientação unitário i expressado no sistema de coordenadas O_w
O_w	Sistema de coordenadas fixo no cenário C_w
P_{ij}	Potencial virtual aplicado no robô i produzido pelo robô j , $i \neq j$
PID	Controlador proporcional, integral e derivativo
r_{cam}	Vetor resultante medido com a câmera

r_{exp}	Vetor resultante do comportamento exploração
r_{luz}	Vetor resultante medido com os sensores de luminosidade
r_{prox}	Vetor resultante medido com os sensores de proximidade
$r_{R\&B}$	Vetor resultante medido com o sensor de proximidade relativa
r_t	Vetor resposta total do sistema
T_a	Ciclo de controle do controlador
T_m	Duração de um experimento particular
v	Velocidade linear do robô, $v \in \mathbb{R} : [-v_{max}, v_{max}]$
$v_{i_{luz}}$	Variável de leitura do alcance do sensor de luminosidade i
$v_{i_{prox}}$	Variável de leitura do alcance do sensor de proximidade i
$v_{i_{R\&B}}$	Variável de leitura da distância relativa ao robô vizinho i
v_i	Variável de escrita da velocidade linear da roda i do robô, $i \in \{l, r\}$
v_{max}	velocidade linear máxima permitida no robô
X	Conjunto de parâmetros do controlador $\theta = \{x_1, \dots, x_{Nparam}\}$
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
COPELE	Coordenadoria de Pós-Graduacao em Engenharia Elétrica
IRIDIA	Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

Lista de Tabelas

2.1	Limitações dos métodos de projeto manual apresentados na bibliografia. . . .	14
2.2	Limitações dos métodos de projeto automático apresentados na bibliografia.	20
3.1	Modelo de referência para o robô e-puck (RM1).	69
6.1	Modelo de referência para o robô e-puck (RM1.3).	163

Lista de Figuras

3.1	Classificação de sistemas com vários robôs: sistemas multi-robô (setas em verde) e enxames de robôs (setas em azul).	33
3.2	Diagrama estímulo-resposta de um robô com comportamentos b_i , coordenadores c_i , vetores r_i , r_{c_i} e estímulos e_i	41
3.3	Robô de tração diferencial num cenário com centro de coordenadas O_w e obstáculos em cor verde.	43
3.4	Vista lateral do robô e-puck apresentando o seus sensores e atuadores.	51
3.5	Robô Foot-bot especificando seus módulos e sensores.	52
3.6	Projeto de síntese de controlador para robô de enxame utilizando o método AUTOMODE-CHOCOLATE, apresentando as principais características.	65
3.7	Projeto de síntese de controlador para robô de enxame utilizando o método EVOSTICK, apresentando as principais características.	67
3.8	Ambiente real (esquerda) e ambiente simulado (direita) com as características de interesse modeladas no cenário.	85
3.9	Fluxograma do algoritmo F-Race iterado	87
3.10	Fluxograma do algoritmo F-Race usado para o método AUTOMODE	90
3.11	Fluxograma do algoritmo de robótica evolutiva usado para o método EVOSTICK	93
4.1	Robô i apresentando a distância relativa d_{ij} e ângulo ϕ_{ij} aos vizinhos, $j \neq i$	99
4.2	Controle de baixo nível do robô, apresentando o controle de velocidade, ângulo e ao módulo para limitar motores.	103

4.3	Imagem superior dos cenários simulados. De cima para baixo e de esquerda à direita: procura de objetos , acesso restrito ao refúgio , agregação , agregação com informação no ambiente , cobertura e monitoramento de região e cobertura da maior rede do enxame . As imagens mostram também a pose inicial dos 20 robôs para cada missão. A imagem inferior apresenta o sistema de coordenadas localizado no centro de cada cenário.	106
4.4	Diagrama de caixas apresentando o desempenho dos métodos CHOCOLATE e EVOSTICK na missão Procura de objetos , quanto maior a função objetivo melhor desempenho do método.	110
4.5	Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão Acesso restrito ao refúgio , quanto maior a função objetivo melhor desempenho do método.	113
4.6	Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão Agregação , quanto maior a função objetivo melhor desempenho do método.	116
4.7	Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão Agregação com informação no ambiente , quanto maior a função objetivo melhor desempenho do método.	118
4.8	Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão cobertura e monitoramento de região , quanto menor a função objetivo melhor desempenho do método.	121
4.9	Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão Cobertura da maior rede do enxame , quanto maior a função objetivo melhor desempenho do método.	124
4.10	Tamanho do maior conjunto de robôs (N_B) em função do tempo para a missão Cobertura da maior rede do enxame	126
4.11	Media e desvio padrão em função do tempo para a missão Cobertura da maior rede do enxame	127

5.1	Diferentes ajustes de parâmetros para a função de Lennard-Jones. Curva "a" é uma coesão rígida, curva "b" é uma coesão líquida e curva "c" os robôs não possuem coesão.	139
5.2	Sistema de controle apresentando o controlador do robô e os sensores e atuadores utilizados.	144
5.3	Controlador supervisor apresentando os comportamentos <i>direcionar</i> , <i>dispersar</i> , <i>formar</i> e <i>reunir</i> com seus vetores de entrada e saída para um único robô. . .	145
5.4	Disposição de 7 robôs que utilizam o comportamento <i>formar</i> unicamente. . .	150
5.5	Disposição de 40 robôs que utilizam o comportamento <i>formar</i> unicamente. .	151
5.6	Disposição de 7 robôs que utilizam os comportamentos <i>direcionar</i> e <i>dispersar</i> . 152	
5.7	Disposição de 40 robôs que utilizam os comportamentos <i>direcionar</i> e <i>dispersar</i> . 152	
5.8	Movimento coletivo com 7 robôs utilizando o comportamento <i>reunir</i>	153
5.9	Movimento coletivo com 40 robôs utilizando o comportamento <i>reunir</i>	154
5.10	Movimento coletivo para o caso de 7 robôs no cenário com obstáculos.	155
5.11	Movimento coletivo para o caso de 40 robôs no cenário com obstáculos.	156
5.12	Grau de acurácia $\delta(t)$ no movimento coletivo de 100 robôs no cenário com obstáculos.	158
6.1	Imagem das arenas simuladas (imagens superiores) e físicas (imagens inferiores): <i>cobertura simples</i> (esquerda), <i>cobertura em rede</i> (centro) e <i>cobertura condicional</i> (esquerda). As imagens mostram também a pose inicial dos 20 robôs para cada missão.	167
6.2	Diagrama de caixas apresentando o desempenho dos controladores para os métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão Cobertura simples , quanto menor a função objetivo melhor desempenho do método.	171
6.3	Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão cobertura simples	173
6.4	Controlador representativo produzido pelo método AUTOMODE-MATE para a missão cobertura simples	175

6.5	Diagrama de caixas apresentando o desempenho dos controladores para os métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão cobertura em rede , quanto maior a função objetivo melhor desempenho do método.	177
6.6	Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão cobertura em rede	180
6.7	Controlador representativo produzido pelo método AUTOMODE-MATE para a missão cobertura em rede	181
6.8	Diagrama de caixas apresentando o desempenho para os controladores dos métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão Cobertura em rede , quanto maior a função objetivo melhor desempenho do método.	183
6.9	Controlador representativo produzido pelo método AUTOMODE-MATE para a missão cobertura condicional	186
6.10	Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão cobertura condicional	187

Capítulo 1

Introdução

A história moderna da robótica começa em 1940, quando os manipuladores robóticos começaram a ser usados na manipulação de material radioativo. Os manipuladores robóticos começaram ser utilizados em outras áreas, principalmente no setor automotivo no fim dos anos 1960. No setor automotivo, o uso de manipuladores se restringe a ambientes estáticos e movimentos predefinidos. Mediante os avanços da microeletrônica, o uso de técnicas de visão computacional para a detecção de objetos proporcionou a flexibilização de restrições de ambientes estáticos e permitiu o uso de planejadores de trajetórias [1].

Um dos desafios da robótica moderna foi projetar robôs móveis para realização de tarefas em ambientes não estruturados. Assim, como primeiro estudo de referência surge em 1966 o projeto que dá fruto à construção do robô de tração diferencial Shakey no Instituto de Pesquisa de Stanford [1]. Antecedentes deste tipo de robô móvel podem ser encontrados no robô de Braitenberg, que usa tração diferencial e os atuadores respondem a estímulos do ambiente de forma direta sem capacidades de planejamento. Em meados dos anos 1970, robôs de diversos tipos começaram a ser adquiridos e construídos nos laboratórios de pesquisa. Paralelamente a estes estudos de pesquisa, havia o crescimento da robótica industrial que ocorreu principalmente no Japão, Estados Unidos e em vários países europeus [1]. Este crescimento da robótica foi fomentado principalmente pela redução do custo de produção dos robôs e os avanços tecnológicos tanto no hardware como no software destes robôs [1].

A cooperação e coordenação de múltiplos robôs tem sido objeto de consideráveis esforços de pesquisa desde a década de 1980 [1]. Um conjunto de robôs que opera de modo coordenado

para executar uma tarefa coletivamente é denominado sistema multi-robô. A motivação básica para estudar sistemas multi-robô é a premissa de que um sistema multi-robô pode executar tarefas com mais eficiência do que um único robô ou pode executar tarefas não viáveis para um único robô. Além disso, os sistemas multi-robô apresentam vantagens como aumentar a tolerância a possíveis falhas nos robôs, proporcionando flexibilidade na execução da tarefa ou aproveitando as vantagens da detecção e atuação distribuídas. O uso de um pelotão de veículos pode ser de interesse em aplicações, tais como exploração de um ambiente desconhecido, controle de navegação e formação, desminagem, transporte de objetos; estes podem envolver veículos terrestres, aéreos, submarinos ou de superfície.

Um exemplo claro de tarefa na qual são necessários conjuntos de vários robôs é a tarefa de resgate de vítimas num ambiente de catástrofe. Um único robô pode fracassar numa tarefa deste tipo, pois seus recursos estão concentrados: uma falha irresolúvel no mecanismo de tração do robô impede que ele explore o ambiente; uma falha em algum sensor impede que detecte sinais relevantes no ambiente, como vítimas, obstáculos, etc. Porém, abordar o problema com sistemas multi-robô poderia evitar estes inconvenientes, visto que o sistema distribui os recursos em vários robôs, permitindo paralelizar a tarefa em vários subprocessos feitos por diferentes robôs simultaneamente para: explorar o ambiente, buscar as vítimas, remover obstáculos, dentre outros.

Os sistemas multi-robô possuem características positivas que são o ponto chave para explicar as várias pesquisas feitas na área e o rápido desenvolvimento destes sistemas. Algumas das características positivas em comparação com um único robô são: paralelismo nas atividades, aumento da quantidade de tarefas realizáveis, tolerância a falhas, distribuição de sensoriamento e atuação [2]. O projeto de hardware dos sistemas multi-robô pode ser mais econômico que o projeto de um único robô complexo no que diz respeito a custo final na sua realização [3].

Existem tarefas que requerem vários robôs para sua resolução e tarefas que poderiam ser resolvidas por vários robôs [3]. Por exemplo, sistemas multi-robô são relevantes em *aplicações industriais*, como exemplos: transporte na manufatura, monitoramento na agricultura e construção civil. Em *aplicações médicas* como robôs assistenciais em hospitais, resgate de vítimas em catástrofes e nano-robôs cirúrgicos. Em *aplicações domésticas*, como robôs aspiradores de pó, limpadores de piscinas, cortadores de grama, robôs assistenciais e de

entretenimento, entre outros. Em *aplicações militares*: exploração; monitoramento de fronteiras; seguimento de químicos; mapeamento; detecção de material perigoso; exploração espacial, entre outras [4–6].

Sistemas multi-robô podem ser classificados de acordo com a quantidade de robôs que compõem o sistema, a composição (sistema homogêneo ou heterogêneo) e o tipo de comunicação envolvido caso possua (alcance, topologia e largura de faixa) [3]. Uma classificação de vários trabalhos relevantes é feita usando essa taxonomia [3]. Os sistemas multi-robô podem ser classificados pelo tipo de arquitetura, se ela é homogênea ou heterogênea [3, 7]. Diferente do realizado em Dudek et al.[3], uma classificação baseada no tipo de comunicação pode ser realizada, catalogada como comunicação baseada no ambiente, no sensoriamento ou em comunicação direta [7]. No tocante às arquiteturas, elas podem ser classificadas como arquiteturas centralizadas e descentralizadas [7]. Por sua vez os autores tratam do grau de capacidade de modelagem das ações de outros robôs do sistema. Uma taxonomia em camadas foi realizada em Iocchi, Nardi e Salerno[8], seguindo parte da taxonomia anteriormente mencionada em [7], que classifica os sistemas multi-robô de acordo com a existência ou não de cooperação, o grau de conhecimento sobre a presença de outros robôs, o grau de coordenação e o tipo de organização da coordenação [8]. Finalmente, foram feitas classificações que abordam conceitos similares [9] e/ou com uma revisão atualizada do estado da arte [10, 11].

Paralelamente com as pesquisas realizadas em sistemas multi-robô se começa a abordar o problema de explorar mais profundamente as capacidades de redundância e tolerância a falhas do sistema, utilizando robôs com capacidades de sensoriamento e atuação limitadas. Buscava-se sistemas escaláveis na quantidade de membros e autônomos em relação à capacidade de tomar decisões individualmente de forma descentralizada. Esta nova abordagem foi nomeada de robótica de enxame (*swarm robotics*) [12]. A robótica de enxame é uma abordagem inspirada fortemente na biologia, em que uma grande quantidade de robôs desenvolve tarefas em um ambiente específico usando controladores simples e descentralizados. Este tipo de sistema possui características interessantes, como redundância, flexibilidade, tolerância a falhas, execução paralela de tarefas e escalabilidade [12, 13]. Nesta abordagem, são reportados experimentos com centenas e milhares de entidades robóticas [14, 15], não só em simulação como também utilizando robôs físicos. Nesses enxames de robôs são exploradas as características emergentes do sistema, produto das interações entre indivíduos entre si e

com o entorno, como observado na natureza em que entidades sem capacidades cognitivas complexas conseguem realizar tarefas grupais por intermédio de comportamentos coletivos.

Comportamento coletivo é definido como a composição de ações individuais de robôs que compõem um sistema. O comportamento de um robô individual é definido como um conjunto de ações ou respostas do controlador do robô devido a vários estímulos (entradas) do sistema. Os comportamentos coletivos permitem resolver tarefas de relevância como as anteriormente citadas nesta seção. Muitas dessas tarefas podem requerer comportamentos com restrições de distâncias, definidos como comportamentos coletivos que se concentram em como organizar e distribuir robôs e objetos no cenário de trabalho [13].

Exemplos de comportamento de organização com restrições de distâncias são observados em agregados de robôs que se agrupam em regiões de interesse; formações de padrões e cadeias para manter certa distribuição e conectividade no cenário de trabalho; montagem e morfogênese para movimento em áreas de difícil acesso; agrupamento e montagem de objetos para coleção de objetos pesados de forma eficiente [13, 16]. Conforme estudado a partir da observação de animais sociais na natureza, os comportamentos de organização com restrições de distâncias sugerem características positivas em comparação a grupos não organizados, como sobrevivência (evitando predadores) e divisão eficiente do trabalho [17], entre outras. Diversos trabalhos no campo da robótica de enxame demonstraram que os comportamentos com restrições de distância emergem no nível coletivo para resolver tarefas de enxame, como apresentado, por exemplo, em Francesca et al.; Francesca et al.; Rubenstein, Cornejo e Nagpal; Hasselmann, Robert e Birattari[15, 18–20]. A ideia principal é encontrar o controlador para cada robô individual que produz comportamentos emergentes no nível do enxame que lida com tarefas coletivas. Comportamentos com restrições de distâncias podem ser classificados em *agregação de robôs*, *montagem e morfogênese de robôs*, *agrupamento e montagem de objetos*, *formação de cadeias* e *formações de padrões* [13, 16].

O comportamento coletivo de **agregação de robôs** permite que os robôs se reúnam em um local específico, sendo um comportamento primitivo de outros mais complexos. Várias pesquisas sobre agregação foram realizadas, em que os controladores de software são sintetizados, por exemplo, utilizando projeto manual com leis de atração-repulsão [21]. Por sua vez, agregação de robôs pode ser abordada por métodos automáticos de sintetize controladores, utilizando robótica evolutiva e rede neuronal [22], usando controladores reativos

simples e robótica evolutiva [23] ou abordado por outros métodos automáticos de síntese de controladores, tal como em [Francesca et al.\[18\]](#). O método automático proposto em [Francesca et al.\[18\]](#), é utilizado em diferentes tipos de tarefas com restrições de distância [19]. Enxames de robôs podem resolver tarefas por intermédio de outros comportamentos com restrição de distância, tais como **montagem e morfogênese de robôs**. Eles são um conjunto de comportamentos coletivos que produzem conexões físicas entre os robôs da equipe. Exemplos destes comportamentos podem ser observados em robôs homogêneos [24], assim como também heterogêneos [25]. Trabalhos de comportamentos com restrição de distâncias em **agrupamento e montagem de objetos** podem ser encontrados na literatura, como em [Beckers, Holland e Deneubourg](#); [Gauci et al.](#); [Petersen, Nagpal e Werfel](#)[23, 26, 27], entre outros.

Formações de padrões e cadeias foram estudadas extensivamente pela comunidade de robótica de enxames [13]. A **formação de cadeias** é um comportamento com restrições fortes de distância no qual um enxame de robôs forma trilhas que conectam diferentes lugares de interesse. Uma pesquisa relevante neste campo apresenta o surgimento do planejamento como comportamento coletivo quando o enxame realiza a sequência de diferentes tarefas na ordem correta [28]. O padrão macroscópico mostrado pelo enxame é uma cadeia de robôs [28]. **Formações de padrões** são comportamentos nos quais a posição relativa entre robôs é relevante para obter o padrão desejado. A interação entre os robôs produzida por esses comportamentos foi modelada com forças eletrostáticas [6], forças de atração-repulsão [21]. O sistema atinge a coesão entre os robôs para a cobertura de área, sem qualquer informação e comunicação global [6]. Outros autores que utilizam o conceito de campos potenciais artificiais para coordenar movimentos de grande número de robôs, como em [Spears et al.\[29\]](#). Observam-se diferentes tipos de formações, como o hexagonal e o quadrado, caracterizados pelas diferentes relações das distâncias entre os robôs. Os resultados são apresentados em simulação e com sete robôs físicos, demonstrando a robustez, auto-organização e escalabilidade do sistema [29]. Em trabalhos relacionados com formação, a função de Lennard-Jones [30] é combinada com a robótica evolutiva para sintetizar os controladores individuais dos robôs [5], por exemplo para modelar as interações de um enxame de satélites [4]. Finalmente, é apresentado um algoritmo implementado em 1.024 robôs, em que a interação dos robôs que executam comportamentos simples permite obter

formações de padrões complexos [15]. Os resultados mostram formações de vários padrões pré-definidos.

Apesar das aplicações do enxame de robôs e das características positivas inerentes a esses sistemas, existe um desafio relevante na síntese do controlador individual de cada robô. Esse desafio tem a ver com a dificuldade de produzir o comportamento emergente a partir desses comportamentos individuais dos robôs. O desafio é produzido pela interação complexa no enxame que torna árduo o projeto do controlador do robô [18]. Normalmente, o projeto de controlador para enxames de robôs é realizado manualmente por tentativa e erro. Este projeto é uma abordagem *ad hoc* fortemente dependente da tarefa, sendo uma abordagem indireta devido a dificuldade de decompor o comportamento coletivo do enxame em comportamentos individuais dos robôs [18, 31]. Uma alternativa ao projeto manual é a síntese do controlador individual de forma automática. No projeto automático a síntese do controlador de cada robô é definida como um problema de otimização em que a tarefa do enxame é expressa como uma função objetivo que guia um algoritmo de otimização que derivará o controlador individual do robô. A síntese automática, em especial a *offline* (realizada *a priori* da fase de execução), provou ser de grande valor para o projeto de controladores, à medida que aumenta a complexidade do ambiente e da tarefa a ser realizada [31].

Nas abordagens ditas automáticas, surge um problema de projeto que tem relação com a definição do poder de representação do método. O poder de representação [18] se refere à capacidade do método de representar de forma abstrata um controlador coletivo que resolva uma tarefa dada. O poder de representação pode ser definido por intermédio do modelo de referência e da estrutura do controlador de cada robô [18]. O modelo de referência do robô pode ser modificado pela utilização de um hardware diferente (sensor, atuador e/ou robô) e/ou software diferente (aquisição e escrita de outros dados do sensor e atuador, respectivamente). Em relação à estrutura do controlador, diferentes tipos de arquiteturas, topologias e as relações das entradas com as saídas produzem diferentes tipos de representação. Estas relações são equivalentes a definir, por exemplo, os módulos de comportamento no projeto automático modular de controladores, ou as entradas e saídas da rede neural, entre outros. Esta definição do poder representacional do método especifica a flexibilidade do método para ajustar as relações de interação entre os robôs e dos robôs com o ambiente no comportamento coletivo emergente. Por sua vez, o poder de representação define as tarefas

coletivas que são realizáveis com o enxame de robôs. É observado na bibliografia, no que diz respeito ao projeto automático de controladores, que as abordagens de robótica evolutiva com arquiteturas de redes neuronais utilizam um poder de representação mais alto que as abordagens modulares que utilizam controladores mais simples [18, 32].

Abordagens que utilizam um poder de representação limitado permitem simplificar o projeto de controlador, manter o desempenho do controlador quando transferidos a robôs físicos e mais previsibilidade do comportamento coletivo. Porém, tarefas que incluem comportamentos coletivos, e particularmente em nosso estudo, com restrições de distância em enxames, podem ser comprometidas pela definição dessas restrições nos controladores individuais dos robôs.

1.1 Descrição do problema

Os enxames de robôs são compostos por uma grande quantidade (dezena, centena ou milhar) de robôs de capacidades limitadas que devem cooperar para realizar tarefas complexas por intermédio de comportamentos coletivos. Essa cooperação tem o propósito de superar dificuldades do sistema para resolver tarefas complexas ou irrealizáveis pelos robôs de forma individual [12]. Como os robôs possuem controladores que são descentralizados, trabalham com informações locais do ambiente, comunicação limitada e ambientes não estruturados [33], o projeto de controladores dos robôs possui uma série de desafios para a obtenção do comportamento coletivo [18].

De forma geral, o projeto automático de controladores para enxames de robôs deve determinar o controlador para cada robô individual. O controlador individual do robô é a porção de software computacional que recebe informação de sensores, processa e produz atuação [31]. Este controlador é o que realiza o mapeamento dos sensores e atuadores e permite desenvolver um conjunto de tarefas coletivas com os outros integrantes do enxame. Tipicamente no projeto automático *offline*, o projeto avalia os controladores candidatos em simulador para logo selecionar aquele de melhor desempenho. Logo, esse controlador é levado a cada robô físico para executar o comportamento no cenário físico. A filosofia de projeto automático tem por objetivo guiar o projeto de controlador incluindo, por exemplo, restrições de distâncias nos controladores individuais dos robôs. Estas restrições de distâncias

permitem que exista convergência a uma solução desejável, obter mais previsibilidade do comportamento coletivo e manter o desempenho do controlador quando transferidos a robôs físicos [18, 34].

Comportamentos com restrições de distâncias para os robôs do enxame manter coesão, possuem relevância na resolução de tarefas coletivas. A coesão é definida como a interação entre robôs mantendo distâncias específicas. A coesão permite interações de robôs sem colisões, manter a conectividade desejada, manter padrões regulares que permite realizar uma grande quantidade de atividades (como cobertura, sensoriamento e atuação distribuídos) para tarefas coletivas tais como mapeamento, monitoramento, resgate de vítimas, remoção de material explosivo, exploração espacial e controle de satélites [35], dentre outros. Para que os robôs do enxame abordem as tarefas acima mencionadas, o método automático de síntese de controlador deve ser especializado. Especializar um método é o procedimento para definir um conjunto de características, para tornar o método capaz de abordar tarefas de enxame *a priori* conhecidas ou desconhecidas. Na especialização do método, é definido o poder de representação da abordagem que determina as tarefas realizáveis pelo enxame de robôs.

Como observado na bibliografia, embora comportamentos com restrições de distância emergem nos enxames de robôs, o controlador gerado automaticamente pelos métodos de projeto de controlador atuais, especificamente o método AUTOMODE e o método EVOSTICK [18, 19], não pode resolver problemas em que as tarefas requerem coesão entre robôs: as restrições de distâncias introduzidas, em ambos métodos, na definição dos controladores individuais não são suficientes para a obtenção de controladores relevantes nestas tarefas de coesão. Portanto, a combinação dos controladores individuais de enxames, quando instanciados nos robôs, não é capaz de resolver a tarefa coletiva.

1.2 Hipótese

Incluir um novo comportamento individual com restrições de distâncias no projeto automático permitirá produzir comportamentos coletivos de coesão no enxame de robôs.

1.3 Contribuição

Apresentamos duas implementações de métodos de projeto automático que incluem restrição de distâncias na arquitetura do controlador: a implementação denominada MATE, uma nova especialização do método AUTOMODE, trata situações em que a distribuição dos robôs no cenário e trabalho desempenha um papel importante na execução de tarefas robóticas de enxame. O método MATE estende o método CHOCOLATE, adicionando um novo módulo chamado *formar*, cujo comportamento leva os robôs a manter distâncias arbitrárias entre si. Apresentamos a abordagem EVOSPACE, uma implementação de uma rede neural com ajuste de parâmetros por intermédio de um algoritmo evolutivo. A abordagem EVOSPACE é similar à abordagem EVOSTICK [18] quanto à arquitetura, porém é incluída a restrição de distâncias entre os robôs na entrada da rede neural.

Por sua vez, apresentamos a comparação de dois métodos de projeto automático de controladores para enxames de robôs: O método AUTOMODE-CHOCOLATE e o método EVOSTICK. Os métodos são apresentados em um conjunto de seis tarefas previamente definidas na bibliografia [18,19]. Analisa-se quantitativamente o desempenho de cada método e o comportamento coletivo de cada método em cada missão. Ambos os métodos já foram previamente definidos na bibliografia [18,19], porém não foram comparados para um mesmo conjunto de tarefas.

Comparamos a abordagem MATE com os métodos de projeto automático CHOCOLATE e EVOSPACE. Os métodos foram avaliados em três tarefas nas quais o desempenho do enxame foi condicionado pela capacidade dos robôs de se organizarem no cenário de trabalho. Apresentamos os resultados de experimentos com um enxame de 20 robôs e-puck tanto na simulação quanto com robôs físicos.

1.4 Justificativa

A incorporação de restrições de distâncias, e especificamente a possibilidade de manter distâncias específicas entre robôs do enxame, possui uma série de características positivas que beneficiariam a síntese automática de controladores em enxames de robôs. Esta inclusão permite produzir comportamentos com restrições de distâncias, em que a coesão entre

robôs é relevante para resolver uma tarefa coletiva particular, tal como monitoramento, mapeamento e movimento coletivo de robôs. Por sua vez, a inclusão permite favorecer tarefas coletivas que requerem de comunicação, cobertura de regiões, formação de padrões, sensoriamento e atuação distribuída. A incorporação de restrições de distâncias na síntese automática de controladores em enxames de robôs permite que o projeto seja restrito a condições conhecidas para que o controlador produzido no ambiente simulado seja relevante. Por sua vez, a incorporação das restrições de distâncias permite resolver uma variedade mais ampla de problemas da robótica de enxame, consideradas de tarefas que são prioritárias em aplicações reais e que anteriormente eram resolvidas parcialmente por métodos automáticos de síntese. Permite que o método seja mais geral no que diz respeito às tarefas que podem ser realizáveis por ele.

Assim, quando a tarefa coletiva requerer, esta inclusão permite que a síntese do controlador seja restrita no conjunto de controladores com ênfase em comportamentos que devem manter certa distribuição no cenário de trabalho.

1.5 Objetivos

Objetivo geral:

O objetivo principal desta tese é projetar e avaliar leis de controle/coordenação em enxames de robôs concebidas para a execução de comportamentos coletivos com restrições de distâncias.

Objetivos específicos:

- Definir o método de síntese do controlador.
- Definir a arquitetura de controle de cada robô.
- Especificar cenário de teste e características dos robôs.
- Elaborar os módulos de comportamento para cada robô.
- Verificar os comportamentos individuais em cada robô.

- Avaliar os comportamentos coletivos no enxame de robôs em simulação.
- Implementar as leis de controle/coordenação nos robôs físicos.
- Validar os comportamentos coletivos nos robôs físicos.

Capítulo 2

Revisão bibliográfica

A revisão bibliográfica presente neste capítulo está centrada no projeto de controlador para robôs de um enxame de robôs. Normalmente, e com algumas exceções, os sistemas considerados nesta revisão utilizam a descrição de enxame de robôs realizada em [Şahin\[12\]](#). As exceções podem ser feitas pela relevância do trabalho e/ou por outras características do sistema que contribuem no estado da arte da robótica de enxame. Assim, são considerados [12]:

- Sistemas compostos por entidades simples no que diz respeito ao hardware e controlador.
- Os robôs são autônomos, possuem controladores individuais e não existem líderes no enxame.
- Sistemas de robôs que utilizam informações locais do ambiente de trabalho, comunicação limitada e sem utilizar posição absoluta dos robôs.
- O sistema é redundante. Serão considerados principalmente, sistemas com robôs homogêneos quanto as suas capacidades.
- O sistema é flexível para se adaptar a diferentes cenários e tipos de tarefas.

Pelas características anteriores o sistema será escalável, no que diz respeito à quantidade de robôs que compõem o enxame. Além disso, o sistema possuirá robustez no que se refere à falha de alguns dos robôs e/ou hardware [13,16]. Nos trabalhos relevantes citados nesta seção, serão abordados casos de estudo que utilizam projeto manual e automático. Por sua

vez, são estudadas principalmente abordagens que resolvem tarefas coletivas com restrições de distâncias e, especificamente, aquelas tarefas que requerem a coesão entre os robôs do enxame os quais devem manter uma distribuição específica no cenário de trabalho.

Em relação ao projeto automático de controladores, a revisão está centrada em técnicas *offline*. Técnicas *offline* são aquelas que o projeto de controlador é feito *a priori* de ser instanciados nos robôs físicos para desenvolver, de fato, a tarefa coletiva. Normalmente, o projeto é feito em simulação, na qual são avaliados um conjunto de controladores candidatos. Contrariamente as técnicas *offline*, as técnicas *online* são aquelas que o projeto de controlador é feito nos robôs (físicos ou simulados) simultaneamente com a resolução da tarefa. A seguir é realizada a classificação dos trabalhos entre o projeto automático e o projeto manual de controladores para robôs de um enxame.

A seguir é realizada a classificação dos trabalhos segundo o tipo de projeto e arquitetura utilizada.

2.1 Projeto manual de controladores para robô de enxames.

Normalmente, o projeto de controlador para enxames de robôs é realizado manualmente. O projeto manual é caracterizado pela definição do tipo de arquitetura de controlador e ajuste de parâmetros de forma manual e iterativa até que o comportamento coletivo é produzido [19]. Geralmente, a construção de controladores para robôs de um enxame utiliza a abordagem baseada em comportamento [36], a abordagem de campos potenciais artificiais [37, 38] ou utilizando a teoria de controle supervisorio [39]. A classificação feita nesta seção considera como abordagens baseadas em comportamento aquelas abordagens manuais que possuem um alto grau de modularidade no qual o robô é controlado por regras simples e não necessariamente campos potenciais. As abordagens de campos potenciais artificiais são consideradas seguindo a abordagem de [38], que é uma abordagem aplicada em várias situações, e considera unicamente campos potenciais locais a cada robô, diferentemente de trabalhos como em Khatib[37].

As limitações do projeto manual são que o projeto de controlador é realizado por tentativa

e erro. Este projeto manual confia-se totalmente na intuição e habilidade do projetista [19]. Assim, o projeto manual não fornece diretrizes gerais para que as especificações do nível de exame sejam atendidas uma vez especificado o comportamento individual de cada robô [18]. Por ser um projeto considerado *ad hoc*, ele é específico para a tarefa colectiva que o controlador foi projetado, não permitindo abordar outros tipos de tarefas colectivas: No caso de comportamentos com restrições de distâncias, estes comportamentos produzidos são específicos para a tarefa a ser abordada pelo exame. Na Tabela 2.1 se observam as principais limitações dos métodos de projeto manual. Os trabalhos apresentados na tabela serão explicados a seguir.

Tabela 2.1: Limitações dos métodos de projeto manual apresentados na bibliografia.

Projeto manual		
Arquitetura	Limitações	Trabalhos realizados
Baseada em comportamentos	- Dificuldade de obter o comportamento coletivo a partir dos comportamentos individuais. - Comportamentos com restrições de distâncias são específicos para as tarefas que o sistema foi projetado.	[33, 36, 40]
Teoria de campos potenciais		[6, 21, 29, 38, 41–49]
Teoria de controle supervisorio		[14, 50, 51]

Fonte: Elaborada pelo autor.

2.1.1 Teoria de controle supervisorio

A teoria de controle supervisorio [39] é uma abordagem para síntese de controladores baseada em máquinas de estados finita e linguagens formais. Na abordagem são modeladas as capacidades dos robôs e as especificações do sistema. As capacidades dos robôs abstraem todas as restrições físicas dos robôs definidas pelo hardware do robô. Esta definição produz o comportamento livre de cada robô. As restrições desses recursos são modeladas por meio das especificações de controle. Essa teoria combina os comportamentos livres e especificações no supervisor de robô para gerar o controlador, o que garante que apenas eventos válidos em cada robô aconteçam.

Uma abordagem utilizando a teoria de controle supervisorio foi definida em Lopes et al. [50]. Foram sintetizados controladores para o robô e-puck [52] e Kilobot [53]. Foram

sintetizados controladores monolíticos e modulares para duas tarefas, uma de segregação e outra de orbita ao redor de um robô estacionado. Para obter o supervisor para cada caso de estudo, foram definidas a modelagem dos comportamentos livres da plataforma e a modelagem das especificações desejadas. Logo, foram combinadas usando a composição síncrona dos autômatos e logo projetado o supervisor que garante a especificação ser cumprida evitando, por exemplo, os estados que potencialmente podem ativar eventos não controláveis. Resultados são obtidos para o e-puck (Kilobot) utilizando 2 (2) robôs físicos na tarefa de orbita e 26 (42) para a tarefa de segregação, respectivamente.

Em um trabalho recente é apresentada a aplicação da teoria de controle supervisorio para enxames de robôs [14]. É realizada a síntese de controladores para os robôs individuais e é apresentada uma forma automática de geração de código para esse controlador. Quatro estudos de caso são apresentados utilizando as plataformas de robôs e-puck e Kilobot. São estudadas tarefas de formação em grupos de robôs, segregação, agregação e agrupação de objetos. Em alguns dos experimentos do artigo é utilizado o mesmo controlador sintetizado nas duas plataformas. Num trabalho subsequente [51], os comportamentos livres do sistema e as especificações são modelados como máquinas de estados probabilísticas. Experimentos com robôs físicos foram realizados utilizando 1.000 Kilobot. Da mesma forma que no trabalho anterior, foi mostrado que a abordagem modular de síntese é mais apropriada de ser implementada em enxames de robôs que a abordagem monolítica.

2.1.2 Abordagem baseadas em comportamento

Uma das primeiras abordagens para sintetizar controladores de software para robôs de enxame foi a abordagem baseada em comportamento [36]. Nesta abordagem reativa são projetados inicialmente os comportamentos primários da plataforma. Cada comportamento relaciona informações de sensores a atuadores utilizando regras simples e esses comportamentos são organizados em camadas hierárquicas, nas quais camadas inferiores podem suprimir ou inibir camadas mais altas [36]. Esta abordagem e sua aplicação em trabalhos sucessivos como [33, 40] foram os primeiros passos do que hoje é conhecido como robótica de enxame.

A pesquisa feita em [Matarić\[33\]](#) apresenta um dos primeiros trabalhos sobre comportamento coletivo em enxames de robôs. São definidos comportamentos primários vetoriais que

são compostos por intermédio de composições temporais para produzir a nível de enxame comportamentos tais como movimento coletivo e agrupação de objetos. Os comportamentos primários em cada robô são combinados utilizando composições temporais e soma vetorial [33]. Esses comportamentos básicos são escolhidos empiricamente observando as restrições dinâmicas do robô e as restrições impostas pela tarefa. Os robôs são considerados homogêneos em hardware e software, sem comunicação direta entre entidades, com controladores descentralizados e sensoriamento local do ambiente. Os resultados são apresentados em ambiente simulado e com robôs físicos, mostrando o desempenho de cada comportamento. Logo, foi realizada a comparação com grupos heterogêneos e com técnicas centralizadas de coordenação.

Os autores [41] apresentam um sistema baseado em comportamentos em que a composição de comportamentos reativos permite realizar a navegação do time orientada a alvos, evitando obstáculos e colisões com outros robôs. Cada robô possui um identificador e podem ser heterogêneos na sua funcionalidade. A composição de comportamentos foi implementada como a soma dos vetores de cada um dos comportamentos sendo estes implementados em simulação e em robôs físicos. Os resultados apresentam os distintos tipos de formação atingidos tanto em simulação como para uma quantidade limitada de robôs reais.

Uma arquitetura baseada em comportamentos é realizada em Parker[40]. A arquitetura é descentralizada pensada para robôs heterogêneos composta de diversos comportamentos que são ativados por intermédio de comportamentos motivacionais que permitem adaptabilidade. Estes comportamentos monitoram o progresso da tarefa realizada por outros robôs e pelo próprio robô, permitindo adaptabilidade em caso de falha própria ou dos outros robôs. Resultados foram apresentados em robôs físicos mostrando a robustez, adaptabilidade a diversas situações para a tarefa de coleta de objetos. A limitada escalabilidade do método é a principal desvantagem.

2.1.3 Abordagem de campos potenciais artificiais

A abordagem de campos potenciais artificiais é uma abordagem para a síntese do controlador individual do robô. Esta abordagem começou a ser utilizada primeiramente em Khatib[37] para aplicações com um único robô. Logo, a abordagem começou a ser utilizada

em trabalhos com enxames de robôs. Exemplos podem ser encontrados em [Spears et al.](#); [Spears e Gordon](#)[29, 38]. Atualmente a abordagem é amplamente utilizada na comunidade de robótica de enxame. A abordagem de campos potenciais artificiais considera cada robô como uma partícula que sensoreia e reage a forças virtuais produzidas por robôs vizinhos. Cada robô computa a força virtual que está sendo colocada nele e esta força é a que guia os atuadores do robô. Obstáculos produzem forças de repulsão nos robôs e objetivos produzem forças de atração. Como estas forças derivam de campos potenciais, as forças levam os robôs a configurações desejadas nas quais a energia do sistema é minimizada. Isto é equivalente a dizer que os robôs se mantêm em posições de equilíbrio sem movimento. A abordagem permite produzir comportamentos emergentes no enxame nos quais as restrições de distâncias permitem que os robôs mantenham uma organização com coesão no ambiente de trabalho.

Uma abordagem que utiliza campos potenciais artificiais para modelar as interações entre robôs é apresentada em [Spears e Gordon](#)[38]. Os autores apresentam um trabalho relevante inspirado nas leis da física que permite coordenar comportamentos de grandes quantidades de robôs de forma descentralizada, robusta e auto-organizada. A abordagem é independente da plataforma robótica usada e permite realizar movimento coletivo coordenado evitando obstáculos, mantendo a coesão e atingindo objetivos. No trabalho podem ser observados distintos tipos de formações de padrões, tais como malhas hexagonais e quadradas. Por sua vez, podem ser observadas formações que formam padrões globais, tal como um hexágono ou um quadrado com vários robôs. Logo, esta abordagem é novamente apresentada em [Spears et al.](#)[29], em que são mostradas também transições entre as formações e a geração de padrões de formação em três dimensões. A abordagem é utilizada também para realização e tarefas tais como monitoramento e vigilância de perímetro. Resultados são apresentados em simulação e com sete robôs físicos, demonstrando a robustez, auto-organização e escalabilidade do sistema. Este trabalho serviu de fonte inspiradora de várias outras pesquisas na área, que aprimoraram a abordagem e expandiram a abordagem para realização de outros tipos de tarefas.

Os mesmos autores abordaram diferentes tipos de formações, tais como formações sólidas, líquidas e gasosas [44]. Este tipo de formações é obtido pela modificação da ganância da lei utilizada no caso das formações sólidas e líquidas, ou a utilização de outra lei de controle, como no caso gasoso. São apresentados resultados destas formações em simulação,

para uma formação hexagonal rígida (solida), para uma formação mais flexível (liquida) evitando obstáculos e para outra lei puramente repulsiva (gasosa) realizando cobertura com monitoramento. Outro trabalho na bibliografia que utiliza a abordagem de campos potenciais artificiais foi apresentado em [Howard, Matarić e Sukhatme](#)[6]. A interação entre os robôs foi definida utilizando a lei eletrostática de partículas eletricamente carregadas. O sistema atinge coesão entre os robôs para realizar tarefas coletivas de cobertura de corredores, sem nenhum tipo de informação global ou comunicação entre os robôs. Diferentemente aos trabalhos anteriores citados em campos potenciais artificiais que baseiam as leis de controle com analogias a fenômenos da física, outros trabalhos utilizam diferentes tipos de funções, por exemplo, utilizando interações proporcionais distância entre os robôs, como em [Kazadi; Shucker e Bennett; Shucker, Murphey e Bennett; Hsu e Liu](#)[43, 45, 47, 48].

Uma abordagem utilizando campos potenciais artificiais para a obtenção de padrões globais é realizado em [Kazadi](#)[48]. Neste trabalho é obtido um único padrão global hexagonal como em [Spears e Gordon](#)[38], mas utilizando uma lei de força que é proporcional à distância relativa entre os robôs. O projeto define uma especificação no nível de enxame que é a de minimizar a energia global do sistema que assegura atingir o padrão global. A partir dessa especificação os robôs produzem o comportamento desejado. Resultados são apresentados apenas em simulação. De igual forma que em [Kazadi; Balch e Arkin](#)[41, 48], em [Shucker e Bennett; Shucker, Murphey e Bennett](#)[43, 47] utilizam a abordagem de campos potenciais. A modelagem das interações entre cada par de robôs foi realizada como um sistema massa-mola. A abordagem é usada em tarefas de cobertura e no seguimento de alvo. A abordagem permite o ajuste de distâncias entre robôs e permite produzir vários padrões com coesão.

Outros trabalhos que modelam as interações entre robôs como campos potenciais artificiais utilizam interações proporcionais ao quadrado da distância para dar ênfase aos robôs mais distantes. Trabalhos que utilizam estas interações podem ser encontrados em [Reif e Wang; Turgut et al.; Gazi e Passino; Mendiburu, Morais e Lima](#)[21, 42, 46, 49]. Uma lei de controle para o movimento coletivo de robôs foi proposta em [Turgut et al.](#)[46], baseado na teoria de campos potenciais artificiais. O movimento coletivo é composto pelos comportamentos de alinhamento de guinada e controle de proximidade de cada robô. Estes comportamentos permitem atingir o movimento coeso do time. Foram realizados testes com robôs físicos, o robô Kobot [54], avaliando variantes dos comportamentos mediante ajuste de parâmetros e

analisando o desempenho do sistema mediante indicadores que permitem observar o grau de alinhamento e a entropia. Os testes foram divididos em três fases demonstrando que o melhor controlador é aquele com controle de velocidade, de proximidade e alinhamento de guinada ativados e ajustados corretamente. No mesmo trabalho é apresentado um sistema local de comunicação sem fio para a transmissão e recepção de ângulos de orientação entre vizinhos onde cada robô possui este sistema mais uma bússola para a definição de um sistema de orientação global.

2.2 Projeto automático de controladores para robô de enxames.

Como foi mencionado no capítulo anterior, o projeto automático de controladores para enxames de robôs possui o objetivo principal de sintetizar controladores para a solução de tarefas coletivas. Este tipo de projeto é promissor em sistemas com vários robôs que devem interagir em ambientes complexos em que não é possível modelar adequadamente o sistema como um todo e/ou descrever a tarefa individual de cada robô a partir da tarefa coletiva [31]. A aplicação de técnicas de projeto automático permite que o usuário por intermédio do conhecimento da tarefa coletiva, projete controladores para cada um dos robôs do enxame. Uma das dificuldades dentro do projeto automático de controladores é a especialização do método. A especialização do método é o procedimento que permite definir um conjunto de características do método antes da síntese do controlador. A especialização permite definir o poder de representação do método. O poder de representação define a flexibilidade do método para ajustar as relações de interação entre os robôs e dos robôs com o ambiente. O poder de representação do método define a complexidade do controlador e define que tipos de tarefas coletivas são realizáveis pelos robôs do enxame.

Várias abordagens estudam a modificação da complexidade do controlador em problemas de projeto automático para tratar as discrepâncias entre desempenhos do controlador em simulação e com robôs reais [18,34], abordar a previsibilidade do comportamento coletivo [32] e simplificação do projeto de controlador [18,34], entre outros. A alteração da complexidade do controlador pode ser obtida modificando o tipo de arquitetura do mesmo. A seguir

Tabela 2.2: Limitações dos métodos de projeto automático apresentados na bibliografia.

Projeto automático		
Arquitetura	Limitações	Trabalhos realizados
Monolítica	<ul style="list-style-type: none"> - Respeito a tarefas coletivas complexas. - Aborda tarefas relativamente simples e específicas. - Utilizado para ajustes de parâmetros unicamente. 	[22, 23, 55–58]
Modular	- Especialização do método é dependente da tarefa.	[34, 59–63]
	- Utilizado para ajustes de parâmetros unicamente.	[4, 5, 64, 65]
	<ul style="list-style-type: none"> - Especialização do método limita as tarefas coletivas possíveis. - Não considera a possibilidade dos robôs manterem distâncias específicas entre si. 	[18–20, 66, 67]

Fonte: Elaborada pelo autor.

são divididos os trabalhos relevantes do estado da arte em **arquiteturas modulares** e **arquiteturas monolíticas**. Na Tabela 2.2 se apresentam as principais limitações das abordagens estudadas na bibliografia.

2.2.1 Arquiteturas modulares

As arquiteturas modulares são estruturas de controle que permitem representar as partes do sistema como componentes, que podem ser criadas, reutilizadas e combinadas de diferentes formas. O controlador gerado usando esta técnica é um sistema complexo de partes simples que interagem de forma desacoplada. A utilização destas arquiteturas permite desdobrar o problema em partes. Cada parte pode ser abordada de forma independente. Assim, são definidos vários controladores para cada robô que na sua composição com os comportamentos de robôs e as relações com o ambiente é produzido o comportamento emergente do sistema. As arquiteturas modulares permitem a escalabilidade das tarefas mais complexas [34].

Vários trabalhos na bibliografia abordaram o uso de arquiteturas modulares. Uma arqui-

tetura hierárquica modular para a síntese de controladores foi estudada em [Duarte, Oliveira e Christensen](#); [Silva et al.](#)[34,59]. O controlador do robô é composto de comportamentos primitivos simples e mecanismos de arbitragem para a seleção do comportamento. Ambos módulos podem ser sintetizados usando robótica evolutiva ou ser pré-programados. Tarefas complexas são divididas em subtarefas e o processo de evolução aborda cada uma separadamente quando a evolução é incapaz de sintetizar o controlador monolítico devido à complexidade da tarefa. O controlador resultante é uma composição hierárquica dos comportamentos anteriormente projetados. Os resultados foram apresentados em um único robô em simulação e com robô físico para tarefas de resgate em corredores, tarefas de limpeza e tarefas de fototaxis, em que *a priori* não foi possível encontrar o controlador monolítico [59]. Logo foi apresentada a mesma arquitetura modular para abordar tarefas em sistemas multi-robô [60]. Foram realizados experimentos em simulação com três robôs que deviam resolver a tarefa de coleta de objetos entre duas localizações separadas por um corredor bloqueado. Da mesma forma que em [Silva et al.](#)[59], a síntese automática do controlador usando arquiteturas monolíticas não conseguiu resolver a tarefa. Porém, a arquitetura hierárquica modular conseguiu abordar o problema projetando controladores para cada subtarefa por intermédio da decomposição da tarefa.

Um trabalho de robótica evolutiva utilizando controladores modulares para a síntese do controlador de robô em enxames de robôs foi realizado em [Duarte et al.](#)[63]. A tarefa a ser resolvida foi a de detecção de intrusos com restrições de bateria. A tarefa foi decomposta e controladores para cada subtarefa individual foram realizados. A arquitetura de controlador escolhida foi a arquitetura hierárquica modular [34], sintetizando tanto a topologia da arquitetura como os parâmetros da rede [55]. A tarefa composta foi resolvida combinando os comportamentos projetados *a priori* usando uma máquina de estado determinística, sintetizada manualmente. Este árbitro escolhe que comportamento é ativado dependendo do estado do robô e dos sensores. Várias tarefas coletivas foram apresentadas modificando a relação entre os estados da máquina de estados, reutilizando os mesmos comportamentos anteriormente projetados. Foram realizados testes de escalabilidade no que diz respeito a quantidade de robôs no enxame. Finalmente foram obtidos resultados em seis robôs físicos obtendo desempenhos similares com os da simulação. Em um trabalho similar ao anterior apresentado em [Duarte, Oliveira e Christensen](#)[61], o projeto de controlador para enxames

de robôs para a tarefa de detecção de intrusos foi realizado, apresentando a escalabilidade do sistema desta vez com 1.000 robôs. Resultados são apresentados unicamente em simulação.

Em um trabalho dos mesmos autores que utilizou os mesmos robôs que em [Duarte et al.\[63\]](#), a síntese automática modular de controladores para enxames de robôs foi realizada [\[62\]](#). Os experimentos foram feitos em robôs aquáticos usando redes neurais em que tanto os parâmetros como a topologia da rede neural é ajustada no projeto de controlador [\[55\]](#). Os controladores foram projetados em simulação e validados em robôs físicos em quatro tarefas isoladas e para uma tarefa mais complexa que inclui as outras como subtarefas. A tarefa composta é a de monitoramento de área, com várias tarefas com restrições de distâncias, como por exemplo, de agregação e dispersão de robôs. Neste trabalho foi utilizado um coordenador simples sequencial para produzir a tarefa coletiva. Resultados apresentados mostram bons desempenhos dos controladores tanto na simulação como nos robôs físicos, apresentando um sistema de robôs escalável e um sistema robusto no que diz respeito às condições do ambiente. Como conclusão dos artigos anteriores, pode se dizer que uma das principais desvantagens do método é o método depende da tarefa. Nos artigos são abordados, normalmente, tarefas que são facilmente divisíveis, não dando indicações sobre como deve ser feita essa subdivisão para outros tipos de tarefas mais complexas.

Uma arquitetura modular para o controle de formação em enxame foi apresentado em [Pinciroli et al.\[68\]](#). São evoluídos controladores que utilizam leis de campos potenciais artificiais (Seção 2.1.3). O sistema se caracteriza por um componente de localização global para os robôs saberem onde deve ser feita a formação e outra local, utilizando a função de Lennard-Jones [\[30\]](#) que permite a interação com o meio externo. Esse método em trabalho sucessivo foi evoluído em simulação com 10 satélites utilizando um algoritmo genético que ajusta o valor de quatro parâmetros, utilizando 1.000 gerações com uma população inicial de 50 candidatos e utilização de mecanismos de elitismo, mutação e cruzamento [\[4\]](#). Resultados em simulações com até 500 satélites apresentam a escalabilidade do sistema no que diz respeito a quantidade de satélites [\[4\]](#).

Uma arquitetura modular foi apresentada para enxames de robôs em [Hettiarachchi e Spears\[5\]](#). A abordagem utiliza robótica evolutiva para sintetizar os controladores dos robôs do enxame. Neste trabalho são evoluídos controladores que utilizam leis de campos potenciais artificiais (Seção 2.1.3). Como as diferenças de desempenho entre a simulação e os robôs

físicos são mais notórias nas abordagens de robótica evolutiva, os autores propõem um processo de otimização *offline-online* híbrido [65]. Este processo tem em conta cenários complexos e tenta evitar as diferenças entre os desempenhos obtidos na simulação e com os robôs físicos. Eles desenvolvem dois controladores usando a função gravitacional e a função de Lennard-Jones. Os parâmetros desses dois controladores são ajustados para encontrar o melhor controlador para navegação em um ambiente desestruturado. Uma função objetivo foi apresentada e tem em conta as colisões dos robôs, a coesão e o sucesso na navegação. Eles mostram que a função de Lennard-Jones possui um desempenho significativamente melhor que a força gravitacional na tarefa de navegação desestruturada, pois a coesão e o sucesso para atingir o objetivo são alcançados independentemente do ambiente desestruturado no qual os experimentos foram realizados. Além disso, a função de Lennard-Jones apresentou baixos níveis de colisão com obstáculos, atingindo o objetivo em tempos razoáveis e com alta conectividade [64].

Uma abordagem promissora para o projeto de controladores em enxames de robôs é apresentada em [Francesca et al.](#); [Francesca](#)[18, 69] chamada de AUTOMODE, acrônimo de projeto automático modular. O método AUTOMODE é uma abordagem *offline* de projeto modular para produzir, de forma automática, o controlador de cada robô para abordar problemas de robótica de enxame. O método AUTOMODE é independente da plataforma de robô e geral, no que diz respeito as tarefas coletivas que podem ser abordadas. Os autores afirmam que reduzir o poder de representação do controlador permite que as diferenças de desempenho entre os robôs simulados e os robôs físicos sejam reduzidas, ao contrário de outras abordagens de robótica evolutiva, que enfrentam dificuldades neste ponto. A especialização de AUTOMODE-VANILLA [18] utiliza máquinas de estados probabilísticas finitas [70], robôs e-puck [52], utiliza o simulador ARGoS [71, 72] e o algoritmo de otimização F-Race [73]. Esta máquina de estados é composta por comportamentos simples (estados) e condições (transições), ambos chamados de módulos. As condições são usadas para produzir as transições entre os comportamentos em resposta a um acontecimento particular. Condições e comportamentos podem ser ajustados por intermédio de um conjunto de parâmetros e pode ser instanciados várias vezes no mesmo controlador, reutilizando os módulos para atender as diferentes situações. O algoritmo F-Race é utilizado para ajustar os módulos (condições, transições e parâmetros) da máquina de estados finitos probabilística. Os resultados foram

mostrados em simulação e com robôs reais em tarefas com restrições de distância de agregação e coleta de objetos. A abordagem foi comparada com um algoritmo de robótica evolutiva padrão que é utilizado frequentemente em outras pesquisas. O desempenho do método VANILLA foi superior ao desempenho do EVOSTICK. O desempenho dos controladores do método AUTOMODE foi comparável em robôs reais e simulação para ambas tarefas, diferente do método EVOSTICK, que apresenta melhor desempenho na simulação com relação a VANILLA, mas o desempenho cai significativamente em experimentos com robôs reais.

Uma nova especialização do método AUTOMODE foi apresentada em [Francesca et al.\[19\]](#), chamada de AUTOMODE-CHOCOLATE. Nesta especialização é utilizado um algoritmo de otimização denominado F-Race iterado [74, 75]. O objetivo da pesquisa foi mostrar que o projeto automático sintetiza controladores de melhor desempenho que os produzidos por operadores humanos. Este experimento foi realizado em cinco tipos de tarefas de enxame diferentes e com vários métodos de projeto: AUTOMODE-CHOCOLATE, AUTOMODE-VANILLA, EVOSTICK e dois projetos manuais: operador restrito e operador não restrito. O operador restrito deve realizar o projeto do controlador usando unicamente os módulos definidos no método AUTOMODE e ajustar os parâmetros manualmente. O operador não restrito deve realizar o projeto de controlador de forma livre. Resultados apresentados mostram que o método AUTOMODE-CHOCOLATE apresentou melhores resultados que os outros métodos devido ao algoritmo de otimização incluído nele. Por sua vez, o operador restrito teve melhor desempenho que o operador não restrito, mostrando a generalidade dos módulos do método AUTOMODE para resolver tarefas.

Uma nova especialização do método AUTOMODE foi realizada em [Kuckling et al.\[67\]](#), chamada de MAPLE, como uma alternativa para a arquitetura de máquina de estados propostas nas instâncias do método AUTOMODE acima mencionadas. Esta especialização explora a viabilidade de usar uma arquitetura de árvore de comportamento para montar módulos preexistentes junto com F-Race iterado para ajustar e combinar módulos preexistentes. Por motivos de comparação, as escolhas de projeto feitas no método MAPLE são as mesmas que no método CHOCOLATE, como o mesmo algoritmo de otimização, módulos preexistentes, robô e simulador. Eles mostram resultados comparáveis com o método CHOCOLATE e resultados significativamente melhores que os resultados do EVOSTICK em duas tarefas diferentes. Ambas as instâncias do método AUTOMODE funcionam de forma semelhante

em simulações e robôs físicos, diferentemente do método EVOSTICK, que apresenta melhor desempenho na simulação que na realidade. Como a especialização de método MAPLE utiliza os mesmos blocos que as versões anteriormente mencionadas do método AUTOMODE, possui a mesma limitação para resolver tarefas não contempladas na especialização.

Uma das limitações encontradas nas especializações de AUTOMODE previamente realizadas, foi referente as dificuldades do método de obter controladores para tarefas que requerem comunicação. Assim, uma nova especialização de AUTOMODE é feita, chamada de GIANDUJA [20]. AUTOMODE-GIANDUJA permite sintetizar controladores modulares com a capacidade de comunicação local entre robôs. Assim, a semântica da mensagem é definida pelo projeto automático. O método estende as capacidades da abordagem CHOCOLATE, pois o algoritmo de otimização define também a semântica das mensagens de transmissão e como o receptor tem que reagir a essas mensagens. O método foi estudado em três missões diferentes e foi comparado com os métodos CHOCOLATE e EVOCOM. O método EVOCOM é similar ao EVOSTICK, mas com a capacidade de comunicação. Os resultados mostram que o método GIANDUJA apresenta um desempenho significativamente melhor do que os outros dois métodos e utiliza de forma significativa a comunicação nas tarefas consideradas. Dos resultados e da análise qualitativa dos comportamentos é concluído que a comunicação não emergiu da especialização do método CHOCOLATE, mostrando que a especialização do método CHOCOLATE não é apropriada para abordar tarefas que requerem comunicação.

Um trabalho de robótica evolutiva para o ajuste de uma arquitetura modular em forma de máquina de estados foi proposta em König, Mostaghim e Schmeck[66]. A forma de evolução é feita *online* e descentralizada, tendo robôs com diferentes controladores em cada experimento. Operadores de elitismo, mutação e recombinação são aplicados na máquina de estados. A recombinação e elitismo permitem a proteção de partes do genótipo que estão relacionados com comportamentos promissores, permitindo que o projeto automático evolua em condições restritas que favorecem soluções promissoras e, por sua vez, restringe o espaço de busca de controladores. As transições entre estados são simples comparações entre valores medidos dos sensores de proximidade do robô entre si ou com constantes. Por sua vez, os estados representam ações que o robô pode desenvolver, tais como ir para a frente, virar a direita ou esquerda, parar e permanecer executando a ação anterior. A maioria desses estados possui um parâmetro a ser ajustado. A evolução de controladores é estudada em

tarefas coletivas como evitar colisões e passagem de robôs por espaços restritos. Resultados foram obtidos para 26 robôs unicamente em simulação. A desvantagem do método é referente a que são obtidos comportamentos bastantes dispares entre os robôs, tendo em conta que o ambiente onde os comportamentos foram evoluídos foi o mesmo para todos os robôs.

2.2.2 Arquiteturas monolíticas

As arquiteturas monolíticas são estruturas de controle que representam o controlador como uma única componente complexa. Na definição utilizada nesta tese, os controladores classificados nesta categoria possuem alto acoplamento ou não permitem *a priori* uma modularidade na estrutura.

Um dos trabalhos mais relevantes na evolução de controladores, muito utilizada na robótica evolutiva é apresentado em [Stanley e Miikkulainen\[55\]](#). No trabalho é apresentado um novo método para projetar controladores utilizando rede neuronal e um algoritmo evolutivo, em que a topologia da rede e os ganhos são ajustados para obter o melhor controlador de acordo com a tarefa específica. O ajuste automático tanto a topologia da rede como os ganhos permite economizar tempo de projeto. O método evolui a arquitetura a partir de estruturas simplificadas, como redes neuronais sem nós ocultos, para no decorrer do método ir aumentando a complexidade do controlador, diferentemente de outros métodos que realizam a evolução com estruturas complexas. Os resultados apresentam as melhorias do ajuste simultâneo da arquitetura e os parâmetros em relação aos métodos tradicionais de evolução.

Outro trabalho relevante que trata o comportamento coletivo com restrições de distância é abordado [\[22\]](#). É proposta a síntese de um controlador para a tarefa de agregar robôs, similar ao comportamento observado em baratas. O controlador foi definido como uma rede neuronal monolítica sem nós ocultos e com todas as entradas conectadas nas saídas. É utilizado um algoritmo evolutivo para definir os parâmetros da rede neural sobre uma população de 100 redes neuronais que são evoluídas sobre 200 gerações. A população da primeira geração é gerada aleatoriamente e as gerações subseqüentes são criadas usando um processo de seleção e reprodução que envolve elitismo e mutação. Para avaliar o desempenho dos controladores nas gerações é utilizada uma função objetivo que maximiza a quantidade

de robôs numa área de agregação específica de duas possíveis. Resultados em simulação mostram que os controladores obtidos conseguem desenvolver a tarefa proposta e que os comportamentos obtidos são similares [22].

Uma abordagem evolutiva utilizando controladores monolíticos para a solução do problema de agregar robôs é analisado [57]. A abordagem sintetiza controladores para robôs s-bots [76] utilizando uma arquitetura de rede neuronal de 12 entradas e 2 saídas que codifica no genótipo os 24 parâmetros a serem ajustados pelo algoritmo evolutivo. Foi definida uma função objetivo que tem em conta a qualidade da agregação e do movimento [57]. Finalmente um estudo de movimento coletivo foi realizado. Análise de escalabilidade referente à quantidade de robôs no enxame são feitos para o estudo de agregação e do movimento coletivo. Resultados são obtidos em simulação.

Em outro artigo que utiliza uma arquitetura monolítica foram sintetizados controladores para robôs simples com o propósito de realizar comportamentos de formação, em ambientes sem obstáculos [56]. Foram utilizados três robôs de tração diferencial equipados unicamente com sensores de proximidade. Foi utilizado um controlador em forma de rede neuronal. Foram avaliadas, durante a evolução dos controladores, a capacidade dos robôs se manter com conectividade, em uma forma predefinida, e se deslocar certa distância em conjunto. Os controladores foram evoluídos em simulação e logo levado para os robôs físicos. Resultados apresentados mostram que os robôs mantêm conectividade no deslocamento. Porém a orientação dos robôs vai mudando durante o percurso do experimento e o deslocamento é a baixa velocidade.

As arquiteturas monolíticas não estão somente restritas a redes neurais. Assim, uma abordagem automática com controlador minimalista foi apresentada em [Gauci et al.\[58\]](#) para a síntese de controladores de um enxame de robôs. A tarefa coletiva a ser desenvolvida foi a de tarefa de agrupamento de objetos. Neste trabalho os autores mostram como é possível utilizar robôs com recursos limitados para resolver a tarefa, mesmo que os robôs não reconheçam a diferença entre objetos ou outros robôs. Eles analisaram três tipos diferentes de sensores e sintetizaram um controlador para cada situação. Os controladores não utilizam memória e foram sintetizados usando a abordagem evolutiva, ajustando o valor de velocidade das rodas que estão relacionadas diretamente aos valores binários dos sensores, minimizando uma métrica de dispersão de objetos. Cada um dos controladores foi evoluído sobre milhares

de gerações. Resultados em simulações e robôs reais mostram que os controladores se ajustam bem à medida que mais robôs são incluídos no enxame e são robustos quando é incluídos ruídos no sistema.

Os mesmos autores resolveram a tarefa de agregação com outro controlador minimalista [23]. Neste trabalho, os robôs foram equipados com um sensor que determina se existe ou não existem robôs na vizinhança do mesmo. O controlador foi ajustado utilizando um algoritmo de busca exaustiva que deve ajustar as velocidades das rodas em duas situações: quando existem robôs na vizinhança e quando não existem. Como são utilizados robôs diferenciais devem ser ajustados 4 parâmetros em total para ambas situações. As simulações apresentam resultados com até 1.000 robôs e em robôs físicos com 40 robôs, em que a tarefa de organização é realizada satisfatoriamente.

2.3 Considerações finais

Neste capítulo, foram estudadas diferentes metodologias de síntese automática de controladores para enxames de robôs, elas foram classificadas em dois grandes grupos: os projetos automáticos que utilizam arquiteturas monolíticas e aqueles que utilizam arquiteturas modulares. Por sua vez, foram discutidas abordagens de projeto manual de controladores para realizar tarefas coletivas com restrições de distâncias com coesão entre os robôs do enxame. As abordagens manuais foram divididas em abordagens baseadas em comportamentos, teoria de controle supervisorio e campos potenciais artificiais.

Como pode ser observado, comportamentos com restrições de distâncias emergem no projeto automática de controladores para robô do enxame. Porém, maioritariamente se observa a emergência de comportamentos simples, principalmente para agregação e agrupação de objetos. No projeto manual, grande quantidade de trabalhos abordam o problema, mas eles não possuem flexibilidade para abordar outras tarefas coletivas além das que o método foi projetado.

Poucos trabalhos tratam a síntese automática de comportamentos com restrições de distância e coesão entre os robôs. Nos trabalhos que utilizam arquiteturas monolíticas com projeto baseado em campos potenciais, foi observado que o problema de restrições de distâncias com coesão entre robôs foi abordado, unicamente realizando ajuste de parâmetros

dos comportamentos individuais de cada robô. Neste tipo de abordagem são ajustados unicamente parâmetros, sem ajuste da topologia do controlador. Por sua vez, arquiteturas que utilizam redes neuronais, possuem mais flexibilidade para ajustar essas relações de interação. Porém, são especializadas fortemente para realizar comportamentos coletivos específicos. Por sua vez, arquiteturas minimalistas monolíticas sofrem com a mesma limitação.

Embora comportamentos com restrições de distâncias tenham sido atacados com arquiteturas modulares por alguns autores, normalmente este tipo de abordagem é dependente da tarefa. Assim, a especialização do método é feita depois de conhecida a tarefa. Na abordagem hierárquica, a especialização é fortemente dependente da tarefa. Primeiramente, é observada a tarefa coletiva e se não é possível encontrar uma solução adequada dela, a tarefa é dividida manualmente em subtarefas até que uma função objetivo apropriada seja encontrada para cada subtarefa. Logo controladores especializados nessas subtarefas são evoluídos: devem ser escolhidas, para cada subtarefa, o tipo de arquitetura, a quantidade de entradas, saídas, dentre outras características. Assim, em base a cada subtarefa é definido cada módulo particular. O usuário deve especializar novamente o método para cada nova tarefa coletiva que se queira abordar com o método.

O método AUTOMODE foi definido como uma abordagem independente da tarefa e, por isto, um método mais geral no que diz respeito as diferentes tarefas coletivas que o enxame pode abordar. No método AUTOMODE, primeiramente se especializa a abordagem para desenvolver tarefas que *a priori* são desconhecidas pelo usuário do método. Definir a abordagem desta forma, possui uma série de desafios pois a especialização pode impedir que certos tipos de tarefas coletivas sejam realizadas. Por exemplo, da revisão bibliográfica, observa-se que o método CHOCOLATE não consegue resolver problemas que são resolvidos unicamente com comunicação. Outra limitação no método AUTOMODE é que o método não considera a possibilidade dos robôs manterem distâncias específicas entre si. O método AUTOMODE tem vantagens, no que diz respeito ao usuário não ter o trabalho intensivo de re-especializar novamente o método para cada nova tarefa que o enxame deva enfrentar.

Por AUTOMODE ser um método geral é desejável que o método aborde uma grande quantidade de tarefas. Tendo em conta as deficiências dos métodos automáticos, especificamente da arquitetura modular do método AUTOMODE e da arquitetura monolítica do EVOSTICK, no que diz respeito as tarefas que podem ser abordadas pelos métodos, nesta tese é proposto

abordar o comportamento coletivo de enxames de robôs com restrições de distâncias, para os robôs manterem distâncias específicas entre eles. Para realizar este propósito, é definida uma nova instância do método AUTOMODE e do EVOSTICK chamadas de MATE e EVOSPACE, respectivamente.

Os métodos MATE e EVOSPACE incluem um novo comportamento que lida com restrições de distâncias a nível dos robôs individuais. Este novo comportamento chamado *formar*, é uma extensão que permite aumentar as atribuições do método para produzir comportamentos coletivos que resolvam tarefas coletivas em que a coesão dos robôs é necessária para a correta solução da tarefa. Nesta nova abordagem, os robôs podem ser organizados como um agregado de indivíduos em um padrão de rede, em que os robôs mantêm distâncias específicas entre si. Este módulo é construído com base em interações locais que são fundamentais para promover a tolerância a falhas e escalabilidade da abordagem.

Capítulo 3

Fundamentação Teórica

Neste capítulo serão apresentadas as ferramentas e conceitos que serão utilizados na tese. Primeiramente será definido o conceito de sistemas multi-robô na Seção 3.1, com as principais características. Logo após, a definição de enxames de robôs será apresentada na Seção 3.2 e serão classificadas as características positivas do seu uso. A definição de comportamentos primitivos dos robôs e a forma de composição desses comportamentos serão apresentadas na Seção 3.3. Após, na Seção 3.4, serão definidos e classificados os comportamentos coletivos em enxames de robôs. Uma descrição de robôs utilizados em enxames de robôs será apresentada na Seção 3.5.

O presente capítulo está estruturado da seguinte forma: Na Seção 3.6 serão definidos principalmente conceitos sobre os controladores de enxame de robôs. Assim, será realizada a divisão de controladores em controladores de baixo nível e supervisores. Dentre os controladores supervisores, a descrição do método de campos potenciais artificiais será realizada e serão colocadas duas leis de controle amplamente usadas nesta abordagem. Logo após, serão definidos os tipos de síntese de controladores para robôs de enxame. Na Seção 3.7 serão definidos dois métodos de projeto automático de controladores para robôs do enxame e como foi realizada a especialização do método em anteriores trabalhos relevantes da bibliografia. Finalmente, na Seção 3.9 são apresentadas as considerações finais do capítulo.

3.1 Sistemas multi-robô

Os sistemas multi-robô são compostos por um conjunto de robôs que compartilham e operam em um ambiente para desenvolver tarefas específicas. Eles são definidos como robôs físicos, com sensores e atuadores que de forma autônoma conseguem se adaptar nas mudanças do ambiente por intermédio de coordenação. Enquadra-se nos sistemas multi-robô os sistemas de robôs terrestres, aquáticos e aéreos. A utilização de sistemas multi-robô para a realização de tarefas conjuntas apresenta várias vantagens tais como especificadas em [Arkin\[2\]](#). Algumas das características positivas explicitadas abaixo são: o paralelismo nas atividades, aumento da quantidade de tarefas realizáveis, tolerância a falhas, distribuição de sensoriamento e atuação.

Paralelismo: A quantidade de robôs do sistema é suficiente para a realização de várias subtarefas em simultâneo para o desenvolvimento da tarefa grupal. Esta característica de realização de múltiplos objetivos em paralelo, que podem estar distribuídos no cenário de trabalho, permite a redução do tempo para completar a tarefa grupal, podendo produzir melhor desempenho global.

Aumento da quantidade de tarefas realizáveis: A cooperação entre os robôs permite que o sistema possa desenvolver tarefas não realizáveis pelos robôs de forma individual. Exemplos deste caso podem ser encontrados na cooperação dos robôs para carregamento e deslocamento de objetos de grande tamanho, navegação em ambientes irregulares por intermédio de morfogêneses, entre outros.

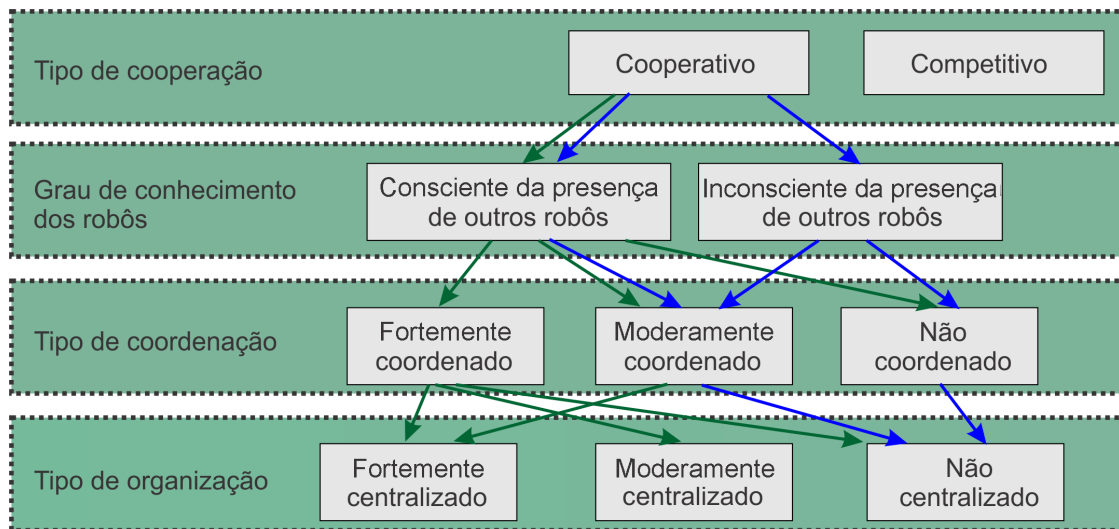
Distribuição de sensoriamento: O sensoriamento se dissemina nos robôs do conjunto que se encontram em várias regiões do cenário de trabalho, assim podem ser geradas redes de sensores para o intercâmbio de informações entre os robôs. Esta capacidade é muito bem explorada em times de robôs que devem realizar monitoramento de perímetro, áreas ou incluso para tarefas de resgate, exploração de ambientes tóxicos, entre outros.

Distribuição de atuação: Os atuadores encontram-se repartidos entre os robôs do sistema em várias regiões do ambiente, assim podem ser modificadas as características do ambiente e incluso de outros robôs de forma distribuída. Isto pode ser positivo em tarefas que exigem pontos de atuação que podem ser simultâneos, tais como na indústria automotiva, exploração na procura de explosivos, etc.

Tolerância a falhas: Perturbações ou defeito de sensores e atuadores degrada o desempenho do sistema de forma gradativa. A avaria das funcionalidades ou mesmo de um robô não impede o desenvolvimento da tarefa. O conceito esta relacionado com a redundância do sistema, no sentido que vários integrantes do time podem ter as mesmas funcionalidades e hardware, podendo desenvolver as mesmas tarefas e serem robôs intercambiáveis e anônimos.

Como explicitado na introdução, existem várias classificações realizadas para sistemas multi-robô na bibliografia [3, 7, 9–11]. Uma das mais relevantes é definida em [Iocchi, Nardi e Salerno](#)[8] que caracteriza este conjunto de robôs quanto a sua forma de coordenação. Esta taxonomia define o nível de cooperação do time, o grau de conhecimento que os robôs possuem entre eles, o tipo e grau de interação existente entre os robôs e finalmente o tipo de organização usada no sistema para realizar as tarefas. A seguir é realizada uma explicação dos conceitos [8]. Na Figura 3.1 são observadas as características inerentes aos sistemas multi-robô (setas verdes) e aos enxames de robôs (setas azuis).

Figura 3.1: Classificação de sistemas com vários robôs: sistemas multi-robô (setas em verde) e enxames de robôs (setas em azul).



Fonte: Adaptação de [Iocchi, Nardi e Salerno](#)[8].

Tipo de Cooperação: A cooperação se define como operação conjunta de robôs para executar uma tarefa global que possivelmente não possa ser realizada por cada um dos robôs individualmente, ou que a execução pode ser melhorada usando mais de um robô,

obtendo melhor desempenho. O grau de cooperação define o tipo do sistema multi-robô, podendo não existir nenhum tipo de cooperação ou incluso competição entre os robôs. Na competição entre os robôs não é compartilhado nenhum objetivo em comum. No outro extremo, encontram-se os sistemas nos quais os robôs são fortemente cooperativos para completar as tarefas coletivas.

Grau de conhecimento dos robôs do time: O conjunto de robôs em um sistema cooperativo pode ou não ter conhecimento dos robôs componentes da equipe. O grau de conhecimento é uma propriedade do sistema que define o nível de cooperação existente. Robôs sem conhecimento de outros robôs executam suas tarefas como se fossem os únicos robôs presentes no sistema, sem levar em conta a presença e as ações dos outros integrantes. Nestas abordagens, não existe comunicação entre os robôs. A cooperação de robôs com conhecimento dos outros membros do time permite realizar uma coordenação mais orientada e eficiente aos objetivos grupais e em ambos os casos permitem atingir tarefas grupais. Nesta última a coordenação pode ser dada por intermédio de comunicação entre os membros.

Grau de conhecimento das intenções dos robôs: O sistema multi-robô pode estar composto por indivíduos que coordenam as suas ações em função das ações executadas pelos outros robôs de tal forma que pode ser obtido um melhor desempenho do sistema. Este tipo de sistema, em que os robôs conseguem discernir sobre os comportamentos de outros robôs, permite executar tarefas em que existem recursos compartilhados e devem ser usados por vários robôs. Sistemas fortemente coordenados baseiam suas ações em um alto grau de interação entre os robôs. Neste tipo de sistema no qual os robôs tem conhecimento das ações dos outros pode ser necessária comunicação direta e/ou indireta entre os robôs. Conhecer o estado dos robôs do time permite tomar decisões e alocar tarefas de forma mais eficiente no intuito de reduzir interferência entre os robôs.

Embora a interação coloque mais flexibilidade no sistema, permitindo uma melhor utilização dos recursos disponíveis, há tarefas que podem ser realizadas efetivamente com pouca ou sem nenhum tipo de comunicação entre robôs. A vantagem destas abordagens são que o projeto é menos complexo, mas podem acontecer interferências e um grande desperdício de recursos e energia, porque os robôs podem estar executando tarefas opostas e díspares.

Grau de organização: O grau de organização define a forma que as decisões são realizadas no sistema de robôs. As decisões podem ser executadas por intermédio de

sistemas centralizados e descentralizados. Nos **sistemas centralizados** as decisões são tomadas por um robô ou vários robôs chamados de líderes e os outros membros recebem as ordens e tem menos autonomia nas decisões. Nos **sistemas descentralizados** os robôs são completamente autônomos e o processo de decisão não depende de nenhum robô líder, as decisões são tomadas em conjunto de forma distribuída no sistema.

Os sistemas centralizados podem ser fortemente centralizados em que um único robô é o líder. Esta liderança pode ser dinâmica ou estática permanecendo sempre sobre o poder de um único robô. Os sistemas podem ser moderadamente centralizados quando possuem topologias hierárquicas com vários líderes em que alguns deles podem subordinar outros líderes. Estas topologias possuem a desvantagem de precisar de comunicação e uma falha em um dos robôs degrada de forma significativa o desempenho do sistema.

A vantagem dos sistemas fortemente centralizados é que permitem encontrar soluções globais e ótimas para um problema dado. Porém, sistemas multi-robô descentralizados são mais robustos no que diz respeito a falhas, pois cada indivíduo decide de forma autônoma sem necessidade de líderes.

Na Figura 3.1 podem ser observadas as características dos enxames de robôs na classificação realizada anteriormente. Os enxames de robôs são sistemas cooperativos nos quais o tipo de conhecimento dos robôs sobre a presença dos outros robôs pode ser considerado desde muito alto quando existe comunicação local entre os robôs, até um conhecimento nulo, quando os robôs realizam tarefas individualmente sem ter conhecimento do seus pares. Por sua vez, os robôs de um enxame de robôs poder ter conhecimento das ações de outros robôs, porém, em menor medida que nos sistemas multi-robô fortemente coordenados. Finalmente, referente ao tipo de organização, pode ser dito que são sistemas distribuídos nos quais são utilizados controladores descentralizados.

Na seguinte seção, serão apresentadas de forma mais detalhada as características de enxames de robôs que se caracterizam por serem robôs autônomos, auto-organizados, redundantes, que executam tarefas de forma paralela e as suas interações são consideradas de locais [12].

3.2 Enxames de robôs

A robótica de enxame (*swarm robotics*) aborda o problema de coordenação de sistemas de robôs com controladores descentralizados, em que grande quantidade de robôs relativamente simples devem ser coordenados usando informações locais do ambiente, interação local e controladores simples [12]. A diferença da abordagem multi-robô é que as missões são realizadas por grande quantidade de robôs e os robôs utilizam controladores descentralizados. Por sua vez, os enxames de robôs geralmente não possuem conhecimento preciso sobre a existência de outros robôs do enxame e não possuem conhecimento preciso sobre as intenções de outros robôs do enxame. Esta abordagem geralmente é inspirada em sociedades animais (formigas, cupins, vespas e abelhas) que podem realizar tarefas além das capacidades de cada robô individual independente da falta de informação global, ruído no ambiente, erros no processamento da informação [77]. A ideia por trás do conceito é a construção de um sistema de robôs de baixo custo com capacidades de sensoriamento e atuação limitadas em vez da construção de um único robô de alto custo e desempenho [12, 78]. Assim o sistema está habilitado para desenvolver o mesmo conjunto de tarefas que um único robô de alto custo, embora seja de forma distribuída.

Os robôs que compõem o enxame são relativamente simples na sua estrutura e controlador, tal que o projeto de controlador individual de cada robô pode gerar comportamentos coletivos desejados que emergem das interações locais entre os robôs e dos robôs com o ambiente [12]. Os enxames de robôs são classificados como homogêneos quando o hardware e o software de controle dos robôs são idênticos. Alternativamente, podem ser denominados de heterogêneos, quando os robôs são dotados de hardware e/ou software de controle que são diferentes [16]. Na seguinte seção são definidas as principais características dos enxames de robôs que por ser sistemas compostos por vários robôs herdam as propriedades explicitadas na Seção 3.1.

3.2.1 Principais características

Os enxames de robôs têm características que são consideradas positivas como a **autonomia** e **auto-organização**. Estes sistemas não dependem de nenhuma entidade centralizada para a tomada de decisões e alocação de tarefas. Por isso, geralmente este tipo de sistema não possuem líderes, nem infraestruturas externas. Estudos baseados na biologia indicam

que na maioria dos insetos sociais não existem mecanismos de coordenação centralizados para controlar a sua operação sincronizada. Embora não seja apresentada essa característica, os sistemas se apresentam robustos, flexíveis e escaláveis [12].

O sistema é fortemente **redundante**, pois a maioria dos robôs do enxame são capazes de executar cada uma das ações individuais que são necessárias para realizar a tarefa dada. Em outras palavras, nenhum robô é indispensável. Em uma aplicação típica de enxames de robôs, os robôs operam em **execução paralela** na realização das tarefas. O paralelismo permite a realização de várias subtarefas simultaneamente para o desenvolvimento do objetivo grupal, em que os robôs se adaptam as características dinâmicas do ambiente e da equipe de forma autônoma, auto-organizada e baseada em informações locais do ambiente. Outra característica é a **localidade de interação** em que cada robô possui um alcance limitado de comunicação e percepção. Como consequência disto, cada robô interage diretamente, somente com um número limitado de robôs que estão em sua vizinhança. A principal implicação do anterior é que cada robô desconhece o tamanho geral do enxame e não é afetado totalmente por ele.

Estas características de autonomia, auto-organização, redundância, localidade das interações e a execução paralela explicitadas acima são características positivas no sistema que promovem tolerância a falhas, escalabilidade e flexibilidade [16]. A **tolerância a falhas** está relacionada com a robustez do sistema que é definida como uma característica desejada que impede que perturbações do ambiente ou falha em algum robô degradem o desempenho do time de forma significativa e que o sistema seja capaz de continuar operando [12]. Falhas são consideradas quando são afetadas as funcionalidades de atuadores ou sensores do robô que dificultam ou impedem de realizar a tarefa atribuída. A multiplicidade de sensoriamento e atuação distribuídos nos indivíduos do enxame pode incrementar a redundância do sistema que fomenta a robustez a falhas.

Outra característica desejada é a **flexibilidade** entendida como capacidade dos indivíduos de lidar com um amplo espectro de tarefas e ambientes diferentes [12], criar diferentes soluções para diferentes tarefas e se adaptar as mudanças no ambiente dependendo das necessidades desse momento específico. Assim, os enxames de robôs tem a flexibilidade de oferecer diferentes técnicas de coordenação de forma auto-organizada, para reagir a contingencias e modificações no ambiente [16]. Finalmente, outra característica encontrada

nestes sistemas de robôs é a **escalabilidade** que se define como a capacidade do sistema operar sem modificações na estrutura de controle quando aumenta a quantidade de membros e o desempenho do sistema permanece inalterado. Isto significa que os mecanismos de coordenação, que garantem a operação do sistema, permaneçam inalterados pelas mudanças no tamanho do enxame [12].

São adicionadas mais duas vantagens na utilização de enxames de robôs as quais são **custos e eficiência energética** [78]. Na primeira, o custo deste tipo de sistema pode ser significativamente mais baixo na concepção, fabricação e manutenção que robôs de alto desempenho. Isto é devido a que os indivíduos de um agregado podem ser produzidos de forma massiva enquanto robôs especializados requerem de maiores custos na fabricação e manutenção. A eficiência energética pode ser observada desde o momento que os robôs possuam sensores e atuadores de menor consumo quando comparado a outros robôs que devem manter um hardware mais demandante, e desde que o sistema de robôs possa conseguir completar a tarefa em menor tempo. Comportamentos coletivos coordenados permitem o uso eficiente da energia do sistema, tal como apresentado em [Turgut et al.\[46\]](#). Isto se torna positivo, por exemplo, em situações em que a obtenção de energia para recarga de baterias seja um problema.

3.3 Definição de comportamento

O comportamento é definido como um conjunto de ações ou respostas realizadas por indivíduos devido a vários estímulos que são as entradas internas e também geradas pela interação com o entorno. Nas plataformas de robôs os comportamentos são codificados no controlador que realiza uma função determinada por intermédio de definição de um conjunto de regras. Comportamento é definido como uma lei de controle que agrupa uma série de restrições com o propósito de atingir e manter objetivos [79]. Comportamentos válidos para robôs individuais são: evitar obstáculos, atingir objetivos, seguir paredes, exploração, entre outros. Assim, por exemplo, o comportamento de seguir paredes é uma lei de controle na qual são sensoreadas entradas dos sensores que depois são usadas para definir ações as quais permitem que o robô mantenha o objetivo de movimento seguindo o a parede. No caso do comportamento de exploração o objetivo a ser manter é evitar colisões enquanto o robô se

locomove no cenário. Comportamentos podem ser compostos por outros comportamentos primitivos e as regras incluídas em cada comportamento podem ser a definição do sentido de movimento do robô, limiares de ativação das regras, entre outros.

Comportamentos primitivos são aqueles que agrupam restrições e leis apropriadas para atingir e manter objetivos. Eles são considerados primitivos para planejamento, aprendizado, estrutura, síntese e análise de comportamentos coletivos [79]. As restrições incluídas num comportamento são as características mecânicas dos atuadores, dos sensores e das interações entre robôs. No projeto desses comportamentos devem ser usadas abordagens pensando no baixo nível do robô e no nível coletivo para definir um comportamento possível e realizável no nível coletivo. Os comportamentos são primitivos na medida em que são necessários para gerar outros comportamentos e são um conjunto mínimo que permite ao robô atingir e manter objetivos. Comportamentos primitivos são definidos como módulos que permitem atingir objetivos de alto nível por intermédio da composição deles, permitindo gerar vários comportamentos, robustos e complexos.

Crerios de projeto e seleção de comportamentos primitivos são baseados em análise de consistência, simplicidade, localidade, repetitividade, estabilidade, robustez e escalabilidade [79]. Definir a consistência é projetar o comportamento que atinja e mantenha o objetivo para o qual foi projetado satisfazendo a especificação. A simplicidade é desejada para correção de erros e análise. A localidade do comportamento permite que seja projetado com informações locais disponíveis pelos sensores de cada robô. Também deve ser analisada a repetitividade do comportamento de forma isolada nas diferentes avaliações. Analisar a sua estabilidade significa ver possíveis oscilações no resultado da resposta desejada frente a perturbações. Analisar a robustez é observar que o desempenho do comportamento não seja degradado frente a perturbações e ruído nos atuadores e sensores. Finalmente observar a escalabilidade do mesmo analisando os efeitos do comportamento frente a variação no número de integrantes do time.

Os comportamentos são considerados módulos de entradas e saídas que interagem com o entorno. As entradas de cada comportamento são estímulos produzidos pelo entorno de cada robô e estes estímulos podem ser ou não observáveis dependendo dos sensores que o robô disponha para adquirir esses sinais. Por sua vez, cada comportamento pode ter como entradas as saídas de outros comportamentos. As saídas dos comportamentos são respostas

que são enviadas para os atuadores e efetadores do robô para realizar as ações necessárias, modificação do entorno do robô, e cumprir as tarefas designadas. Então, um robô realiza o sensoriamento do ambiente por intermédio do seus sensores e recebe estímulos que logo devem ser processados para decidir a ação que o robô deve realizar.

Assim, sendo S o conjunto de todos os estímulos que podem ser detectados pelo robô, ou seja, adquiridos pelos sensores. Cada estímulo é representado como $e_i = (p, \lambda)$, em que p é a classe e λ é a magnitude do estímulo, respectivamente [2]. A classe p são os conjuntos de entidades que o robô pode sensoriar, tais como obstáculos, robôs, objetivos, etc. A magnitude do estímulo λ são as quantidades mensuráveis pelos sensores, tais como a distância, temperatura, grau de concentração, entre outros. λ pode ser definida como contínua ou discreta.

A relação entre estímulos $e_i \in S$, $i \in [0, N_e]$ com a resposta $r_i \in R$, $i \in [0, N_n]$ do robô se dá por intermédio de uma função $b: S \rightarrow R$, $b \in B$, apresentada na Equação (3.1), que mapeia as entradas (estímulos) com a saída definida como a ação do robô [2]. Esta função comportamental b quando dado estímulos $e_i \in S$ fornece uma ação resposta $r_i \in R$ para o robô. Sem perder generalidade e para simplificar a nomenclatura, a partir de agora será omitida a variável tempo nos vetores, ângulos e variáveis dependentes do tempo.

$$r_i = b(e_1, e_2, \dots, e_{N_e}). \quad (3.1)$$

Esta resposta $r_i \in \mathbb{R}^3$ é definida como um vetor com módulo e orientação, em que o módulo é a magnitude da resposta do robô e a orientação caracteriza a direção de movimento. No caso bidimensional a resposta r_i é definida como um vetor de módulo $|r_i|$ e fase θ_d como especificado na Equação (3.2).

$$r_i = |r_i| e^{j\theta_d}, \quad (3.2)$$

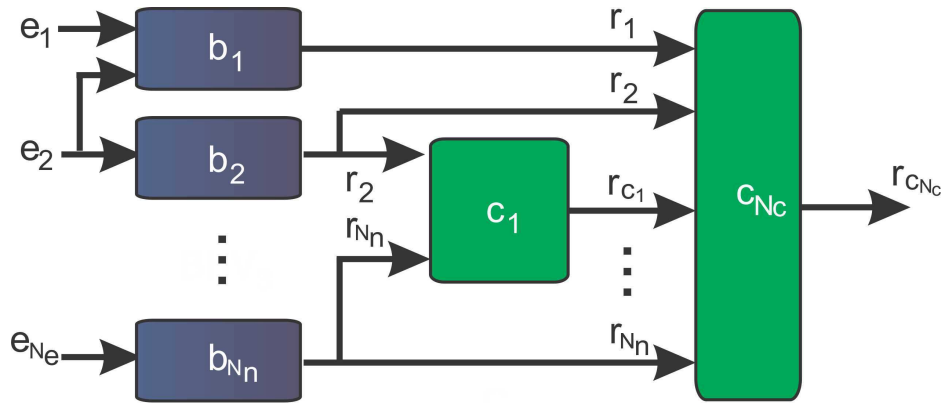
em que $|r_i|$ é a magnitude da resposta referenciada no sistema de coordenadas fixo do robô O_c e θ_d o ângulo direção da resposta referenciada desde o eixo x_c do sistema O_c . A presença de um dado estímulo observável $e \in S$ é necessária mas não é suficiente para produzir uma resposta r . Assim, a resposta r depende de λ e de uma variável limiar τ que define a ativação ou não do comportamento. Por isto, deve ser superado o limiar τ para que r seja produzida

por intermédio de e .

3.3.1 Coordenação de comportamentos

A coordenação de comportamentos pode ser definida como uma estratégia para controle das ações encapsuladas nos comportamentos dos robôs. Na Figura 3.2 observa-se um diagrama estímulo-resposta [2] de um robô, em que são observados estímulos do ambiente $e_i \in S, i \in [0, N_e]$ que são a entrada de comportamentos, $b_i \in B, i \in [0, N_n]$. Cada um desses comportamentos terá uma única saída resposta $r_i \in R, i \in [0, N_n]$, correspondentes para os atuadores do robô ou para novos módulos comportamentais. Os coordenadores $c_i \in C, i \in [0, N_c]$, vão permitir escolher o comportamento mais adequado dependendo do estímulo de outros comportamentos. A saída do coordenador é uma resposta $r_{c_i} \in R, i \in [0, N_c]$, produto das entradas respostas r_i geradas de comportamentos b_i e respostas de outros coordenadores r_{c_i} . Assim, a função de coordenação determina qual dessas saídas serão selecionadas para operar no robô, gerando novos comportamentos. Cabe mencionar que novos coordenadores podem ser adicionados em cascata para gerar coordenações dessas respostas.

Figura 3.2: Diagrama estímulo-resposta de um robô com comportamentos b_i , coordenadores c_i , vetores r_i, r_{c_i} e estímulos e_i



Fonte: Elaborada pelo autor.

Genericamente, a função de coordenação $c : R \rightarrow R$ é definida como na Equação (3.3).

$$r_{c_i} = c_i \left(r_1, r_2, \dots, r_{N_n}, \dots, r_{c_1}, r_{c_2}, \dots, r_{c_{N_c}} \right). \quad (3.3)$$

As duas formas mais utilizadas de coordenar comportamentos são a cooperativa e a competitiva. Na primeira é por intermédio da composição de comportamentos ponderando de cada um deles [2] e na última é por intermédio da escolha de um único comportamento individual sobre outros mediante a competição de comportamentos [36]. Cada uma destas classes possuem vantagens e desvantagens que devem ser observadas na construção do sistema.

Na Equação (3.4) apresenta-se a formulação matemática, na forma linear, da função coordenação que engloba os dois tipos de coordenação citada acima:

$$r_{c_i} = \sum_{j=1}^{N_n} a_j r_j + \sum_{k=1}^{N_c} a_k r_{c_k}. \quad (3.4)$$

Então r_{c_i} realiza a soma ponderada de vetores com seus respectivos pesos a . No caso da coordenação cooperativa não existe restrição das ponderações a , no caso de competição a escolha será de um único comportamento e assim um único peso a será diferente de zero, ativando a resposta desse único comportamento.

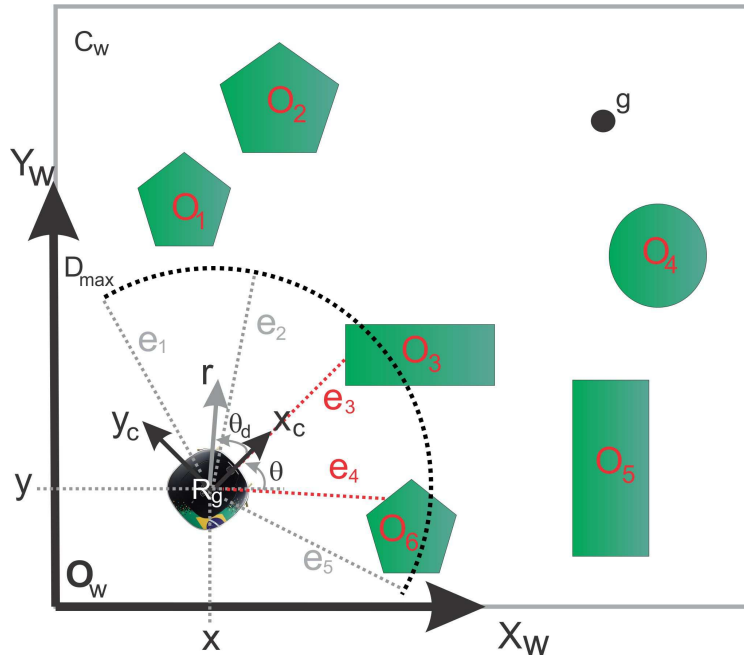
Caso seja desejado um comportamento suave e contínuo pode-se experimentar esquemas que envolvam composição de módulos. De igual forma, pode-se experimentar com o ajuste das ponderações para cada comportamento na hora da composição para obter o resultado desejado. No entanto, é mais difícil de estimar ou analisar como vários comportamentos afetam ao sistema simultaneamente. A escolha de um único comportamento por vez permite maior previsibilidade, mas o vetor resposta da coordenação possui descontinuidades.

Para exemplificar, na Figura 3.3 apresenta-se um robô móvel de tração diferencial com o modelo descrito na Equação (3.5).

$$\begin{cases} \dot{x} = v \cos(\theta) \\ \dot{y} = v \sin(\theta) \\ \dot{\theta} = \omega \end{cases} \quad (3.5)$$

A velocidade $v \in \mathbb{R}$ é a velocidade linear do robô e $\omega \in \mathbb{R}$ a velocidade angular do robô. A pose do robô é representada com as variáveis x , y e θ representada respeito ao sistema de coordenadas O_w . O robô possui cinco sensores de proximidade e cinco sensores de luminosidade num ambiente dinâmico $C_w \in \mathbb{R}^2$ que deverá evitar obstáculos de um conjunto

Figura 3.3: Robô de tração diferencial num cenário com centro de coordenadas O_w e obstáculos em cor verde.



Fonte: Elaborada pelo autor.

O para atingir um objetivo g definido como uma fonte luminosa. Por sua vez, deve explorar o ambiente caso nem o objetivo nem obstáculos sejam observados. O robô possui uma posição x, y e orientação θ no sistema de coordenadas $O_w(x_w, y_w)$ fixo no cenário C_w . O ângulo θ_d é o ângulo entre o sistema de coordenadas do robô $O_c(x_c, y_c)$ e o vetor de direção r definido na Equação (3.2). Um bom controlador para esse robô tentará minimizar esse ângulo θ_d a cada instante de tempo para produzir a direção especificada pelo vetor r .

Nesse cenário o robô recebe os estímulos $e \in S$ observáveis pelos sensores proprioceptivos e exteroceptivos do robô em um instante de tempo particular, em que p são as classes de entidades detectáveis, ou seja os obstáculos O e objetivo g observáveis no alcance dos sensores em um dado instante de tempo, λ são as distâncias aos obstáculos e a distância à fonte luminosa. Com essa informação o controlador deverá produzir a resposta r que vai permitir o robô se movimentar no cenário cumprindo a especificação de evitar obstáculos e atingir o alvo. Uma possível solução de controlador da Equação (3.4) é a definida na Equação (3.6)

que aplica a técnica de ativação competitiva de comportamentos.

$$r_t = \begin{cases} r_{prox} & p = p_o \quad \wedge \quad \tau_{\lambda_o min} < \lambda_o < \tau_{\lambda_o max} \\ r_{luz} & p = p_g \quad \wedge \quad \tau_{\lambda_o min} < \lambda_g < \tau_{\lambda_g max} \quad \wedge \quad r_{prox} = 0 \\ r_{exp} & CC \end{cases}, \quad (3.6)$$

em que r_{prox} é a resposta do comportamento evitar obstáculos, r_{luz} é a resposta do comportamento para atingir o objetivo g e r_{exp} é a resposta do comportamento de exploração. τ_{λ_o} e τ_{λ_g} são os limiares que ativam os comportamentos, p_o é o obstáculo mais próximo e p_g o objetivo mais próximo g , λ_o e λ_g são as distâncias medidas ao obstáculo e objetivo mais próximo, respectivamente. Pode ser observado na Figura 3.3 que nesse instante de tempo somente dois sensores detectam obstáculos, o mais próximo é o estímulo e_4 produzido pelo obstáculo O_6 da classe p de obstáculos, como a distância esta no intervalo $\tau_{\lambda_o min} < \lambda_o < \tau_{\lambda_o max}$, o comportamento r_{prox} é ativado para evitar obstáculos.

O vetor r_t é produzido pelo controlador supervisor e transformado nos comandos para controle do robô utilizando o controlador de baixo nível (Seção 3.6.1). Este controlador de baixo nível depende do tipo de robô a ser comandado. No caso do robô de tração diferencial é realizada uma transformação $r_t \rightarrow \begin{bmatrix} v_r & v_l \end{bmatrix}$. Um controlador com este tipo de transformação será estudado na Seção 4.2.1.

3.4 Comportamento coletivo

O comportamento coletivo em enxames de robôs é definido como a composição de comportamentos primitivos dos robôs individuais. Em enxames de robôs este comportamento surge da interação entre robôs simples com capacidades limitadas de sensoriamento e atuação. Embora os indivíduos sejam limitados quanto as suas capacidades, eles produzem comportamentos que são mais complexos que os comportamentos simples de cada robô individual. Este comportamento coletivo observado possui as características mencionadas na Seção 3.2.1. Esses indivíduos poderiam não ser necessariamente limitados quanto a suas capacidades, mas são relativamente simples em comparação com o potencial que poderia ser alcançado nesse sistema. Alguns comportamentos observados nunca poderiam ser executados por um único indivíduo e sim na cooperação ou competição do grupo. Esta cooperação

permite completar as tarefas, como se fosse uma única entidade complexa, com robustez, flexibilidade e tolerância a falhas, sem a necessidade de controle centralizado nem modelo global do ambiente, fornecendo uma ótima solução para problemas sofisticados em larga escala.

Os comportamentos coletivos usualmente encontrados na natureza e utilizados em enxames de robôs são classificados em quatro categorias diferentes, as quais são comportamentos com restrições de distâncias, de navegação, de tomada de decisão coletiva e outros comportamentos relevantes [13]. O primeiro grupo engloba os comportamentos que tem por objetivo a organização e distribuição das robôs no cenário de trabalho. A segunda categoria inclui os comportamentos que se encarregam da coordenação do robô no que diz respeito à navegação. Na terceira categoria abarca os comportamentos de decisão coletiva para o consenso de decisões e alocação de tarefas entre os indivíduos. Na última categoria é o grupo de comportamentos coletivos que não se encaixam em nenhuma das categorias anteriores. Essas classes de comportamentos com sua respectiva explicação e exemplos se encontra desenvolvida na Seção 3.4.1.

O comportamento coletivo na natureza pode ser observado em sociedades animais tais como formigas, abelhas, pássaros e colônias de peixes [78] em que cada agregado tem a capacidade de resolver facilmente problemas complexos, embora um único indivíduo da mesma espécie geralmente não consegue resolver de forma isolada. Observam-se mecanismos de alocação de tarefas descentralizados que permitem dividir as tarefas entre os indivíduos para poder abordar uma tarefa complexa [78]. É reportado também que comportamentos coletivos na natureza permitem abordar o problema de sobrevivência de uma espécie [78], em que são observadas a divisão das tarefas para a defesa e proteção da colônia.

3.4.1 Tipos de comportamentos coletivos

Como foi mencionado acima, o comportamento coletivo em enxames de robôs é definido como a composição de comportamentos primitivos dos robôs individuais. Os comportamentos coletivos podem ser classificados em quatro categorias diferentes, as quais são comportamentos com restrições de distâncias, de navegação, de tomada de decisão coletiva e outros comportamentos relevantes.

Comportamentos com restrições de distâncias

Neste comportamento os robôs se organizam no cenário de trabalho por intermédio de agregados de indivíduos, formação de padrões, cadeias e estruturas de robôs fisicamente conectados.

Agregação é o comportamento a nível coletivo que possui o objetivo de reunir um grupo de robôs em uma região do cenário determinada [13]. Este comportamento é elementar pois para realizar outras tarefas mais complexas, é necessário que os robôs se encontrem inicialmente com conectividade. Agregação é um comportamento muito útil, pois permite que os robôs permaneçam o suficientemente próximos para interagir. Alguns dos trabalhos que estudam este comportamento utilizando controladores projetados de forma automática com arquiteturas modulares são apresentados em Duarte, Oliveira e Christensen; Duarte et al.; Francesca et al.; Francesca et al.; Kuckling et al.; Hasselmann, Robert e Birattari[18–20, 60, 62, 67]. Trabalhos que abordam o projeto de controlador com arquiteturas monolíticas e projeto automático são apresentados em Francesca et al.; Dorigo et al.; Gauci et al.[22, 23, 57]. Finalmente, trabalhos que abordam o projeto de controlador de forma manual são apresentados em Lopes et al.; Gazi e Passino[14, 21], dentre outros.

Agregação é muito comum na natureza, com aplicações inspiradas na organização das baratas [22, 23], colônias de abelhas, formigas e pinguins [13]. O comportamento complementar da agregação é **Dispersão** de robôs. Este comportamento permite que os robôs se distribuam no cenário de trabalho para se manter afastados abrangendo mais superfície. Por ser ambos comportamentos elementares, podem ser à base de outros comportamentos tais como exploração ou movimento coletivo.

Formação de padrões é um comportamento coletivo com restrições fortes de distâncias em que a coesão é relevante na resolução da tarefa coletiva. A coesão é definida como a interação entre robôs mantendo distâncias específicas. A coesão permite o enxame de robôs atingir certa forma global, regular e repetitiva. A formação pode ser usada para atingir certa topologia de rede, cobertura de uma região ou formar uma configuração no cenário de trabalho para o movimento coletivo dos robôs [6]. Exemplos de estudos que utilizam formação de padrões são encontrados em projeto manual de controladores [6, 29, 41, 43, 44, 46–48, 80, 81] e projeto automático de controladores [5, 56, 62, 68, 82]. Os tipos de padrões encontrados são

padrões hexagonais, circulares, entre outros [4, 29]. Usualmente os robôs devem respeitar distâncias entre eles para gerar o padrão desejado. Estes comportamentos se encontram presentes tanto na biologia como na física: No primeiro caso na distribuição de colônias de bactérias no ambiente e no segundo caso na distribuição das moléculas e nas formações de cristais [13]. Alguns exemplos que utilizam estes comportamentos para cobertura de regiões do cenário de trabalho são apresentados em Yang, Ding e Hao; DE SOUZA e Endler; Howard, Mataric e Sukhatme[6, 83, 84], dentre outros. Uma variação desses comportamentos são os comportamentos de **Formação de cadeias**, em que os robôs devem conectar dois pontos, como ser a base e a fonte de alimento, por exemplo [28, 85]. Estes comportamentos também podem ser usados como princípio para a navegação das entidades e são observados na natureza principalmente nas formigas [13].

O comportamento coletivo **Montagem de robôs** é um comportamento com restrições de distâncias que produz a conexão física entre os robôs da equipe. Estes sistemas podem se conectar por intermédio do processo de **morfogênese de robôs** que permite que os robôs se conectem num determinado padrão de formação ou estrutura para realizar uma tarefa específica. Estes tipos de comportamento foram estudados em trabalhos relevantes como [15, 25, 57, 86, 87]. Como observado, a montagem e morfogênese de robôs tem diferentes propósitos tais como a estabilidade na navegação [76] e aumento da força total de translado de objetos [24]. Fontes de inspiração são encontradas em várias espécies de formigas em que elas se ensamblam fisicamente em forma de linha para criar pontes, paredes e barcas para navegar em superfícies aquáticas [13].

Da mesma forma que existe o comportamento de montagem e morfogênese em robôs, existe o comportamento coletivo **Agrupação de objetos**, fundamental para tarefas de construção em que devem ser agrupados objetos com proximidade entre si. Trabalhos de agrupação de objetos são encontrados em Lopes et al.; Gauci et al.; Francesca et al.; Kuckling et al.[14, 18, 58, 67]. **Montagem de objetos** por sua vez, é um comportamento coletivo em que o enxame de robô posiciona os objetos com uma conexão física estabelecida, como em Petersen, Nagpal e Werfel[27]. Observam-se na natureza em colônias de insetos na agrupação de alimento como nas formigas, na criação das colmeias nas colônias de abelhas, entre outros [13].

Comportamentos de navegação

Comportamentos de navegação são comportamentos coletivos que tem o propósito de coordenar os movimentos no enxame de robôs. Este comportamento permite o desenvolvimento de outros comportamentos mais complexos. Dentro destes comportamentos se encontra a exploração coletiva, movimento coordenado e transporte coletivo [13].

Exploração coletiva é um comportamento coletivo que tem o propósito de reconhecimento de um ambiente ou áreas de interesse da mesma [13]. Geralmente estes comportamentos estão combinados com outros tais como formações de padrões ou de cadeias para realizar essa exploração. Estes padrões permitem a geração de estruturas virtuais entre dois pontos objetivos que servem como caminho para serem usados por outros robôs para a localização, intercâmbio de mensagens e navegação. Este comportamento pode ser usado para a navegação guiada do enxame e pode ser combinado com o de dispersão para cobertura de área e exploração. Um trabalho relevante que estuda técnicas de exploração coletiva em enxames de robôs foi realizado em [Kegeleirs, GARZÓN RAMOS e Birattari\[88\]](#).

Movimento coordenado é o comportamento coletivo de navegação em que os robôs se deslocam de forma coesa e ordenada para atingir um objetivo particular. No movimento coordenado os robôs devem manter distâncias predefinidas entre si. Para tal propósito, podem ser utilizados comportamentos de formação de padrões. Este comportamento é muito útil, pois permite a redução de interferência física entre membros, redução da oclusão no sensoriamento e navegação mais precisa, como observado em algumas colônias de pássaros e peixes [13], em que este comportamento fomenta nos robôs um consumo eficiente de energia e aumento das chances de sobrevivência. Abordagens relevantes se estudaram em [Pincioli et al.; Hettiarachchi e Spears; Turgut et al.; Spears et al.; Ferrante et al.; Navarro e Matía; Mendiburu, Morais e Lima; Elkilany, Abouelsoud e Fathelbab\[5, 29, 46, 49, 68, 89–91\]](#).

O **Transporte coletivo** é um comportamento coletivo do enxame que tem por objetivo o transporte de objetos ou entidades entre locais. Estes objetos transportados geralmente são pesados ou de grande tamanho e não podem ser carregados pelos robôs de forma individual [13]. O comportamento de transporte pode ser composto com outros comportamentos tais como o de montagens de robôs para obter mais força de transporte ou a utilização do comportamento de tomada de decisão para escolher a direção de transporte. Por sua

vez pode ser usado, por exemplo, o comportamento de formação de cadeias para definir a orientação de translado do objeto [13]. Exemplos deste tipo de transporte se encontram em [Chen, Gauci e Groß](#); [Ruiz, Bacca e Caicedo](#)[92, 93].

Tomada de decisão coletiva

A tomada de decisão coletiva refere-se a como os robôs escolhem as soluções possíveis a um determinado problema. Na primeira categoria se encontra o comportamento coletivo de **consenso** em que os robôs devem decidir um valor possível entre um conjunto válido de valores [13]. Assim, deve ser decidida uma escolha entre uma série de alternativas possíveis para maximizar o desempenho do sistema como um todo. Em enxames de robôs o consenso pode ser atingido usando comunicação direta ou indireta em que são adquiridas variáveis tais como velocidade e ângulo de orientação do enxame, entre outros [13]. Assim, o consenso permite definir a direção de navegação, o caminho mais curto, a zona mais apropriada de agregação, a primeira tarefa a ser realizada, etc. Exemplos deste comportamento coletivo se encontram em [Trianni et al.](#); [Scheidler et al.](#); [Trabattoni, Valentini e Dorigo](#); [Strobel, Ferrer e Dorigo](#)[94–97]

No segundo grupo encontra-se o comportamento coletivo de **alocação de tarefas** em que os robôs do enxame devem distribuir uma série de tarefas possíveis entre si de tal forma que operem em paralelo. A forma de alocação geralmente é dinâmica para maximizar o desempenho do sistema e geralmente essa alocação se baseia em critérios tais como a observação do ambiente de trabalho e de outros robôs [13]. O tipo de comunicação usado para realizar a alocação pode ser de forma direta ou indireta. Os comportamentos de alocação de tarefas se apresentam apropriados em contextos de movimento coletivo, montagens e agregação de objetos. Fontes de inspiração na natureza se encontram nas abelhas e formigas [13]. Exemplos de este tipo de comportamento se encontram em [Agassounon e Martinoli](#); [Pang et al.](#); [Irfan e Farooq](#); [Parker](#)[40, 98–100].

Outros comportamentos relevantes

Outros comportamentos coletivos relevantes de interesse que não pertencem às categorias anteriores são a interação com humanos, detecção de falhas e regulação do tamanho do

agregado de robôs [13].

O comportamento coletivo de **interação com humanos** tem o objetivo de exercer controle sobre o enxame de robôs por intermédio de um operador que aplica uma ação de controle e recebe uma realimentação do sistema. Embora os enxames de robôs sejam projetados para trabalhar de forma auto-organizada e distribuída, formas de controle externo podem ser necessárias [13]. As abordagens mais comuns são usando interfaces gráficas ou de realidade aumentada nas quais o operador seleciona os robôs para controlar e recebe estímulos resposta da ação de controle aplicada. Outra forma de controle é por intermédio do reconhecimento de fala, gestos e rostos adicionando outros mecanismos como os comportamentos de tomada de decisão para o consenso conjunto dos robôs do enxame. Neste tipo de sistemas são aplicados controladores hierárquicos devido a que o poder do controle do operador é considerado limitado, pois o controle geralmente não é aplicado a todos os robôs [13].

No comportamento coletivo para a **detecção de falhas**, os robôs possuem mecanismos de sinalização em que o conhecimento do estado dos robôs vizinhos permite determinar se eles apresentam defeitos, por exemplo, quando não existe sincronização com os robôs vizinhos. Embora os enxames de robôs sejam robustos a falhas, o comportamento de detecção de falhas pode ser necessário em sistemas em que os membros devem estar cientes do estado atual dos vizinhos para a tomada de decisão apropriada. Este comportamento é baseado principalmente no comportamento dos vagalumes [13]. Na **regulação do tamanho** do enxame de robôs, o comportamento coletivo estima e ajusta o número de robôs no enxame para determinar um desempenho ótimo. Este comportamento é também inspirado nos vagalumes, em que cada robô emite um sinal e estima os sinais recebidos por seus vizinhos em cada instante de tempo [13].

3.5 Robôs utilizados em enxames de robôs

As pesquisas em robótica de enxame tem utilizado uma ampla variedade de robôs para realizar tarefas cooperativas no contexto de enxames de robôs. Nestas pesquisas, o enxame é projetado para possuir grandes quantidades de robôs. Normalmente, os robôs possuem sensores e atuadores com desempenhos moderados e de baixo custo [101]. Por sua vez, os

robôs utilizados possuem pequenas dimensões, para facilitar experimentos de laboratório [101]. Dentre os robôs terrestres de enxame mais relevantes, se encontram os robôs Kilobot [53], Khepera [101], Kobot [54], Foot-bot [102], e-puck [52] e S-bot [76], dentre outros. Nesta seção se descrevem dois robôs usados nesta tese: os robôs Foot-bot e os robôs e-puck.

3.5.1 Robô e-puck

O robô e-puck é um robô de tração diferencial com chassis aproximadamente circular de aproximadamente $0,07\text{ m}$ de diâmetro. Na figura 3.4 é apresentado o robô e-puck com a descrição do seus sensores e atuadores usados.

Figura 3.4: Vista lateral do robô e-puck apresentando o seus sensores e atuadores.

Modulo de visão:
 Câmera VGA,
 Espelho hiperbolico 360°
 3 LEDs separados 120°

Microcontrolador, bateria superior.
 Comunicação: Wi-Fi, Bluetooth, USB.

Modulo do sensor de proximidade relativa.

Chassi do robô:
 Anel de LEDs,
 8 sensores de proximidade/luminosidade,
 2 atuadores (motores das rodas),
 3 sensores de chão,
 Bateria inferior.



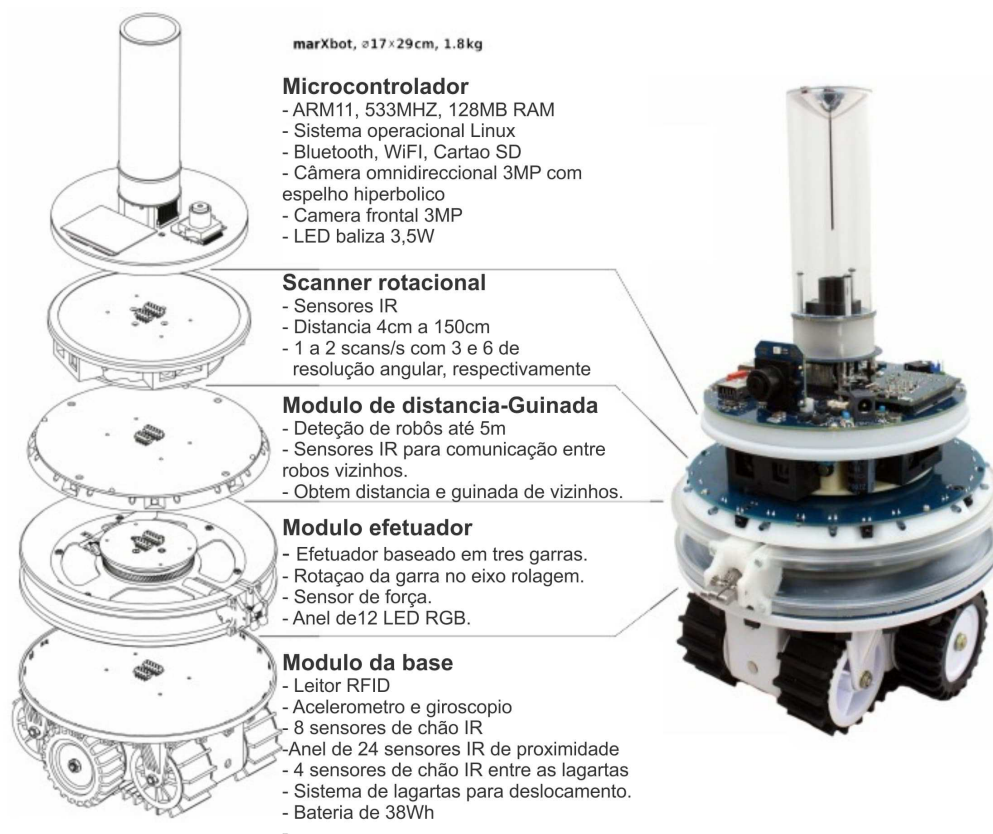
Fonte: Adaptação de [Francesca et al.\[19\]](#).

O e-puck possui 8 sensores infravermelhos que permitem medir proximidade e luminosidade, simultaneamente. Utilizando o sistema de coordenadas do robô O_c da Figura 3.3, os sensores de proximidade e luminosidade estão localizados no perímetro do robô da seguinte forma : 4 na frente, 2 nas laterais e 2 na parte traseira em $15^\circ, 45^\circ, -15^\circ, -45^\circ, 150^\circ$ e -150° , aproximadamente. Possui 3 sensores infravermelhos para medir a cor do chão localizados na parte inferior frontal do chassi do robô. Por sua vez possui um sistema de comunicação baseado em infravermelho e modulação do sinal para troca de mensagens com outros robôs,

consiste em 12 transmissores e 12 receptores localizados uniformemente no perímetro circular do robô. Este mesmo sistema permite por intermédio da medição da potência do sinal, estimar a posição de robôs da vizinhança [103, 104]: estima a distância e ângulo em relação ao sistema de coordenadas O_c do robô. O e-puck possui o microcontrolador Overo Gumstick que permite embarcar Linux. O e-puck possui também um sistema de visão de baixo custo composto de uma câmera e um espelho hiperbólico que permite um ângulo de visão de 360° .

3.5.2 Robô Foot-bot

Figura 3.5: Robô Foot-bot especificando seus módulos e sensores.



Fonte: Adaptação de [Bonani et al.\[102\]](#).

Os robôs Foot-bot possuem tração diferencial composta de dois motores de $2W$ que alimentam um sistema de lagartas e rodas (Figura 3.5). O robô possui uma velocidade de 30 cm/s e é um robô autônomo utilizado no sistema multi-robô desenvolvido no projeto

Swarmanoid [25], sendo uma adaptação do robô MarXbot [102]. Ele é um robô modular tanto mecanicamente, como na parte elétrica e no software, alimentado com uma bateria de 10 Ah recarregável de forma autônoma numa estação de carga.

O robô possui uma arquitetura multiprocessador que consiste num processador central que toma conta de tarefas de grande poder computacional, e vários microcontroladores que tomam conta dos sensores e atuadores. Esta arquitetura permite a paralelização dos processos obtendo melhores desempenhos que um único microcontrolador. O robô possui um modelo de comunicação e de detecção de robôs vizinhos que permite a obtenção de localização relativa entre distâncias de 10 cm até 5 m. A base do robô inclui sensores infravermelhos para a detecção de obstáculos e sensores para a detecção do chão. Estes sensores de obstáculos são 24 e estão distribuídos ao redor do perímetro do robô, 8 sensores para a detecção da cor do chão e outros 4 para a detecção de contato com o chão. Ele conta também com um acelerômetro e um giroscópio de três eixos para a medição de orientação do robô.

O módulo do efetuator permite a montagem entre robôs. Além disso, o módulo contém LEDs RGB. O módulo do scanner de distância tem 4 sensores de distância infravermelhos montados numa plataforma giratória, dois deles para pequenas distâncias ([40,300] mm) e outros dois para cobrir distâncias longas ([200,1500] mm). Finalmente, o módulo superior inclui duas câmeras, uma baliza LED, o processador principal e seus periféricos, incluindo uma placa Wi-Fi, dentre outros.

3.6 Projeto de controle para enxames de robôs

Como os robôs do sistema precisam ser controlados para atingir nosso propósito de movimento coletivo, são utilizados controladores que são os encarregados de ajustar o comportamento do robô ao desejado por intermédio da manipulação dos sinais de controle. Esses sinais devem minimizar o erro entre as variáveis a serem controladas e as referências definidas pelo projetista do sistema de controle. Dependendo do nível de abstração dos controladores podem ser considerados como controladores de baixo nível e supervisores.

3.6.1 Tipos de controladores

Na robótica de enxame têm-se dois tipos de controladores em função da sua interação com o hardware do robô: o controlador de baixo nível e o controlador supervisor.

Controlador de baixo nível

O controlador de baixo nível do robô é um controlador que permite a tradução dos comportamentos definidos no controlador supervisor para referências que devem ser seguidas pelo robô para conseguir o comportamento desejado. Estas referências se traduzem em comandos para os atuadores do robô. Exemplos de controladores deste tipo são controladores PID [105], controladores robustos [105], controladores que utilizam lógica difusa [106], controladores adaptativos [107], entre outros. Estes controladores se comunicam com o hardware e são dependentes da plataforma. Caso seja utilizada outra plataforma este controlador deverá ser modificado para se adaptar as necessidades de movimento do robô.

Um dos controladores mais populares nas aplicações de engenharia é o PID [105]. O amplo uso do controlador PID extensivamente na bibliografia se refere a facilidade na sua implementação, pois tem a flexibilidade na escolha de parâmetros e graus de liberdade para tentar eliminar o erro de estado estacionário e escolha da resposta adequada para o projeto com os termos proporcional e derivativo. Um controlador PID calcula o valor de erro como a diferença entre uma variável de processo medida e um ponto de ajuste desejado, chamado de referência [105]. O controlador tenta corrigir o erro gerando uma saída de controle $\omega(t)$.

O cálculo do controlador PID envolve três diferentes parâmetros constantes, os quais são proporcional k_p , integral k_i e derivativo k_d . O termo proporcional do controlador depende do erro presente, o integral sobre a acumulação de erros em instantes de tempos passados, e o termo derivativo é uma previsão de erros nos instantes de tempo futuros [105]. Algumas aplicações podem exigir o uso de apenas uma ou duas ações para fornecer o sistema de controle apropriado. Um controlador PID será chamado de PI, PD, P na ausência das ações de controle respectiva. Controladores PI são bastante comuns, já que a ação derivativa é sensível ao ruído de medição. A necessidade do termo integral é para tentar evitar erros no estado permanente e que o sinal de saída atinja o valor desejado. Os outros dois termos do

controlador permitem definir a resposta adequada para cumprir as especificações.

Os termos proporcional, integral e derivativo são somados para calcular a saída do controlador PID. Assim, o controlador de baixo nível na versão digital para implementação em microcontrolador trabalha com amostras do sinal analógico. A Equação (3.7) apresenta a função de transferência discreta do PID [105].

$$u(k) = k_p \theta_d(k) + k_i T_a \left(\sum_{i=1}^k \theta_d(i-1) + \theta_d(k) \right) + k_d \frac{\theta_d(k) - \theta_d(k-1)}{T_a}, \quad (3.7)$$

em que $u(k)$ é a variável a ser controlada, T_a é o período de amostragem, $\theta_d(i-1)$ é o erro acumulado até iteração $i-1$, $\theta_d(k)$ e $\theta_d(k-1)$ é o erro no instante de tempo k e $k-1$, respectivamente.

O termo proporcional produz uma ação de controle proporcional ao sinal de entrada. Se o ganho k_p é muito alto o sistema poderá ir para a região de instabilidade [105]. Um ganho baixo produz um controlador pouco sensível aos erros na entrada e, por tanto, uma saída de controle muito pequena. O termo integral inclui proporcionalmente ao erro e a duração do mesmo. A integral no PID é a soma dos erros instantâneos ao longo do tempo e do *offset* acumulado que deveria ter sido corrigido anteriormente. O erro acumulado logo é multiplicado pelo ganho k_i .

Finalmente o termo derivativo é calculado pela determinação da derivada do erro multiplicada pela taxa de variação do ganho derivativo k_d . A magnitude da contribuição do termo derivativo para a ação de controle global é chamado de ganho derivativo, k_d . Controle derivativo é usado para reduzir a magnitude do sobrepasso produzida pelo componente integral [105]. No entanto, o termo derivado retarda a resposta transitória do controlador. Além disso, a diferenciação de um sinal amplifica o ruído e, portanto, este termo no controlador é fortemente sensível ao ruído no termo de erro, e pode causar instabilidade se o ruído e o ganho derivativo são suficientemente grandes [105].

Controlador supervisor

O controlador supervisor do robô é o encarregado de especificar a direção de movimento do robô em um nível abstrato no sentido que a interação com o hardware é nula e por isto, o controlador pode ser implementado em qualquer plataforma robótica. Este controlador

modela os comportamentos individuais de cada robô, os quais enviam comandos em forma de vetores para o controlador de baixo nível. Logo, o controlador de baixo nível é quem transforma esses vetores em sinais de controle para a atuação do robô. O controlador supervisor trata o robô como uma partícula pontual e por isto o controlador é considerado independente da plataforma robótica. Normalmente, este tipo de controlador não modela as colisões entre robôs nem as colisões entre os robôs e o ambiente.

Normalmente, nas arquiteturas monolíticas que utilizam redes neurais, tanto o controlador de baixo nível como o supervisor estão encapsulados no mesmo controlador, não existindo distinção entre eles, como observado em alguns exemplos tais como os apresentados em Duarte, Oliveira e Christensen; Duarte et al.; Duarte et al.; Duarte, Oliveira e Christensen; Francesca et al.; Dorigo et al.[22, 57, 60–63], dentre outros. Nas arquiteturas modulares normalmente estes controladores estão separados, para permitir a independência entre o método e o robô utilizado, exemplos podem ser encontrados em Francesca et al.; Hasselmann, Robert e Birattari; Kuckling et al.; Pinciroli et al.; Hettiarachchi e Spears; Hettiarachchi e Spears[4, 5, 18, 20, 65, 67], dentre outros. Porém, existem exceções e podem ser observadas arquiteturas modulares nos quais não existe distinção entre os dois controladores, como no trabalho realizado em König, Mostaghim e Schmeck[66].

Este controlador supervisor normalmente é projetado utilizando a abordagem baseada em comportamento [36] ou a abordagem de campos potenciais artificiais [37, 38] como as principais. A seguir será apresentada a teoria de campos potenciais artificiais.

A teoria de campos potenciais artificiais foi primeiramente definida em Khatib[37]. Inicialmente este enfoque era aplicado para robôs individuais ou pequena quantidade de robôs onde o ambiente exercia unicamente forças virtuais sobre o robô, sendo repulsivas quando se tratava de obstáculos ou atrativas quando se tratava de objetivos de interesse. Logo, o enfoque foi sendo implementado em outros tipos de robôs móveis e em sistemas com vários robôs. Uma generalização destes conceitos foram utilizados no Enfoque Psicomimético [29]. Este enfoque se diferencia de [37], em que os robôs não calculam o campo potencial de forma global, senão, que cada robô calcula a força que aplicam unicamente os robôs da sua vizinhança, reduzindo tempo de computo [38]. Este enfoque utiliza os mesmos conceitos dos potenciais virtuais para modelar as interações entre indivíduos de um sistema de robôs. Estas interações são consideradas como forças virtuais locais que modelam as interações entre

indivíduos. A abordagem psicomimética é amplamente utilizada na atualidade em enxames de robôs. Neste enfoque, são utilizadas várias leis da física para modelar as interações entre os robôs, como a lei da gravitação universal [44], a lei de interação de um sistema masa-mola [47], a lei de Lennard-Jones [68], dentre outras. No entanto, por ser motivado em leis físicas, não quer dizer que se restrinja somente a elas [29]. O objetivo da abordagem é construir sistemas escaláveis, auto-organizados e tolerantes a falhas principalmente como é observado em sistemas físicos. A abordagem foi aplicada para construir uma variedade ampla de formações e comportamentos de robôs estáticos e dinâmicos. Essas aplicações incluem malhas geométricas regulares para sensoriamento distribuído, bem como comportamentos dinâmicos de vigilância de áreas e perímetros.

Utilizando a teoria de campos potenciais artificiais são observados comportamentos com restrições de distâncias, tais como formações de padrão hexagonal (todas as distâncias entre robôs vizinhos são iguais) e padrão quadrado (dois tipos de distâncias) [29]. São observadas também formações de padrão similar às colmeias de abelhas, nas quais deve existir comunicação global de identificadores para a definição de cada robô [29]. Por sua vez, são publicadas formações de cadeias de robôs nas quais é explorada a assimetria das regiões de atração e repulsão [29]. São observadas estas técnicas de interação em formações para a realização de cobertura de perímetro e área [108], entre outros. Por sua vez, podem ser utilizados líderes para realizar configurações iniciais dos integrantes que finalmente podem servir para realizar outras tarefas de interesse, tais como montagens de robôs e agrupação de objetos. Nestas abordagens são observadas formações do tipo estrela e árvore [109].

Na teoria de campos potenciais artificiais, cada entidade sensoreia e reage às forças virtuais calculadas de forma local. Os sensores das entidades devem sensorear os estímulos para calcular as forças que serão exercidas no robô. As forças virtuais levam o sistema de robôs para uma configuração desejada que minimize a energia potencial do sistema [29]. A estrutura permite produzir o controle distribuído de grande quantidade de robôs físicos. Nesta abordagem, os robôs são tratados como partículas físicas tridimensionais de massa pontual que possuem uma posição e uma velocidade tal que a variação da velocidade em

cada entidade pode ser calculada como apresentado na Equação (3.8).

$$\Delta v = \frac{F \Delta t}{m}, \quad (3.8)$$

em que m é a massa da partícula e F a força virtual calculada. A força é limitada por um valor F_{max} que é equivalente a limitar a velocidade máxima do robô restringindo o controle nos atuadores. Caso exista mais de um robô na vizinhança, a força resultante F em cada robô é calculada como a soma de cada uma das forças produzidas por cada robô presente na sua vizinhança.

Lei de Gravitação Universal

A lei de Gravitação Universal modela a interação entre duas partículas por intermédio da fórmula apresentada na Equação (3.9), em que $p = 2$, F é a magnitude do vetor força aplicado na partícula i produzida pela partícula j . A constante de gravitação universal é G , as massas das partículas são m_i e m_j e d_{ij} é a distância entre as duas partículas. No caso da robótica pode ser aproveitado para modelar as interações entre robôs e entre os robôs e o ambiente.

$$F_{ij} = \frac{Gm_i m_j}{d_{ij}^p}. \quad (3.9)$$

Como a lei é somente atrativa, é modificada para incluir uma parte repulsiva [29]. Assim, a direção da força tem o sentido da interação entre as partículas, variando a direção se $d_G > d_{ij}$ ou $d_G \leq d_{ij}$, em que d_G é a distância limiar que define se a força F será de repulsão ($d_G \leq d_{ij}$) ou atração ($d_G > d_{ij}$). Esta lei de interação é mais apropriada para modelar principalmente interações sólidas na qual observa-se rigidez nas formações.

Função de Lennard-Jones

A função de potencial de Lennard-Jones modela a interação entre moléculas e átomos. De igual forma que a Equação (3.9), a função de Lennard-Jones permite modelar as interações entre robôs e dos robôs com o ambiente. O potencial de Lennard-Jones pode ser dado pela

seguinte expressão generalizada [89].

$$P_{ij} = 4\epsilon \left[2 \left(\frac{\sigma}{d_{ij}} \right)^{2\alpha} - \left(\frac{\sigma}{d_{ij}} \right)^\alpha \right], \quad (3.10)$$

em que a distância entre o robô i e o robô j ($i \neq j$) é definida como d_{ij} , ϵ determina a profundidade do campo potencial ou o que é equivalente a intensidade das forças. Os parâmetros σ e α são aqueles que permitem definir a distância desejada d_{ij}^* entre os robôs, que é definida como $d_{ij}^* = 2^{1/\alpha}\sigma$. Em longas distâncias, a força atrativa predomina e faz com que as entidades se aproximem, isto acontece quando $d_{ij} > d_{des}$. Em curtas distâncias, quando $d_{ij} < d_{ij}^*$, a força repulsiva predomina e faz com que as entidades se afastem.

Assim, quando esta função potencial é derivada realizando a derivada negativa em função da distância entre as entidades tem-se a força da interação, apresentada na Equação (3.11).

$$F_{ij} = -\frac{\partial P_{ij}}{\partial d_{ij}} = -\frac{4\alpha\epsilon}{d_{ij}} \left[2 \left(\frac{\sigma}{d_{ij}} \right)^{2\alpha} - \left(\frac{\sigma}{d_{ij}} \right)^\alpha \right]. \quad (3.11)$$

Quando é substituído o valor de σ na equação se obtém a Equação (3.12). O valor de σ é obtido de $d_{ij}^* = 2^{1/\alpha}\sigma$.

$$F_{ij} = -\frac{\partial P_{ij}}{\partial d_{ij}} = -\frac{2\alpha\epsilon}{d_{ij}} \left[\left(\frac{d_{ij}^*}{d_{ij}} \right)^{2\alpha} - \left(\frac{d_{ij}^*}{d_{ij}} \right)^\alpha \right]. \quad (3.12)$$

O valor mínimo de P_{ij} se encontra na distância desejada $d_{ij} = d_{ij}^*$ e esse valor de potencial é $P_{ij} = -\epsilon$. Nesta situação não atuam forças no robô (a força é $F_{ij} = 0$) alcançando o balanço natural. Quando são modeladas interações entre robôs e existe mais de um vizinho a força resultante F é calculada como a superposição das forças entre o robô e seus vizinhos. Quando é alcançado o balanço natural do sistema os robôs mantêm o equilíbrio nas posições de menor energia do sistema. O comportamento emergente é uma formação de padrão hexagonal produto das interações dos robôs, se as distâncias desejadas d_{ij}^* entre robôs são consideradas todas iguais. Dependendo das configurações dos parâmetros podem ser modelados diferentes tipos de rigidez na formação, tais como formações sólidas, líquidas ou gasosas.

3.6.2 Síntese de controladores

Tanto o controlador de baixo nível como o supervisor podem ser projetados com técnicas manuais ou automáticas. No Capítulo 2 foi realizada a definição de síntese de controlador.

A síntese de controladores em enxames de robôs permite obter o controlador de cada robô individual. Este controlador produz um repertório de comportamentos que permitem que o robô realize tarefas coletivas com os outros robôs do enxame.

Projetar o controlador de robôs dada uma missão particular requer que as especificações sejam fornecidas no nível do enxame, e que essas especificações sejam transformadas em comandos de baixo nível na implementação dos comportamentos individuais de cada robô, em que as interações com outros robôs e com o ambiente geram o comportamento coletivo que satisfaz essa especificação. O problema do projeto de controladores para enxames de robôs é abordado com técnicas manuais e automáticas de projeto [13]. O objetivo de ambas é que o enxame produza comportamentos coletivos para realizar as tarefas definidas pelos usuários. Isto significa projetar o controlador individual de cada robô que instanciado nos robôs do enxame produzam comportamentos coletivos desejados que resolvem determinada tarefa por intermédio das interações entre os robôs e dos robôs com o ambiente.

O **projeto manual** de controlador é uma abordagem *ad hoc* fortemente dependente da tarefa a ser realizada pelo enxame. O projeto é indireto, pois não existe uma metodologia que permita derivar os comportamentos individuais dos robôs a partir do comportamento coletivo desejado [18]. Assim, o projetista implementa, testa e modifica o comportamento dos robôs individuais, de forma recursiva, até que o comportamento coletivo emergente é produzido por intermédio de interações complexas robô-robô e robô-ambiente [18], sendo dificilmente previsível a partir do conhecimento das regras do robô individual. Geralmente estes sistemas são construídos de forma modular definindo os comportamentos primitivos de baixo nível para logo adicionar comportamentos mais complexos em camadas superiores. Entre estes métodos se encontram geralmente os métodos baseados em robótica comportamental, campos potenciais artificiais e a teoria de controle supervisorio (Seção 2.1).

Um caminho possível para a solução desta dificuldade é gerar o controlador dos robôs do enxame utilizando **projeto automático**, sem ajuste manual [18]. Métodos automáticos permitem sintetizar o controlador individual de cada robô por intermédio de sucessivas

avaliações do comportamento coletivo do enxame no ambiente em que devem desenvolver a tarefa coletiva [18]. A síntese é guiada por intermédio de um algoritmo de otimização que deve maximizar, ou minimizar, uma função objetivo para decidir entre as possíveis estruturas de controle e ajustar os parâmetros da arquitetura. Assim, nesta abordagem, a tarefa é especificada no nível macroscópico como uma especificação de usuário (a função objetivo), que guia um algoritmo de otimização que derivará o comportamento individual dos robôs. Este comportamento individual instanciado em cada robô do enxame definirá o comportamento coletivo que resolva essa tarefa. Métodos automáticos para a síntese de controladores de robôs de um enxame podem ser divididos em duas classes principais: A abordagem monolítica e a abordagem modular (Seção 2.2). A primeira, normalmente baseada em robótica evolutiva [110], utiliza arquiteturas em forma de redes neurais [111] e um algoritmo artificial evolutivo para ajustar os parâmetros e/ou a arquitetura da rede [112]. A segunda utiliza arquiteturas modulares como máquinas de estado finitas [70], dentre outras. Nas arquiteturas modulares o algoritmo de otimização que guia o processo pode não ser necessariamente baseado em evolução artificial.

3.7 AutoMode e EvoStick

3.7.1 O poder de representação

Como definido na introdução, o poder de representação se refere a capacidade do método de representar de forma abstrata um controlador coletivo que resolva uma tarefa dada. Este poder de representação define implicitamente o conjunto de tarefas que são possíveis de ser realizadas com o enxame, e por sua vez, define o conjunto de tarefas coletivas que são realizáveis pelo enxame. O poder de representação está definido pela relação entre o *modelo de referência* e o *tipo da estrutura do controlador*.

Definir o **modelo de referência** é especificar as características do robô, definir os sensores e atuadores e a relação deles com o software do controlador: devem ser definidas as variáveis de leitura dos sensores e de escrita dos atuadores com seus intervalos de medida que serão disponíveis para o controlador do robô [18]. Assim, de cada sensor podem ser coletados, por exemplo, o módulo de cada medida e o ângulo de onde são produzidos esses estímulos

do ambiente. Por sua vez, dos atuadores, por exemplo, pode ser escrita a variável que comanda a velocidade das rodas do robô, ou que liga uma luz LED do robô, etc. Para mais informações, na Seção 3.8.1 será apresentado um modelo de referência possível para o robô e-puck, descrito na Tabela 3.1, com a especificação do seus sensores e atuadores utilizados do robô e as variáveis de leitura e escrita dos sensores e atuadores, respectivamente.

Definir a **estrutura do controlador** é especificar a arquitetura e a topologia do controlador. Por sua vez, específica de que forma esse robô vai processar a informação dos sensores e de que forma vai comandar o robô para interagir no ambiente em função do modelo de referência definido. Diferentes tipos de arquiteturas para realizar controle em enxames de robôs foram propostos na bibliografia. As arquiteturas mais relevantes, no contexto de projeto automático, são as máquinas de estado finitas [70] e redes neurais [111]. A escolha da topologia da rede neuronal ou da máquina de estados influencia a relação de como o robô vai interagir com o meio externo. No projeto automático de controladores para enxames de robô estas escolhas são denominadas de restrições de projeto [18]. Assim, devem ser escolhidas as características dessa arquitetura. Por exemplo, se a arquitetura é uma rede neuronal, devem ser escolhidas a quantidade de camadas ocultas e o número de nós, a relação entre as entradas e saídas de rede, entre outros. No caso de uma máquina de estados deve ser definido o número de estados, o número de transições da rede, a relação entre as entradas e saídas dentro de cada estado, entre outros. Cada uma dessas arquiteturas e topologias produziram diferentes tipos de poder de representação da plataforma.

Para exemplificar como é feita a definição do modelo de referência e da arquitetura e topologia de rede é colocado o seguinte exemplo: Suponha que um enxame de robôs esta composto por robôs homogêneos tanto em software como no hardware. Os robôs possuem tração diferencial e um conjunto de sensores e atuadores: sensores para detectar as cores do chão e sensores de proximidade com obstáculos e outros robôs. Por sua vez os robôs possuem atuadores que comandam as rodas do robô.

Definir o modelo de referência para este robô supondo que se utilizam todos os atuadores é definir os sensores e atuadores a ser usados: por exemplo, desse robô se define que vai ser utilizado os sensores de proximidade, vão ser adquiridos a intensidade medida de cada sensor: $v_{i_{prox}} \in \mathbb{R}$ ($v_{i_{prox}} : [0,1]$) com $i \in \{1,2,\dots,n\}$. Por sua vez, se define que vai ser utilizado o sensor de chão: gnd_i com $gnd_i \in \{preto,branco,cinza\}$. Define-se os atuadores e

os comandos a serem escritos: $v_i \in \mathbb{R}$ ($v_i : [-0,16; 0,16] m/s$) com $i \in \{l,r\}$ as velocidades da roda esquerda e direita. Definir a arquitetura e a topologia é definir, por exemplo, que vai ser utilizada uma rede neuronal sem nós ocultos em que todas as entradas da rede se conectam com as saídas. Por sua vez se define que cada uma dessas leituras do sensor é uma entrada da rede e cada uma das variáveis de comando é uma saída da rede. Nesta simples definição da arquitetura, foram colocadas restrições nas plataformas no que diz respeito a como o robô interage com o meio externo.

Após a definição do modelo de referência, é visualizada grande quantidade de tarefas coletivas que estes robôs podem realizar em conjunto [69]. Este conjunto de tarefas é denominado de *tarefas coletivas possíveis* [18]. Estas definições do tipo de robô com seu hardware e software define as interações entre robôs e as interações entre robôs com o ambiente possíveis de serem feitas com esse conjunto de sensores e atuadores. Depois de definir o tipo de arquitetura e topologia, foi definido como cada entrada (sensor) se relaciona com cada saída (atuador). Como pode ser observado no exemplo, estas definições já condicionam o tipo de tarefas que esse enxame de robôs pode realizar. Nestas condições, o conjunto de tarefas foi reduzido para um subconjunto denominado de *tarefas realizáveis* pelo enxame de robôs [18].

Nos métodos de projeto automático que sintetizam controladores para enxames de robôs, existe um problema de projeto referente à definição do poder de representação do método. Esta definição do poder representacional do método especifica a flexibilidade do método para ajustar as relações de interação entre os robôs e dos robôs com o ambiente no comportamento coletivo emergente. Este poder de representação uma vez definido, condiciona o enxame de robôs no que diz respeito às tarefas que podem ser realizáveis pelo enxame.

3.7.2 AutoMoDe

AutoMode é uma abordagem de projeto automático modular de controladores para enxames de robôs [18]. O resultado da síntese é um controlador modular que instanciado em cada robô permite resolver tarefas coletivas definidas por um usuário. Esta abordagem se caracteriza por ser um processo *offline* em que todo o procedimento do projeto é feito em simulação e a validação dos controladores é realizada primeiramente na simulação, e logo em

robôs físicos.

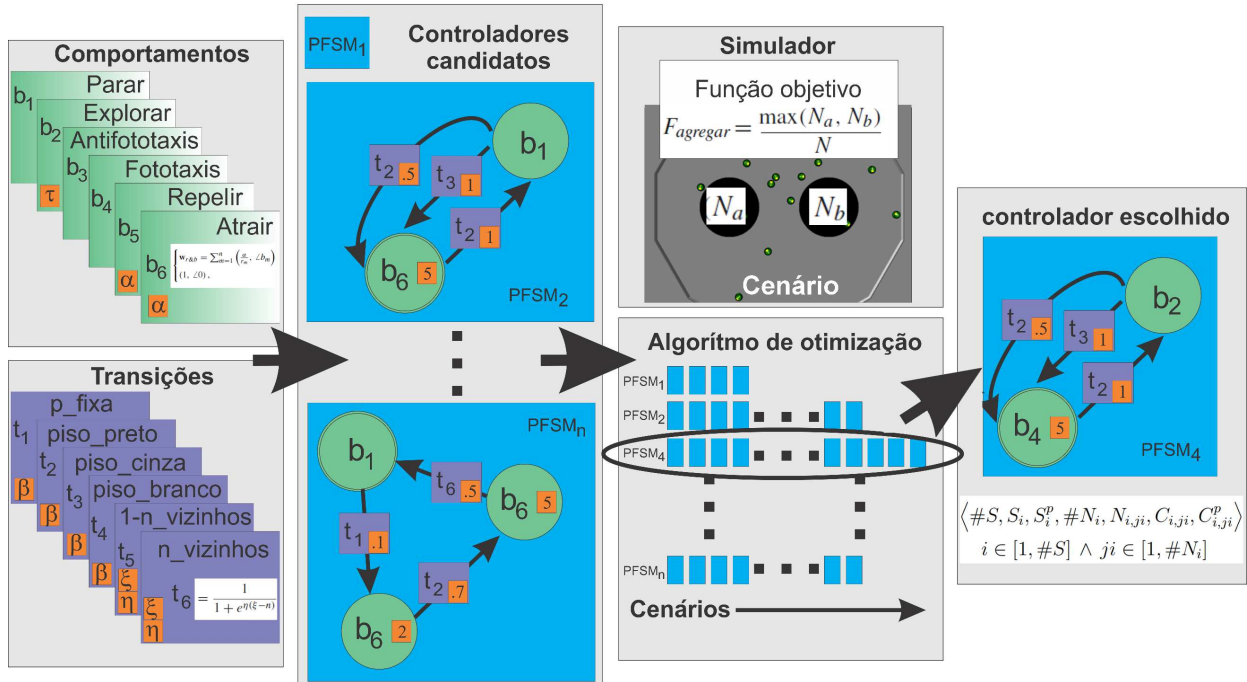
Esta abordagem é semelhante ao aprendizado de máquina em que se tem conjuntos de instâncias de treinamento e outras para teste [113]. As instâncias de treinamento são os cenários nas quais os robôs desenvolvem as tarefas coletivas e nelas que se busca o controlador individual dos robôs (a configuração) que maximize a função objetivo do sistema. Nas instâncias de teste será avaliado esse controlador e será observado se o comportamento coletivo resolve de fato a tarefa. Ou seja, se o controlador individual implementado em todos os robôs permitem o desenvolvimento do comportamento coletivo que resolve a tarefa grupal.

Desde o ponto de vista do aprendizado de máquina, a escolha do controlador mais apropriado para a tarefa deve ser o suficientemente geral para resolver instâncias não vistas no treinamento da mesma tarefa que quer ser abordada. O objetivo implica encontrar uma configuração de controlador que minimize uma medida de custo sobre o conjunto de instâncias observadas e que logo generalize para instâncias semelhantes, mas não observadas na fase de ajuste ou treinamento. A ideia da abordagem é tratar as diferenças de resultados entre a simulação e o mundo físico como um problema de generalização, tentando reduzir o sobreajuste do controlador a instâncias (cenários) particulares [18]. Isto é obtido por intermédio da injeção de blocos de comportamento e transições definidos *a priori* que reduzem o poder representacional do controlador, pois o software de controle gerado esta restrito a pertencer ao espaço de controladores que se podem compor a partir desses módulos dados.

Como qualquer método automático, o método AutoMode precisa especializar-se [114]. A pessoa que realiza o procedimento é denominada de especialista, pois, sua experiência é fundamental para definir os módulos que tentaram explorar todas as capacidades dos robôs [18]. Na definição dos módulos, o especialista estará determinando indiretamente as tarefas coletivas realizáveis pelos robôs. Por outro lado, na definição do modelo de cada robô, o especialista esta definindo as características dos robôs e indiretamente as tarefas coletivas que são possíveis com esse conjunto de robôs. Uma das características desejadas no projeto é realizar o casamento entre as tarefas coletivas possíveis e realizáveis, ou seja, que os módulos projetados pelo especialista explorem todas as características fornecidas pelo modelo de referência dos robôs [18].

Na Figura 3.6 apresenta-se o esquema de projeto de controlador para robô de enxame. Nesta seção, descreve-se resumidamente o esquema de projeto do método AUTOMODE-

Figura 3.6: Projeto de síntese de controlador para robô de enxame utilizando o método AUTOMODE-CHOCOLATE, apresentando as principais características.



Fonte: Elaborada pelo autor.

CHOCOLATE [18]. Maiores detalhes serão apresentados na Seção 3.8. Primeiramente, o especialista define a **plataforma robótica** que vai utilizar para definir o enxame, nos experimentos realizados no método AUTOMODE são utilizados os robôs e-puck [52]. Depois disso, o especialista define os módulos individuais denominados de comportamentos, transições e os seus respectivos parâmetros a ajustar. O **espaço de busca do controlador** esta composto pelos controladores candidatos: as possíveis máquinas de estado probabilísticas finitas [70] que podem ser formadas pela combinação desses comportamentos e transições. Na Figura podem ser observadas as n distintas máquinas de estado formadas por esses módulos. Esse espaço de busca vai ser explorado por intermédio do **algoritmo de otimização** F-Race iterado [74, 75] que avalia cada configuração num conjunto de instâncias, que são os cenários nos quais os robôs realizam a tarefa. O projeto automático esta guiado por uma **função objetivo** determinada pelas especificações do usuário. Esta função objetivo deve ser definida em cada tarefa coletiva. O projeto de controlador é feito *offline* em simulação, utilizando o **simulador computacional** ARGoS [71, 72]. No simulador devem ser definidas as **condi-**

ções experimentais, como geometria do cenário, tamanho dos obstáculos, iluminação de fonte luminosa, entre outros.

Os controladores são avaliados em simulação num conjunto de cenários, em que normalmente é modificada a pose inicial dos robôs. Uma vez que o algoritmo de otimização encontra o candidato de controlador que apresenta melhor desempenho na tarefa, esse controlador escolhido é verificado em simulação e carregado nos robôs, para ser validado na tarefa no ambiente físico.

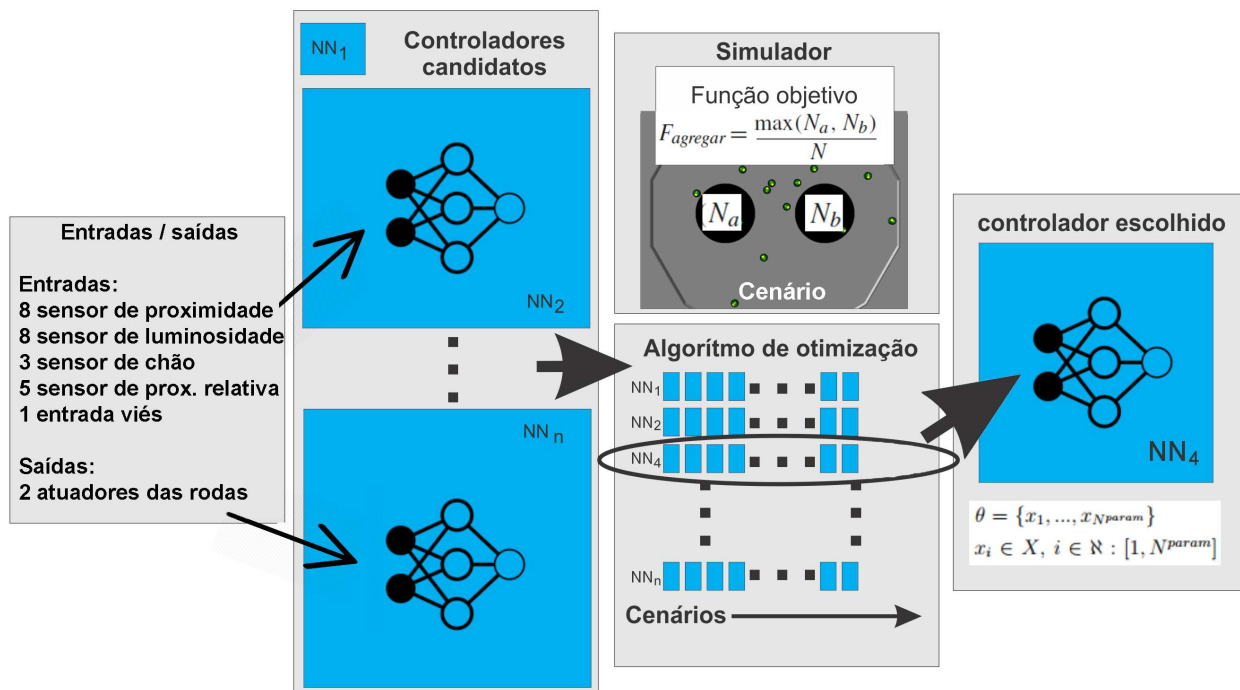
Como exemplo, se observa na Figura 3.6 o caso de síntese de controlador para robôs na tarefa de agrupamento de robôs em duas regiões circulares pretas (N_a e N_B), em que deve ser maximizada a quantidade de robôs numa das duas regiões. No início da síntese, n máquinas de estados probabilísticas finitas são produzidas, chamadas de PFSM, cada uma delas é avaliada sobre os mesmos cenários e baixo as mesmas condições iniciais. Nas sucessivas avaliações nos cenários, os controladores menos promissores são descartados e os que agregam mais robôs em alguma das regiões são preservados. Finalmente quando o número de experimentos é concluído o controlador com melhor desempenho é selecionado, no caso da figura, o controlador PFSM₄. O controlador é uma representação paramétrica, esta representação será descrita na Seção 3.8.2.

3.7.3 EvoStick

EVOSTICK é um método de projeto automático monolítico para sintetizar os controladores individuais dos robôs do enxame [18]. O método é utilizado para comparação com outros métodos de síntese automático. Nesta seção, descreve-se resumidamente o esquema de projeto do método EVOSTICK [18]. Maiores detalhes serão apresentados na Seção 3.8. O método EVOSTICK é composto de uma rede neuronal sem nós ocultos em que todas as entradas da rede são conectadas com as saídas, por intermedio de pesos [111]. A arquitetura da rede é fixa e são modificados unicamente os parâmetros (pesos) da rede neuronal. O ajuste dos parâmetros se realiza por intermédio de um algoritmo evolutivo que ajusta os parâmetros da rede de acordo com procedimentos de elitismo e mutação. Por ser um método de projeto automático, deve ser especializado da mesma forma que o método AUTOMODE. Seguindo a descrição do procedimento definida em Francesca, Birattari e Dorigo[114], o especialista deve

definir a **plataforma robótica** que vai utilizar para definir o enxame, nos experimentos realizados no método Evostick até agora são utilizados os robôs e-puck [52]. O especialista deve definir o **espaço de busca do controlador** que neste caso é composto pelas possíveis redes neurais contendo os parâmetros da rede e seus respectivos valores. O espaço de busca vai ser explorado utilizando um **algoritmo de otimização**. No caso do método EVOStICK é um algoritmo evolutivo padrão [18]. Como no método AUTOMODE, o projeto automático esta guiado por uma **função objetivo** determinada pelas especificações do usuário. Esta função objetivo deve ser definida em cada tarefa coletiva. O projeto de controlador é feito *offline* em simulação, utilizando o **simulador computacional** ARGoS [71,72]. No simulador, antes de realizar os experimentos, devem ser definidas as **condições experimentais**, como a geometria do cenário, tamanhos dos objetos, entre outros.

Figura 3.7: Projeto de síntese de controlador para robô de enxame utilizando o método EVOStICK, apresentando as principais características.



Fonte: Elaborada pelo autor.

Uma vez que o algoritmo evolutivo encontra a rede neural que apresenta melhor desempenho na tarefa, esse controlador é verificado em simulação e validado nos robôs físicos.

Como exemplo, se observa na Figura 3.7 o caso de síntese de controlador para robôs na tarefa de agrupamento de robôs em duas regiões circulares pretas (N_a e N_B), em que deve ser maximizada a quantidade de robôs numa das duas regiões. No início da síntese, n redes neurais são produzidas, chamadas de NN, cada uma delas é avaliada sobre os mesmos cenários e baixo as mesmas condições iniciais. Nas sucessivas avaliações nos cenários, os controladores que agregam mais robôs em alguma das regiões são preservados, e os controladores menos promissores são descartados e são criados novos controladores a partir dos controladores promissores. Finalmente quando o número de experimentos é concluído o controlador com melhor desempenho é selecionado, no caso da figura, o controlador NN₄. O controlador é uma representação paramétrica dos pesos da rede neuronal, esta representação será descrita na Seção 3.8.2.

A seguir, são apresentados com mais detalhes o processo de configuração que deve ser feito pelo especialista, tanto no método AUTOMODE-CHOCOLATE como no método EVOSTICK.

3.8 Especialização de AutoMoDe e EvoStick

Como mencionado na seção anterior, métodos de projeto automático devem ser especializados. Especializar um método é o procedimento para definir um conjunto de características que permitem ao método sintetizar controladores para os robôs. A especialização do método para sintetizar o controlador é realizada *a priori* ou *a posteriori* do conhecimento das tarefas coletivas que os robôs devem realizar. A seguir, é realizado um detalhamento das características que devem ser definidas para especializar o método, pela ordem de projeto:

3.8.1 A plataforma robótica

O especialista deve escolher os robôs constituintes do enxame baseado nas capacidades que são necessárias para abordar tarefas grupais. Estas características dos robôs influenciam nos comportamentos constituintes do controlador projetado. No método AUTOMODE-CHOCOLATE e no método EVOSTICK foram escolhidos os robô e-puck [52]. O robô foi apresentado na Seção 3.5.1. O robô e-puck é um robô de tração diferencial apropriado para pesquisas em enxame de robôs. O modelo de referência [18] do robô e-puck, para os métodos

AUTOMODE-CHOCOLATE e EVOSTICK é o apresentado na Tabela 3.1.

Tabela 3.1: Modelo de referência para o robô e-puck (RM1).

Tipo de entrada (sensor)	Variável	Valores
Proximidade	$v_{i_{prox}}, i \in \{1,2,\dots,8\}$	$v_{i_{prox}} \in \mathbb{R} : [0,1]$
	$\phi_{i_{prox}}, i \in \{1,2,\dots,8\}$	$\phi_{i_{prox}} \in [-\pi,\pi] rad$
Luminosidade	$v_{i_{luz}}, i \in \{1,2,\dots,8\}$	$v_{i_{luz}} \in \mathbb{R} : [0,1]$
	$\phi_{i_{luz}}, i \in \{1,2,\dots,8\}$	$\phi_{i_{luz}} \in [-\pi,\pi] rad$
Chão	$gnd_i, i \in \{1,2,3\}$	$gnd_i \in \{preto,branco,cinza\}$
Proximidade relativa	$v_{i_{R\&B}}, m \in \{1,2,\dots,20\}$	$v_{i_{R\&B}} \in \mathbb{R} : [0;0,4] m$
	$\phi_{i_{R\&B}}, m \in \{1,2,\dots,20\}$	$\phi_{i_{R\&B}} \in \mathbb{R} : [-\pi,\pi] rad$
	$n_{R\&B}$	$n_{R\&B} = \{0,1,\dots,N\}$
Tipo de saída (atuador)	Variável	Valores
Motores das rodas	$v_i, i \in \{l,r\}$	$v_i \in \mathbb{R} : [-0,12;0,12] m/s$

Ciclo de controle: $T_a = 0,1 s$.

Fonte: Adaptação de [Francesca et al.\[18\]](#).

O modelo de referência é uma representação que define as características e capacidades da plataforma robótica, tais como os sensores e atuadores que são disponíveis no controlador em cada ciclo de controle [19]. Cada sensor (atuador) é abstraído em variáveis que serão lidas (escritas) pelo controlador para realizar o controle da plataforma. Assim, o modelo de referência permite visualizar os recursos que a plataforma disponibilizará para o software de controle [18]. A definição do modelo de referência especifica indiretamente as interações entre robôs e entre os robôs com o ambiente. Por sua vez, essa representação define os tipos de tarefas que podem ser realizadas pelo enxame de robôs quando o controlador individual é instanciado em cada robô. O modelo de referência possui relevância, pois permite comparar entre diferentes métodos de projeto [18]. Se os métodos utilizam o mesmo modelo de referência é garantido que eles utilizaram os mesmos recursos e informações dos sensores e atuadores.

Da Tabela 3.1 pode ser observado que o ciclo de controle é de $T_a = 0,1 s$, quando as variáveis são atualizadas [19]. As leituras do sensor de proximidade são armazenadas em duas variáveis: O alcance de cada sensor $v_{i_{prox}} \in \mathbb{R} : [0,1]$ e o ângulo $\phi_{i_{prox}} \in [-\pi,\pi]$ com $i \in \{1,2,\dots,8\}$. Este valor de ângulo é fixo para cada sensor i , definido pela posição desse sensor em relação ao sistema de coordenadas do robô O_c . O sensor possui distância máxima de

aproximadamente $0,03 m$ quando não detecta obstáculos e a variável retorna um valor próximo a zero ($v_{i_{prox}} = 0$). Quando o valor é próximo de ($v_{i_{prox}} = 1$) é porque o obstáculo está muito próximo do robô (menos de $0,01 m$). Da mesma forma as medidas do sensor de luminosidade são armazenadas em duas variáveis, $v_{i_{luz}} \in \mathbb{R} : [0,1]$ e $\phi_{i_{luz}} \in [-\pi,\pi]$ com $i \in \{1,2,\dots,8\}$. Um valor $v_{i_{luz}} = 1$ define a presença de uma fonte luminosa no ambiente e $v_{i_{luz}} = 0$ a ausência. As variáveis $gnd_i \in \{preto,branco,cinza\}$ com $i \in \{1,2,3\}$ armazenam os valores detectados da cor do chão. Referente ao sensor de proximidade relativa, são armazenados os valores de posição relativa de m robôs na vizinhança. As variáveis $v_{i_{R\&B}} \in \mathbb{R} : [0;0,4] m$, $\phi_{i_{R\&B}} \in \mathbb{R} : [-\pi,\pi] rad$ com $i \in \{1,2,\dots,20\}$ armazenam os valores de distância e orientação relativa com o robô m vizinho. Por sua vez, $n_{R\&B} \in \{1,2,\dots,20\}$ armazenam a quantidade de vizinhos detectados pelo robô. Finalmente as variáveis $v_i \in \mathbb{R} : [-0,12;0,12] m/s$ com $i \in \{l,r\}$ são escritas com a velocidade linear da roda esquerda e direita, respectivamente.

3.8.2 O espaço de busca do controlador

AutoMoDe-Chocolate

O espaço de busca do controlador representa os possíveis controladores Θ que podem ser produzidos pela abordagem. No caso do método AUTOMODE-CHOCOLATE, esses controladores são expressos como máquina de estado finita probabilística [70], composta por módulos de comportamento, módulos de transições, cada um com os seus parâmetros específicos. O projetista deve escolher o intervalo das variáveis que serão amostradas para a geração do controlador. Este intervalo deve ser escolhido de tal forma de tentar incluir todas as possíveis soluções sem prejudicar a busca.

O espaço de busca (candidatos de controlador) é definido da seguinte forma [18]:

$$\langle \#S, S_i, S_i^p, \#N_i, N_{i,j_i}, C_{i,j_i}, C_{i,j_i}^p \rangle \quad i \in [1, \#S] \wedge j_i \in [1, \#N_i], \quad (3.13)$$

em que os controladores $\theta \in \Theta$ tem a forma da Equação (3.13).

- $\#S \in [1,4]$ é o número de estados da máquina de estados probabilística.
- $S_i \in [1,6]$ é o estado i definido como um dos seis comportamentos projetados *a priori*: *explorar*, *parar*, *fototaxis*, *antifototaxis*, *atrair* ou *repelir* (Estes comportamentos serão

descritos a seguir).

- S_i^p são os parâmetros do estado S_i caso o comportamento escolhido tenha um parâmetro configurável: α_a (se *atrair* for escolhido), α_b (se *repelir* for escolhido) ou τ (se exploração for escolhido).
- $\#N_i \in [1,4]$ é a quantidade de transições de saída do estado i .
- $N_{i,ji} \in [1,\#S]$ é o estado sucessor ji do estado i . Podem ser escolhidos: *explorar*, *parar*, *fototaxis*, *antifototaxis*, *atrair* ou *repelir*.
- $C_{i,ji} \in [1,6]$ é a condição associada a transição que une estado i com estado sucessor ji . As condições são *chão preto*, *chão cinza*, *chão branco*, *contagem direta de vizinhos*, *contagem inversa de vizinhos* ou *probabilidade fixa*. (Estas condições serão descritos a seguir).
- $C_{i,ji}^p$ são os parâmetros da condição $C_{i,ji}$ caso a condição escolhida tenha algum parâmetro configurável: β_p (se *chão preto*), β_c (se *chão cinza*), β_b (se *chão branco*) ou β_f (se *probabilidade fixa* for escolhida), η_{nc} e ε_{nc} (se *contagem direta de vizinhos* for escolhida) ou η_{anc} e ε_{anc} (se *contagem inversa de vizinhos* for escolhida).

Os módulos definidos *a priori* são tanto comportamentos, como as transições da máquina de estados (Figura 3.6). Os comportamentos representam ações que cada robô individual pode desenvolver tendo em conta seus sensores e atuadores. No método AUTOMODE são definidos seis comportamentos: *explorar*, *parar*, *fototaxis*, *antifototaxis*, *atrair* e *repelir* [18].

Explorar: Neste comportamento se um robô encontra um obstáculo ou outro robô, ele gira sobre si mesmo por um número aleatório de ciclos de controle escolhidos no intervalo $\tau \in \mathbb{N}$, $\tau : [1,100]$ então os robôs seguem em frente até encontrar outro robô ou obstáculo. Do comportamento é obtido diretamente as velocidades das rodas v_i , $i \in \{l,r\}$ (esquerda e direita).

Fototaxis: Nesse comportamento, se o robô percebe fontes de luz, ele se move em direção à luz; Caso contrário, ele se move para a frente. O comportamento produz o vetor de movimento $r_t = r_{luz} - 5 r_{prox}$. Este comportamento considera o vetor resultante de evitar obstáculos (r_{prox}). Os vetores r_{luz} e r_{prox} são definidos a seguir:

$$r_{prox} = \begin{cases} \sum_{i=1}^8 v_{i_{prox}} e^{j\phi_{i_{prox}}} & \text{if } \forall v_{i_{prox}} \geq 0, i \in \{1..8\} \\ e^{j0} & \text{cc} \end{cases}, \quad (3.14)$$

$$r_{luz} = \begin{cases} \sum_{i=1}^8 v_{i_{luz}} e^{j\phi_{i_{luz}}} & \text{if } \forall v_{i_{luz}} \geq 0, i \in \{1..8\} \\ e^{j0} & \text{cc} \end{cases}. \quad (3.15)$$

Antifototaxis: Comportamento oposto a *fototaxis*, quando o robô percebe a fonte luminosa, ele se movimenta em direção oposta à fonte de luz. Caso contrário, o robô se movimenta para a frente. O comportamento produz o vetor resultante $r_t = -r_{luz} - 5r_{prox}$.

Atrair: Nesse comportamento, os robôs se movem em direção à sua vizinhança de robôs. O vetor de direção é calculado como $r_t = \alpha_a r_{R\&B} - 5r_{prox}$. O comportamento usa o sensor de proximidade relativa para medir a presença de outros robôs (ângulo e distância relativa a esse robô contido na vizinhança). O vetor $r_{R\&B}$ é definido como:

$$r_{R\&B} = \begin{cases} \sum_{m=1}^n \frac{1}{v_{i_{R\&B}}} e^{j\phi_{i_{R\&B}}} & \text{if } m \neq 0 \\ e^{j0} & \text{cc} \end{cases}. \quad (3.16)$$

O parâmetro $\alpha_a \in \mathbb{R}$, $\alpha_a \in [1,5]$. Neste comportamento, o robô vai se atrair com os n vizinhos da vizinhança.

Repelir: Quando o comportamento *repelir* é ativado, os robôs se afastam dos robôs da sua vizinhança. O vetor de direção é calculado como $r_t = -\alpha_b r_{R\&B} - 5r_{prox}$ que incorpora a capacidade de evitar obstáculos. Como o comportamento de *atrair* explicado anteriormente, este comportamento usa o sensor de proximidade relativa para calcular a posição relativa dos robôs da vizinhança. O parâmetro $\alpha_b \in \mathbb{R}$, $\alpha_b \in [1,5]$. Neste comportamento, o robô vai se repelir dos n vizinhos da vizinhança.

Parar: Os robôs não aplicam nenhuma força no motor das rodas. Assim, ele permanece detido no cenário ($v_r = v_l = 0$).

Por sua vez, são definidas seis transições as quais são *chão preto*, *chão cinza*, *chão branco*, *contagem direta de vizinhos*, *contagem inversa de vizinhos* e *probabilidade fixa* [18].

Chão preto: Transição que é ativada quando o sensor de chão i possui um valor $gnd_i = 0$, para algum $i \in \{1,2,3\}$, com probabilidade β_p , sendo β_p um parâmetro ajustável no projeto

automático.

Chão cinza: Transição que é ativada quando o sensor de chão i possui um valor $gnd_i = 0,5$, para algum $i \in \{1,2,3\}$, com probabilidade β_c , sendo β_c um parâmetro ajustável.

Chão branco: Transição que é ativada quando o sensor de chão i possui um valor $gnd_i = 1,0$, para algum $i \in \{1,2,3\}$, com probabilidade β_b , sendo β_b um parâmetro ajustável.

Contagem direta de vizinhos: Transição que possui os parâmetros ajustáveis η_{nc} e ε_{nc} . O parâmetro $\eta_{nc} \in \mathbb{R} : [0,20]$ define o degrau da transição e $\varepsilon_{nc} \in \mathbb{N} : [0,10]$ é outro parâmetro ajustável que determina a posição deste degrau de probabilidade. A transição é ativada com probabilidade:

$$z(n) = 1/(1 + e^{\eta_{nc}(\varepsilon_{nc} - n_{R\&B})}), \quad (3.17)$$

em que $n_{R\&B}$ é o número de robôs na vizinhança do robô.

Contagem inversa de vizinhos: A transição é ativada com probabilidade $1 - z(n)$, em que $z(n)$ foi definida acima. Também possui dois parâmetros, os quais são ε_{anc} e η_{anc} , com o mesmo significado que a transição acima mencionada.

Probabilidade fixa: A transição é ativada com uma probabilidade fixa β_f .

EvoStick

O espaço de busca do controlador representa os possíveis controladores Θ que podem ser produzidos pela abordagem. No caso do espaço de busca do método EVOSTICK, esses controladores são expressos como redes neurais sem nós ocultos em que todas as entradas estão ligadas às saídas [111]. A rede é expressa da seguinte forma:

$$\begin{cases} u = \sum_{i=1}^n w_i x_i + b \\ y = g(u) \end{cases} \quad (3.18)$$

em que w_i é a matriz de ganhos (pesos) da rede, b a matriz de viés da rede. Ambas formam o espaço de parâmetros X . A matriz u é a matriz de saídas da rede e a matriz y é a matriz que aplica a cada termo de u uma função logística sigmoide. A rede tem 24 entradas e 2 saídas [18]. As entradas e as saídas da rede neuronal (Figura 3.7) são definidas em base ao modelo de referência da Tabela 3.1.

- 8 entradas do sensor de proximidade $v_{i_{prox}}$, $i \in \{1,2,\dots,8\}$, com $v_{i_{prox}} \in \mathbb{R} : [0,1]$.
- 8 entradas do sensor de luminosidade $v_{i_{luz}}$, $i \in \{1,2,\dots,8\}$, com $v_{i_{luz}} \in \mathbb{R} : [0,1]$.
- 3 entradas do sensor de chão gnd_i , $i \in \{1,2,3\}$, com $gnd_i = \{preto,branco,cinza\}$.
- 4 entradas do sensor de proximidade relativa para cada projeção do vetor $r_{R\&B}$ da Equação (3.16) com nos vetores $a_i = e^{j\pi \cdot i/4}$, $i = [-3, -1, 1, 3]$. Os vetores a_i estão representados em função de O_c .
- 1 entrada do sensor de proximidade relativa para a quantidade de robôs na vizinhança $n_{R\&B}$. O valor da entrada é obtido de $1 - 2(1 + e^{n_{R\&B}})$ (Normalização da Equação (3.17) [18]).
- 2 saídas que representam as velocidades das rodas, escaladas no intervalo $v_i \in \mathbb{R} : [-0,16; 0,16] m/s$, com $i \in \{l,r\}$ as velocidades das rodas esquerda e direita, respectivamente.

A rede neural é caracterizada por 50 parâmetros (pesos da rede) $x \in X$, $x \in \mathbb{R}$ a ser ajustados no intervalo $x : [-5,5]$.

3.8.3 O simulador computacional

Um simulador de robôs é uma ferramenta essencial usada para modelagem e criação de aplicações sem depender do robô físico. Os simuladores permitem prototipar as entidades antes de ter as plataformas físicas e incluso em ambientes controlados reduzindo tempo e custos, pois, permite depuração das aplicações sem risco de danificar o hardware das entidades. Existem vários simuladores de robôs no mercado, alguns deles são proprietários de empresas que não permitem modificação do código fonte e outros são de código livre que permite que usuários realizem modificações no software. Além disso, os simuladores podem ser especializados para alguma plataforma em particular ou ser mais gerais permitindo a modelagem de vários tipos de robôs e simular vários deles no cenário de trabalho.

Os simuladores multi-robô são ferramentas indispensáveis para a modelagem de sistemas de vários robôs operando em simultâneo no ambiente. Nestes tipos de simuladores deve existir flexibilidade para adicionar de forma simples robôs, sensores, modificação do ambiente

físico, entre outros. Além disso, os simuladores devem possuir um desempenho aceitável no tempo de execução do experimento. Este último ponto é crítico quando existem vários robôs no ambiente e incluso quando são feitas simulações exaustivas para achar a solução para um problema particular. A seguir serão apresentados os simuladores mais utilizados em sistemas multi-robô e enxames de robôs: ARGoS [71, 72], Gazebo [115] e V-REP [116].

ARGoS

Um dos simuladores mais usado na comunidade científica é o ARGoS [71, 72], especificamente usado nos métodos AUTOMODE e EVOSTICK para a simulação do enxames de robôs. Este simulador está projetado para simular centenas de robôs de diferentes tipos, tais como robôs móveis terrestres e aéreos. A arquitetura do simulador permite a execução de vários processos simultaneamente e permite dividir o espaço simulado em múltiplos sub-espacos com diferentes motores físicos funcionando em paralelo. A arquitetura do ARGoS é fortemente modular, permitindo a adição de recursos personalizados e alocação adequada de recursos computacionais. Isto permite escolher o grau de precisão da simulação e conseqüentemente os custos computacionais da mesma.

O espaço simulado do ARGoS está organizado em estruturas de dados chamadas de entidades que contém o estado completo da simulação. Estas entidades armazenam, por exemplo, a posição, orientação, dimensões, velocidade das rodas dos robôs. Cada uma destas classes está organizada em hierarquias, assim a pose do robô esta contida na classe entidade posicional, adicionando a esta classe as dimensões do robô se obtém a classe entidade física, etc. Os sensores e atuadores são módulos que permitem ter acesso ao estado dessas entidades do espaço simulado por intermédio da leitura e escrita, respectivamente.

A atualização das entidades físicas é realizada pelo motor físico do simulador que modifica os valores da classe (pose do robô). O simulador utiliza quatro motores os quais são dois para motores bidimensionais e os outros para motores tridimensionais. Estes motores podem ser usados simultaneamente em uma simulação sem solapamento, por exemplo na simulação de robôs móveis terrestres e aéreos, podem ser usados dois motores diferentes para cada tipo supondo que não existe interação física entre ambas classes de robôs. O módulo de visualização permite a leitura do espaço simulado, esse módulo pode ser em forma de texto

que permite a integração com outros softwares de análise de dados.

Outras características apresentadas no ARGoS é que permite a transferência do controlador programado na simulação para o robô físico sem modificação do código, significando redução de tempo na implementação do controlador. O simulador utiliza a linguagem C++ e outras linguagens não compiladas como LUA. O simulador permite personalizar a inicialização e finalização do experimento, por intermédio de funções definidas pelo usuário antes e depois do tempo de passo de controle.

O simulador ARGoS possui o mesmo espírito que outros simuladores tais como o Stage [117] que propõe escalabilidade no número de robôs na simulação e reusabilidade. Porém, o simulador Stage possui uma série de limitações na quantidade de motores de simulação (somente simulações de cinemática) e não podem ser simulados fielmente alguns dos tipos de coordenação de comportamentos, tais como montagem e morfogêneses, agrupação e montagem de objetos que são comportamentos que envolvem efetadores de manipulação de objetos. Outra das limitações do Stage é que os sensores desprezam o ruído que poderia existir nas medidas.

O simulador ARGoS é um simulador apropriado para sistemas com vários robôs desenvolvido no projeto Swarmanoid [25], em que são apresentadas formas de coordenação entre diversos tipos de robôs móveis e é utilizado em projetos reconhecidos da União Europeia para validação de algoritmos e estratégias em times de robôs heterogêneos com controladores distribuídos.

Gazebo

O simulador Gazebo [115] oferece a habilidade de testar em simulação algoritmos, estratégias e conceitos com alta acurácia em ambientes tridimensionais complexos. Reproduz com fidelidade ambientes e estruturas dinâmicas tais como robôs, atuadores, sensores e objetos arbitrários. Gazebo é completamente de código aberto e disponível gratuitamente com uma ampla contribuição de desenvolvedores.

A maioria dos aspectos da simulação são controláveis, desde as condições de luz a coeficientes de atrito, gravidade, etc. O Gazebo simula o hardware que é visto como tal por aplicações cliente. Os objetos podem ser simulados com massa, velocidade, atrito e

numerosos atributos que permitem o comportamento realista do robô operando no ambiente. Os robôs são estruturas dinâmicas compostas de corpos rígidos, sensores, atuadores e juntas nas quais podem ser aplicadas forças para gerar locomoção e interação com o ambiente.

As bibliotecas externas que utiliza o Gazebo são desacopladas e interatuam em baixo nível para impedir que sejam dependentes dos modelos do Gazebo, exemplo é a biblioteca para simulação da dinâmica e cinemática associada com as articulações de corpos rígidos. Este tipo de biblioteca inclui várias funcionalidades tais como detecção de colisões, especificação de malhas por intermédio de várias geometrias. Utiliza visualizadores para visualizações interativas dos modelos. Gazebo é usado com diversas interfaces externas tais como Player [118] e ROS [119]. Estas interfaces externas tratam o Gazebo como dispositivos físicos normais, assim as implementações feitas de drivers e algoritmos são intercambiáveis no simulador ou no mundo físico [115].

Gazebo é um simulador de alta fidelidade validado e usado nas competições de robótica virtual pela DARPA (Agência de Projetos de Pesquisa Avançada de Defesa). Nestas competições são utilizados robôs semi-autônomos para resolver diversas tarefas complexas, tais como locomoção e manipulação, nas quais é requerido ser simuladas características cinemáticas e dinâmicas do robô. São desenvolvidos cenários de desastre nos quais os robôs devem interagir e operar com equipamentos em condições que representam risco para seres humanos.

V-REP

V-REP é uma plataforma de simulação de robôs que tem o intuito de ser um simulador versátil e escalável [116]. Ele suporta várias linguagens de programação, como LUA, C/C++, Python, Java, Matlab, Urbi e permite a utilização de scripts para a programação de robôs, o que reduz tempo de prototipação para os usuários, pois não é necessário compilar o código a cada nova modificação no mesmo. Como os simuladores anteriormente apresentados, o V-REP permite que o usuário escolha do tipo de motor de simulação de cinemática e dinâmica das plataformas.

V-REP possui um editor de cenário por intermédio de interface que permite a importação do modelo produzido. Porém, não fornece uma forma padrão de realizar estas modificações

diretamente por intermédio de código, o que dificulta a automatização dos experimentos caso se queiram realizar vários deles em sequência [120]. O simulador possui fácil uso para os usuários, permitindo a manipulação dos objetos do ambiente simulado em tempo de execução.

Por sua vez, os modelos são armazenados como malhas de subcomponentes. Assim é possível manipular partes individuais do modelo, modificando as texturas e materiais, por exemplo. Por sua vez, os modelos dos robôs disponíveis por padrão, são extensos e documentados em detalhe. A forma de adicionar sensores, atuadores e incluso robôs é na forma de *plug-in*, o que facilita o seu uso. Possui integração com o ROS na forma de *plug-in* que permite adicionar as características positivas da comunicação de dados específicos de sensores, atuadores com os controladores na forma de publicador-subscritor entre nós.

A modelagem é uma ferramenta importante para representar os robôs no simulador. A modelagem deve representar as características físicas relevantes do robô de acordo com o experimento que deseja ser realizado. Uma representação adequada desses modelos, somada a modelos precisos de sensores e do ambiente permite que os resultados obtidos sejam mais similares com os encontrados nos experimentos com robôs físicos. Por sua vez, é desejado que a simulação dessas características tenha um desempenho aceitável no tempo de execução do experimento, tendo em conta que em cenários com vários robôs é necessário simular centenas de robôs interagindo entre eles e com o ambiente.

3.8.4 A função objetivo

A função objetivo é o indicador de desempenho, ou custo, que permite avaliar o comportamento coletivo do enxame que resolve a tarefa especificada pelo usuário. A função objetivo permite de forma artificial, colocar pressão em face de favorecer certos comportamentos tendentes a resolver a tarefa, da mesma forma que pressões existentes na natureza favoreçam a evolução das espécies [110]. Esta função matemática deve ser escolhida tendo em conta que ela será a que guiará o processo de otimização que realizará a busca do controlador individual do robô. A função objetivo permite recompensar aquele controlador individual do robô que vai na direção correta na construção do comportamento coletivo que resolve a tarefa.

O projeto automático de controladores para robô é dependente da formulação de uma função objetivo apropriada para obter o comportamento coletivo desejado [31]. É a responsável por catalogar as soluções com valores reais de desempenho e algumas das vezes é a limitante para obter um controlador de qualidade pois a função não consegue determinar características desejadas da solução entre desempenho dos controladores avaliados [31]. A forma de construir uma função objetivo depende de diversos fatores e muitas das vezes não é trivial. A experiência do projetista para modelar o problema e colocar as devidas pressões no espaço de busca de controladores é uma das chaves para a solução do problema. Embora não exista um casamento direto entre funções objetivo e um problema específico, existem várias classificações feitas na bibliografia as quais vão ser mencionadas e explicadas nesta seção. Estas classificações podem ser usadas como guias para a construção ou escolha da função objetivo de acordo com a finalidade da missão [121]. Neste trabalho serão classificadas as funções de um único objetivo. Assim, as funções objetivo podem ser classificadas de acordo ao trabalho realizado em [Floreano e Urzelai](#)[121]. Esta classificação descreve as características da função objetivo e permite a sua comparação. Na primeira descrição, categoriza as funções como sendo funcionais ou de comportamento:

Funcional - Comportamental

Esta classificação é realizada tendo em conta o tipo de pressão que é imposta no algoritmo de otimização para achar a solução. Num dos extremos, uma função objetivo funcional é aquela que coloca as pressões no ajuste do comportamento medindo tempo de subida do controlador, tempo de estabelecimento, sobrepasso, etc. É uma função objetivo que se ocupa de descrever características microscópicas de cada robô. Por outro lado, as funções objetivo não funcionais realizam uma caracterização macroscópica da tarefa e não se foca tanto no desempenho de como foi realizado esse comportamento, e sim da conclusão da tarefa [121].

No exemplo de deslocamento de um robô, colocado em [Floreano e Urzelai](#)[121], pode ser observado que uma função objetivo funcional vai descrever o funcionamento do robô desde o ponto de vista da medição da rotação das rodas. Esta função fará que o robô se desloque no ambiente, porém não será robusta as mudanças no ambiente, mas o controle do robô será o desejado. Por exemplo, o robô se deslocará sem movimentos bruscos e as respostas de

controle serão como a desejada. Por outro lado, uma função objetivo comportamental foca mais nos efeitos desse comportamento e não no funcionamento interno do mesmo. Assim, no mesmo exemplo, se quisermos obter o deslocamento do robô, uma forma pode ser definindo a função objetivo que mede tempo, distância percorrida do robô, entre outros. Assim, da mesma forma que a abordagem funcional, a função objetivo comportamental possui vantagens e desvantagens: as vantagens são que a tarefa é resolvida independentemente de como ela foi feita, aqui não interessa a causa de como o movimento foi gerado. Estas funções objetivo podem se adaptar mais facilmente as mudanças de ambiente. Porém, uma função objetivo puramente comportamental, mesmo que resolvendo a tarefa, pode ter um controlador brusco, respostas lentas, etc. Um termo médio entre estes dois conceitos é a composição de termos comportamental e funcional [31, 121].

Externa - Interna

Esta classificação é realizada tendo em conta o ponto de vista do robô. No extremo se tem a função objetivo externa que é calculada utilizando medidas que são indisponíveis para o robô, mas disponíveis para um agente externo [121]. No outro extremo, se tem a função objetivo que é calculada utilizando medidas que são disponíveis para o robô [121]. Exemplo de funções objetivo externas pode ser aquela que mede a posição relativa entre todos os robôs de um enxame, como cada robô possui capacidades de sensoriamento limitadas, não sempre eles conseguiriam realizar este cálculo. Porém, um agente externo, como um sistema de seguimento de todos os robôs do enxame poderá fazer este cálculo. Exemplos de funções objetivo internas podem ser a medida do estado da bateria, do estado dos sensores de cada robô [121]. Podem ser encontradas funções intermediárias entre as duas categorias. Geralmente as funções objetivo externas são utilizadas em projeto automático de controladores sintetizado em simulação, pois se tem todos os valores das variáveis disponíveis [121]. Este tipo de função objetivo pode ser limitante na implementação em robôs físicos, pois pode ser necessário de dispositivos custosos para a medição das variáveis ou pode ser até irrealizável esta medida [121].

Explícita - Implícita

Esta classificação é realizada tendo em conta as restrições colocadas na função objetivo. Num dos extremos, uma função objetivo explícita possui uma grande quantidade de restrições [121]. Estas funções objetivo levam a comportamentos mais desejados, pois as interações entre os robôs são definidas mais claramente. Porém, tem as desvantagens de ter o trabalho manual de decidir como devem ser essas restrições e como são colocadas na função objetivo [121]. Por outro lado, as funções objetivo implícitas possuem menos restrições, deixam o processo de otimização mais livre no que diz respeito as pressões que são colocadas na direção de certas soluções [121]. Porém, tem a desvantagem que uma tarefa pode apresentar bons desempenhos, mas os comportamentos individuais dos robôs podem não serem os desejados. Funções objetivos podem ter componentes explícitas, assim como também implícitas. Exemplo, uma função pode ser considerada mais explícita que outra quando mais restrições são colocadas para descrever a tarefa. Pode ser que com menos restrições a tarefa seja resolvida de igual forma com o comportamento coletivo, por mais que o comportamento individual de cada robô não seja o mais eficiente.

Outra classificação para a construção ou escolha da função objetivo foi descrita em [31]. Foi definida uma classificação de funções objetivo de acordo ao conhecimento embarcado na função necessário para produzir os controladores e logo é feita a revisão bibliografia de vários trabalhos que são catalogados nessa classificação. São citados trabalhos que utilizam a abordagem de robótica evolutiva e em geral são trabalhos que consideram um único robô. Porém, a classificação é igualmente valida para sistemas multi-robô e enxames de robôs. A classificação em Nelson, Barlow e Doitsidis[31] não é totalmente ortogonal nas suas definições e, por sua vez, possui outro foco de classificação comparado com [121]. A classificação é ordenada de maior nível de conhecimento *a priori* até o nível mais baixo de conhecimento *a priori* embarcado na função objetivo como descrito abaixo:

Funções objetivo de treinamento de dados

As funções de treinamento de dados são as funções objetivo entendidas como a comparação de medidas de um conjunto de dados de treinamento conhecido e o conjunto de saídas. A ideia é minimizar o erro entre esses dois conjuntos. Estas funções são muito usadas em

aprendizado de máquina para ajustar os parâmetros de redes neurais, por exemplo. São as funções que mais introduzem informação *a priori* para formular a função.

Funções objetivo comportamentais

Diferente de [121], as funções comportamentais aqui classificadas tem em conta variáveis que podem ser medidas pelos robôs, baseadas no seu comportamento individual, como entradas de sensores ou saída de atuadores [31]. Aqui é medido como o robô está se comportando e não o que está fazendo. Exemplos destas funções pode ser a medida de quantidades relacionadas com o baixo nível de bateria do robô, como medidas de presença de objetos com o sensor de proximidade, medida de intensidade de fonte luminosa com sensores de luz, etc. Esta definição utiliza o mesmo nome das funções que em Floreano e Urzelai[121], mas possuem significados diferentes.

Funções objetivo incremental funcional

Diferentemente que em Floreano e Urzelai[121], aqui são classificadas as funções objetivo que podem ser modificadas durante o processo de síntese de controlador. Este tipo de função pode ser utilizado para descrever tarefas complexas para a obtenção de um controlador final. Aqui as restrições são colocadas na função objetivo e no espaço de busca de controladores. Pode ser aplicada quando uma tarefa complexa a ser realizada pode ser decomposta em várias tarefas mais simples. Por exemplo, a tarefa de transporte de objetos pode ser decomposta em duas funções objetivo, uma para recompensar movimentos em linha reta, evitando obstáculos e outra para atingir um destino final [31].

Funções objetivo ajustada

Estas funções objetivo são compostas de termos de agregação (será explicado a seguir) que medem diretamente como a tarefa está sendo realizada e possui também termos que medem como o robô está se comportando para realizar essa tarefa [31]. Estas funções objetivo se encontram em um ponto intermediário da classificação Funcional - Comportamental realizada em Floreano e Urzelai[121].

Funções objetivo incremental do ambiente

As funções objetivo neste grupo aumentam a complexidade do ambiente em que os robôs realizam as tarefas [31]. Por exemplo, são adicionados mais obstáculos, modificadas as condições do ambiente entre uma e outra evolução do controlador. Exemplos podem ser a busca de objetos por parte dos robôs aumentando o grau de dificuldade para achar eles, tarefas de navegação com aumento de dificuldade pela adição de obstáculos, etc. Foram feitos poucos trabalhos utilizando este tipo de função objetivo [31].

Funções objetivo competitiva e co-competitiva

Este tipo de função objetivo mede a competição de grupo de robôs que compartilham o cenário de trabalho em ambientes com recursos que eventualmente são compartilhados. Aqui aconteceram interferências entre robôs e uma tarefa que esteja realizando um robô pode interferir no desenvolvimento da tarefa de outro robô, prejudicando o seu desempenho. Por outro lado, as funções objetivo co-competitiva medem o desempenho de dois conjuntos de robôs que executam duas tarefas diferentes que competem no mesmo ambiente [31]. Exemplo destas funções podem ser a tarefa de predador-presa, jogos de futebol de robôs e capturar a bandeira, etc.

Funções objetivo de agregação de dados

Estas funções objetivo, como as funções objetivo comportamentais em [Floreano e Urzelai\[121\]](#), medem o desempenho do que o robô está fazendo e não o desempenho do comportamento do robô. Estas funções de agregação de dados foram definidas como funções objetivo comportamental em [Floreano e Urzelai\[121\]](#). Assim, elas focam mais nos efeitos do comportamento e não o funcionamento interno do mesmo. Neste grupo se encontram as medições diretas de como está sendo realizada a tarefa, tais como a distância para um objetivo, o tempo que o robô demora em chegar nele, quantidade de itens coletados, podem ser alguns exemplos.

Finalmente, outra descrição pode ser realizada baseado em [Nelson, Barlow e Doitsidis\[31\]](#) em relação com a topologia da função objetivo, assim as funções objetivo se dividem em **funções de minimização ou de maximização**, em que o algoritmo de otimização deve

minimizar custo ou maximizar desempenho da função objetivo. As funções objetivo podem ser **lineares ou não lineares**, na qual são definidas tarefas em que a entrada das variáveis modificada o desempenho de forma proporcional ou o desempenho segue uma relação não linear com as variáveis de entrada. As funções objetivo podem ser **medidas em um tempo específico do experimento**, por exemplo, no final do experimento, para realizar a pressão nesse ponto específico, não se está interessado como a tarefa é desenvolvida num intervalo de tempo e sim particularmente o valor num ponto específico. Por sua vez, podem ser **medidas num intervalo de tempo determinado**, definidas como integrais (somas) em intervalos de tempo contínuo (discreto). Nesta definição é de interesse conhecer como a tarefa é mantida nesse intervalo de tempo. Também, podem ser **medidas a partir de um evento específico no experimento**, por exemplo, um robô deposita um objeto numa região, um estímulo de uma fonte luminosa é detectado pelo robô, etc. Ademais, as funções objetivo podem ser **definidas a troços dentro do experimento** em que num intervalo de tempo (ou num evento) a função muda de morfologia no mesmo experimento. Finalmente nesta classificação, a função pode ser **definida com várias formas entre gerações (instâncias)**, mudando a sua morfologia a cada experimento.

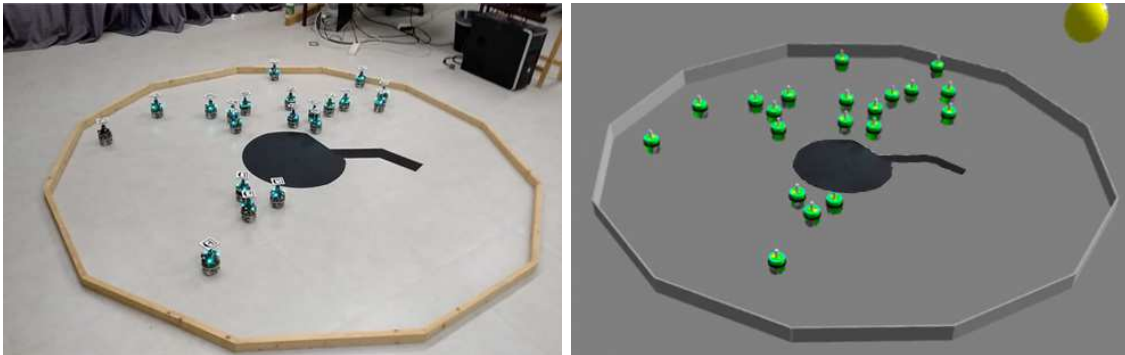
3.8.5 Condições experimentais

Selecionar as condições do ambiente para um experimento específico é uma tarefa muito importante, pois serão definidas condições controladas no ambiente de laboratório para testar a hipótese de trabalho. Assim, devem ser selecionadas cuidadosamente as condições do ambiente no qual o enxame desenvolverá a tarefa, tais como as dimensões e geometria do cenário, dos obstáculos, tipo de superfície onde os robôs vão se deslocar, tipo de iluminação do ambiente, entre outros.

Estas representações que afetam diretamente o projeto do controlador devem ser modeladas de acordo com as necessidades, replicando o ambiente real em que os robôs serão instanciados. Abaixo, na Figura 3.8 (esquerda) é apresentado um exemplo de cenário real em que os robôs devem realizar uma tarefa coletiva. A tarefa coletiva é agregar os robôs na região preta da arena. O cenário é uma região dodecagonal de $4,91 m^2$ e paredes de madeira de $0,07 m$ de altura que limitam a extensão da mesma e impede os robôs ir além

desse perímetro. O cenário possui uma região preta no centro da arena, uma fonte luminosa (parte superior direita da imagem) e 20 robôs.

Figura 3.8: Ambiente real (esquerda) e ambiente simulado (direita) com as características de interesse modeladas no cenário.



Fonte: Elaborada pelo autor.

Como pode ser observado na Figura 3.8 (direita) foi modelada a mesma geometria do cenário, possuindo as mesmas dimensões que o cenário real. As paredes do cenário foram modeladas como prismas retangulares que impedem os robôs saírem do perímetro. Foi representado a cor cinza e a região preta do chão com o mesmo efeito que no cenário real e respeitando a localização da região. A fonte luminosa foi modelada neste exemplo, pois é de interesse no experimento. A fonte luminosa esta representada pela esfera circular amarela na Figura 3.8 (direita), na parte superior direita da imagem, respeitando a mesma posição e modelando a intensidade como a observada no cenário físico. A iluminação do ambiente não é modelada, considerando que os robôs podem filtrar-la. Finalmente, os robôs foram posicionados como seus análogos reais para o começo da tarefa.

Dos experimentos analisados na bibliografia em que os métodos AUTOMODE e EVOSTICK são comparados, as condições experimentais são as mesmas, no que diz respeito ao cenário e a sua modelagem, como pode ser observado nos experimentos realizados em [Francesca et al.](#); [Francesca et al.](#); [Hasselmann, Robert e Birattari](#); [Kuckling et al.](#)[18–20,67].

3.8.6 O algoritmo de otimização

O algoritmo de otimização é um procedimento que explora o espaço de busca de controladores Θ para decidir qual é a configuração $\theta \in \Theta$ mais apropriada para resolver uma tarefa dada. Em AUTOMODE, é realizada a minimização da função de custo do sistema $C(\theta)$, avaliada num conjunto de instâncias amostradas de um conjunto de instâncias indistinguíveis [122].

$$\bar{\theta} = \arg \min_{\theta \in \Theta} C(\theta). \quad (3.19)$$

AutoMoDe

No método AUTOMODE o critério a ser minimizado é o valor esperado custo da solução encontrada pelo controlador θ nas k instâncias nas quais foi avaliada θ , uma estimativa a esse valor está dado na Equação (3.20). Esses valores são armazenados numa matriz de valores C .

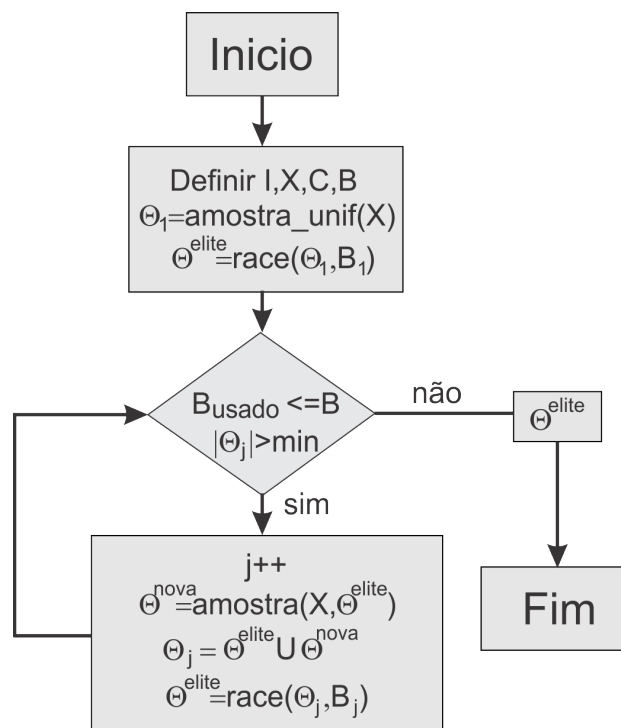
$$C_k^\theta = \frac{1}{k} \sum_{j=1}^k C_j(\theta), \quad (3.20)$$

O algoritmo de otimização cataloga as configurações com valores que representam o desempenho (custo) da configuração em uma instância específica. A catalogação é feita pela função objetivo, que vai guiar o processo de otimização. Em geral, no processo do algoritmo, as configurações menos promissoras são descartadas e as de melhores desempenho são preservadas. Por sua vez outras configurações são criadas para as sucessivas iterações do algoritmo. Essas configurações são geradas em função das configurações de melhor desempenho para as sucessivas iterações do algoritmo. A geração de configurações pode ser feita por intermédio de procedimentos de recombinação, mutação ou critérios estatísticos, entre outros. Finalmente quando o número de experimentos é concluído a configuração com melhor desempenho é selecionada.

Existem vários tipos de algoritmos para realizar a busca da configuração. No caso específico do método AUTOMODE-CHOCOLATE, é utilizado o F-Race iterado [73, 74]. F-Race é um método de configuração automática de parâmetros para a escolha da melhor configuração que satisfaz especificações definidas pelo usuário. Em AUTOMODE, F-Race é

um algoritmo que avalia os candidatos de controlador do conjunto Θ e descarta aqueles que têm baixo desempenho, baseado em critérios estatísticos. O critério estatístico usado é o teste de Friedman não paramétrico de análise de variância por classificação (ou *ranks*) [123]. A eliminação de candidatos de controlador acelera o processo e permite avaliar os controladores mais promissores em maior quantidade de instâncias para obter estimativas mais confiáveis do melhor controlador para a tarefa dada.

Figura 3.9: Fluxograma do algoritmo F-Race iterado



Fonte: Elaborada pelo autor.

Na Figura 3.9 é apresentado um fluxograma do F-Race iterativo fundamentado em López-Ibáñez et al.[75]. Em linhas gerais, o especialista deve definir um conjunto de instâncias $I \in \{I_1, \dots, I_{|I|}\}$ nas quais as configurações $\theta \in \Theta$ serão avaliadas. No método AUTOMODE, as instâncias se definem como os cenários de trabalho, nos quais os robôs do enxame desenvolvem as tarefas. Por sua vez, as configurações se definem como os controladores individuais de cada robô do enxame que permite executar as ações de cada robô. Os cenários são amostrados do conjunto de cenários $I \in \{I_1, \dots, I_{|I|}\}$. Por sua vez, devem ser definidos

os parâmetros $x_i \in X$, $i \in \mathbb{N} : [1, N^{param}]$, constituintes dos controladores dos robôs. Estes parâmetros vão ser ajustados no processo de otimização, com N^{param} o número de parâmetros do controlador. Cada controlador é parametrizado, definido como $\theta = \{x_1, \dots, x_{N^{param}}\}$, com $x_i \in X$, $i \in \mathbb{N} : [1, N^{param}]$. Uma vez definidos os parâmetros, deve ser definidos o intervalo de valores de cada variável.

O método AUTOMODE-CHOCOLATE possui 133 parâmetros. Alguns destes parâmetros são condicionais, então, normalmente é uma quantidade menor de parâmetros a ajustar. Por exemplo, se o controlador é uma máquina de estados de um estado, sem transições de saída, a quantidade de parâmetros a ajustar é definir qual é o estado (1 parâmetro) e definir os possíveis valores de variáveis definidas dentro desse estado (no método AUTOMODE os comportamentos possuem no máximo uma variável parâmetro). Então, neste caso específico é necessário ajustar dois parâmetros.

Da Equação (3.13) os parâmetros são: o número de estados da máquina de estados $\#S$ (1 parâmetro), a definição de cada um desses $\#S$ estados (4 parâmetros), a definição de cada um dos parâmetros S_i^p do estado S_i caso existissem (12 parâmetros), a definição da quantidade de transições de saída $\#N_i$ do estado i (4 parâmetros), a definição do estado $N_{i,ji}$ sucessor ji do estado i (16 parâmetros), a definição da condição associada $C_{i,ji}$ a transição que une estado i com estado sucessor ji (16 parâmetros) e a definição dos parâmetros $C_{i,ji}^p$ da condição $C_{i,ji}$ caso existissem (80 parâmetros). A determinação de todos esses parâmetros define uma única configuração θ que tem a forma da Equação (3.13).

Os parâmetros $x_1, \dots, x_{N^{param}}$ são amostrados no início do algoritmo, de acordo a seus intervalos de valores possíveis. A amostragem utilizada é uma amostragem aleatória para cada parâmetro. Cada parâmetro tem associada uma distribuição independente dos outros parâmetros. A distribuição para parâmetros numéricos (inclusive variáveis ordinais) é uma distribuição normal truncada e para o caso de parâmetros categóricos é uma distribuição discreta [75].

Logo, é definida uma métrica de custo, a função objetivo, que é o critério que permite definir a qualidade dos controladores $\theta \in \Theta$. Estes desempenhos dos controladores são armazenados numa matriz de custo C . Finalmente, deve ser definido a quantidade de experimentos B que vão ser realizados no projeto de controlador. Cada experimento consiste na avaliação de uma configuração num cenário particular. Usualmente B é escolhido

entre $B = 50.000$ e $B = 200.000$ experimentos para a obtenção do controlador de melhor desempenho.

Uma vez amostrados todos os parâmetros de todos os controladores, o espaço de busca de controladores Θ é obtido. Para cada iteração ("Race") do F-Race, se define a quantidade máxima de experimentos B_j que essa iteração vai utilizar. Essa quantidade de experimentos B_j , $j \in \mathbb{N} : [1, N^{iter}]$, depende da quantidade de iterações N^{iter} do algoritmo F-Race iterado, a iteração atual $j \in \mathbb{N} : [1, N^{iter}]$ e a quantidade de experimentos realizada até um momento determinado do algoritmo (B_{usado}) [75].

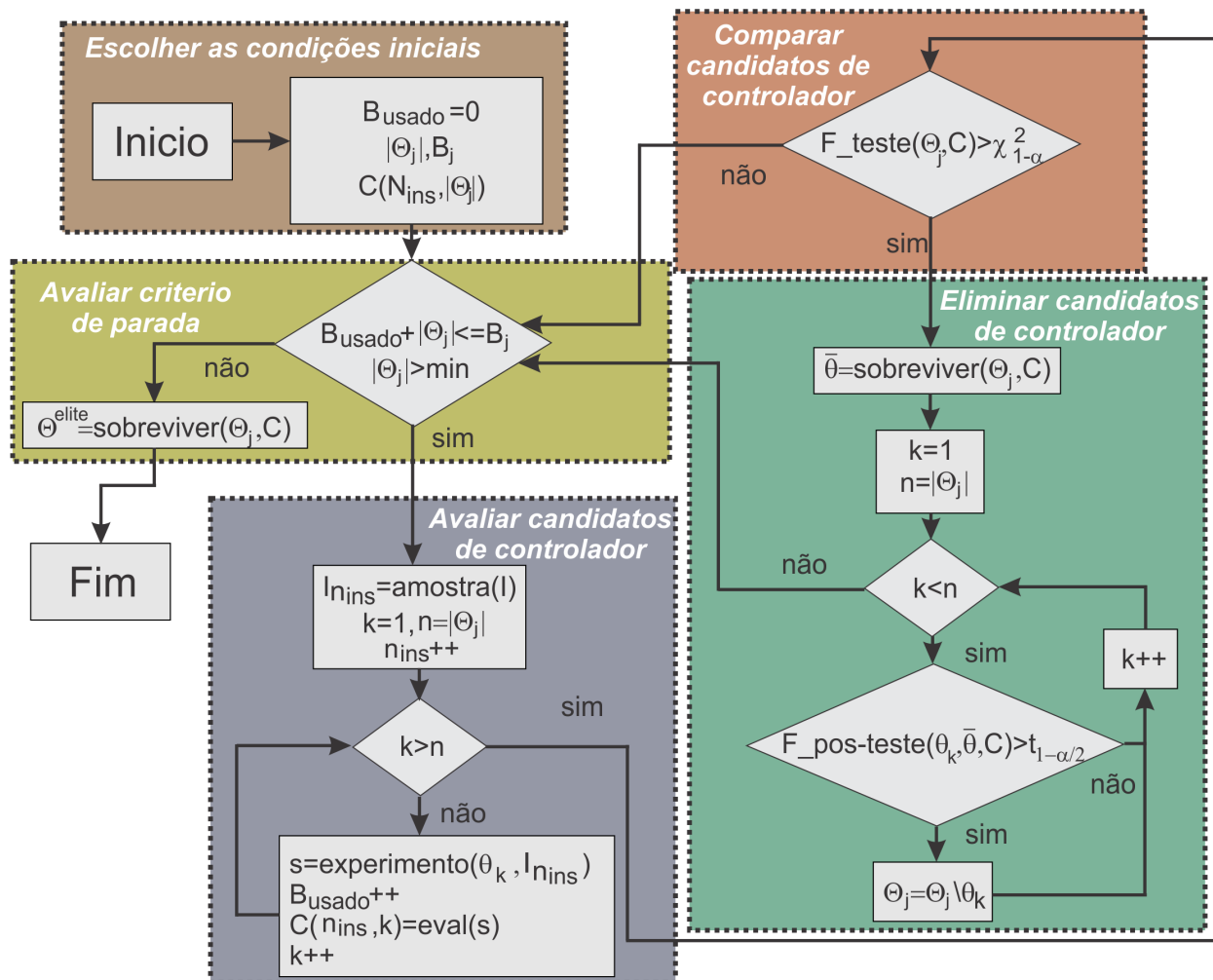
Logo, os controladores são avaliados nos cenários, por intermédio do algoritmo F-Race. A função "race", detalhadamente explicada em Birattari[122] como o F-Race, é apresentada como fluxograma na Figura 3.10. Este fluxograma será explicado nas seguintes páginas da tese. Da avaliação realizada com o algoritmo F-Race, os controladores de melhor desempenho, Θ^{elite} , são conservados para a seguinte iteração do algoritmo. Os controladores que são estatisticamente inferiores, no que diz respeito a desempenho, são descartados. O critério de parada do algoritmo é realizado quando a quantidade, B_{usado} , de experimentos realizadas nas iterações do F-Race é maior que a quantidade B de experimentos máxima do algoritmo. Por sua vez, o algoritmo finaliza se a quantidade de configurações que vão ser avaliadas $|\Theta_j|$ é menor ou igual que uma quantidade mínima de controladores permitidos na iteração.

Dos melhores controladores, Θ^{elite} , obtidos da iteração anterior do F-Race, são obtidos os novos controladores a serem avaliados. Os novos controladores Θ^{nova} são obtidos por amostragem dos parâmetros das configurações Θ^{elite} . A amostragem se realiza utilizando a distribuição associada dos parâmetros de cada controlador $\theta \in \Theta^{elite}$: Se o parâmetro é numérico (categórico) então a amostragem é realizada utilizando uma distribuição normal truncada (distribuição discreta), respectivamente [75]. Uma vez obtida a nova configuração $\theta \in \Theta^{nova}$, cada parâmetro desta nova configuração atualiza a sua distribuição. Se o parâmetro é numérico (categórico) se modifica a média e a variância (valores de probabilidade discreta), respectivamente [75]. Por sua vez, este procedimento se repete até completar a quantidade necessária de $\theta \in \Theta^{nova}$, que depende da iteração j do algoritmo. Logo é obtido Θ_j da união dos controladores amostrados e dos controladores de melhor desempenho na iteração j do algoritmo: $\Theta_j = \Theta^{elite} \cup \Theta^{nova}$.

Na Figura 3.10 é apresentado um fluxograma do F-Race iterativo (Função "race" da Fi-

gura 3.9). Em linhas gerais, o F-Race está composto pelos blocos de **escolher as condições iniciais**, **avaliar critério de parada**, **avaliar candidatos de controlador**, **comparar candidatos de controlador** e finalmente **eliminar candidatos de controlador**.

Figura 3.10: Fluxograma do algoritmo F-Race usado para o método AUTOMODE



Fonte: Elaborada pelo autor.

Escolher as condições iniciais

Como a função "race" é utilizada dentro do algoritmo F-Race iterado, estas condições iniciais já estão definidas quando é chamada a função. São definidos os controladores iniciais Θ e a quantidade máxima de experimentos permitida B_j . O número de instâncias N_{ins} é o número máximo de instâncias que podem ser amostradas pelo algoritmo.

Avaliar critério de parada

O critério de parada do algoritmo acontece se a quantidade de experimentos restantes

$B_j - B_{usado}$ não é suficiente para completar a avaliação dos $|\Theta_j|$ controladores candidatos restantes na nova instância. Ademais, o critério de parada acontece se o número de controladores restantes $|\Theta_j|$ na iteração é menor que um número mínimo de controladores candidatos. Caso o critério de parada seja afirmativo, se escolhem os controladores Θ^{elite} em função dos controladores restantes Θ_j . Esta escolha se realiza observando a matriz de custo nas instâncias anteriores, observando a posição de desempenho do controlador candidato nas instâncias avaliadas.

Avaliar candidatos de controlador

Uma instância $I_{n_{ins}}$ é definida como um cenário particular com a geometria do cenário e condições iniciais dos robôs estabelecidas. Esta instância $I_{n_{ins}}$ se obtém por amostragem aleatória e, logo, nesse cenário são avaliados os $n = |\Theta_j|$ controladores: é realizado um experimento no simulador, na instância $I_{n_{ins}}$, com cada configuração $\theta_k \in \Theta_j$, obtendo a solução s do problema que retorna um custo $C(n_{ins}, k)$ quando é avaliada na função objetivo.

Uma vez realizada a avaliação de todos os candidatos de controlador Θ_j na instância $I_{n_{ins}}$, se observa os valores de T_{first} e T_{each} [75]. Se o primeiro teste estatístico de Friedman $F_{teste}(\Theta_j, C)$ ainda não foi realizado, devem ser realizadas T_{first} avaliações em cada configuração para realizar o teste estatístico de Friedman $F_{teste}(\Theta_j, C)$. Caso contrário, se o primeiro teste estatístico foi realizado, novos testes são realizados mais frequentemente, a cada T_{each} instâncias [75].

Comparar candidatos de controlador

A comparação entre controladores candidatos se realiza por intermédio do teste $F_{teste}(\Theta_j, C)$ para determinar se existem diferenças entre os controladores Θ_j . O teste escolhido no método AUTOMODE-CHOCOLATE é o teste não paramétrico de Friedman [123]. O teste de Friedman é um teste estatístico não paramétrico, pois não se realizam suposições sobre o tipo de distribuição dos dados. Por sua vez, permite trabalhar com uma quantidade de amostras (controladores) pequenas. Primeiramente, para cada instância são ordenados os controladores candidatos. O controlador de maior desempenho vai ter atribuído o valor (posição) 1, a segunda o valor 2, etc. Em caso de empates é usado o valor médio dessas posições. Assim, a posição de uma configuração j numa instância l se calcula como $R_{lj} = pos(\theta_j)$, e a soma de posições de uma mesma configuração se calcula como $R_j = \sum_{l=1}^k R_{lj}$. O teste de Friedman

considera o estatístico da Equação (3.21) [74, 122, 123].

$$T = \frac{(m-1) \sum_{j=1}^m \left(R_j - \frac{k(m+1)}{2} \right)^2}{\sum_{l=1}^k \sum_{j=1}^m R_{lj}^2 - \frac{km(m+1)^2}{4}}, \quad (3.21)$$

em que a hipótese nula é que todos os candidatos de controlador são de desempenho similares e por isto não existe eliminação de candidatos de controlador, porque não temos a evidência estatística suficiente em contra dessas candidatos de controlador. O estatístico T possui uma distribuição aproximada $\chi^2(m-1)$ [123]. Se o valor de T é maior que $1 - \alpha$ quantil, em que α é o grau de significância do teste, a hipótese nula é rejeitada com esse grau de significância, quer dizer que existem controladores que possuem pior desempenho que outros controladores e devem ser eliminadas.

Eliminar candidatos de controlador

Se a hipótese nula do $F_{teste}(\theta_j, C)$ é rejeitada são realizadas comparações entre pares de controladores candidatos, são realizado pós-testes de Friedman [74, 122, 123], entre o melhor candidato de controlador $\bar{\theta}(\theta_j)$ encontrado até esse momento em θ_j e os outros θ_k . O estatístico utilizado é apresentado na Equação (3.22).

$$t = |R_j - R_h| \left(\frac{2k \left(1 - \frac{T}{k(m-1)} \right) \left(\sum_{l=1}^k \sum_{j=1}^m R_{lj}^2 - \frac{km(m+1)^2}{4} \right)}{(k-1)(m-1)} \right)^{-1/2}, \quad (3.22)$$

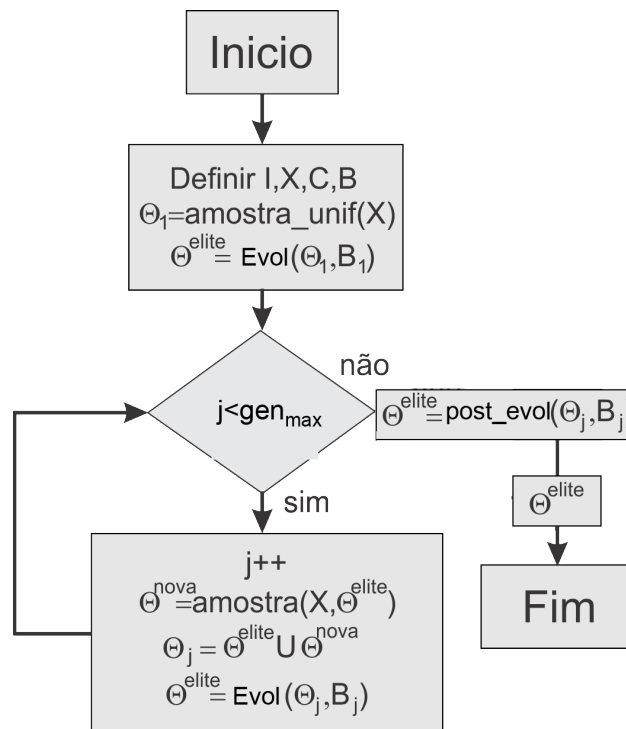
em que o estatístico t possui uma distribuição t-Student, $t(m-1)$ [123]. Assim, se o valor de t é maior que o quantil $1 - \alpha/2$ da distribuição t-Student, se elimina a configuração de menor desempenho (θ_h) e se realiza novamente este mesmo teste, comparando a melhor configuração com todos os outros controladores sobreviventes. Eliminar um candidato de controlador θ_k é realizado por intermédio da operação $\Theta_j = \Theta_j \setminus \theta_k$.

EvoStick

No método EVOSTICK o algoritmo de otimização utilizado para ajustar os parâmetros da rede neuronal foi um algoritmo evolutivo padrão [18]. Na Figura 3.11 se apresenta um fluxograma do algoritmo. As condições iniciais do algoritmo são as mesmas que para F-Race.

Neste caso, Θ_j são os candidatos de controlador do robô em forma de rede neuronal na geração j . A evolução dos controladores é feita na função "evol" que representa a evolução de uma geração de controladores, cada controlador é avaliado B_j vezes em cada geração j , $B_j = 10$. Em cada geração j existe o processo de escolha dos candidatos de melhores desempenho Θ^{elite} . Por sua vez, são escolhidos novos controladores candidatos Θ^{nova} por intermédio de processos de mutação de todos os parâmetros da rede. Na próxima geração do algoritmo começa com os candidatos de controlador $\theta \in \Theta^{elite} \cup \Theta^{nova}$.

Figura 3.11: Fluxograma do algoritmo de robótica evolutiva usado para o método EVOSTICK



Fonte: Elaborada pelo autor.

O algoritmo possui as seguintes características:

- A população inicial é de $\Theta_1 = 100$ candidatos de controlador aleatoriamente produzida, na função "amostra_unif", amostrando cada um dos 50 parâmetros no intervalo $[-5,5]$.
- Em cada geração do algoritmo os candidatos de controlador são avaliados $B_j = 10$ vezes em diferentes instâncias do cenário (diferentes condições iniciais dos robôs no ambiente), utilizando a função "evol". Este procedimento decide os melhores candidatos de controlador por geração. Normalmente, são escolhidas entre $j = 50$ a $j = 150$ gerações.

- Os melhores 20 candidatos de controlador (incluídos em Θ^{elite}) são mantidos para as próximas gerações e outros 80 (incluídos dentro de Θ^{nova}) são produzidos por intermédio de mutações de cada parâmetro usando uma distribuição normal $\chi \sim N(\mu, \sigma^2)$ de média $\mu = 0$ e variância $\sigma^2 = 1$ [18].
- Uma pós-avaliação é realizada (função "post_evol") tendo em conta os melhores candidatos de controlador. Normalmente, essa pós-avaliação avalia $B_j = 500$ vezes cada candidato de controlador.

3.9 Considerações finais

Neste capítulo foi definido o conceito de sistemas multi-robô com as principais características. Logo após, foi definido o conceito de enxames de robôs e foram classificadas as características positivas destes sistemas. A definição de comportamentos primitivos dos robôs e a forma de composição desses comportamentos foi apresentada. Esta definição é importante para explicar como serão compostos os controladores que serão utilizados nos capítulos subsequentes.

Após, foram definidos e classificados os comportamentos coletivos em enxames de robôs. Esta classificação permite dividir os trabalhos realizados na robótica de enxame e permite centrar a nossa contribuição principalmente nos comportamentos de formação de padrões, em que a coesão é relevante na resolução da tarefa coletiva. Logo, foram apresentados os robôs que são utilizados frequentemente em enxames de robôs, centrados nos robôs que serão utilizados nesta tese.

Foram definidos conceitos sobre os controladores do enxame de robôs. Assim, os controladores foram subdivididos em controladores de baixo nível e controladores supervisores. Esta divisão do controlador permite definir as arquiteturas presentes nos capítulos subsequentes.

Logo, foram definidos os tipos de sínteses de controladores para robôs de enxame e foram definidos dois métodos de projeto automático de controladores para robôs do enxame. As definições anteriores possuem relevância para compreender os métodos que são comparados na tese e observar as vantagens e limitações de cada método.

Capítulo 4

AutoMoDe-Chocolate e EvoStick

Neste capítulo são comparados, por intermédio de simulação, dois métodos de projeto automático: AUTOMODE-CHOCOLATE [18] e EVOSTICK [18]. O objetivo é observar o desempenho de ambos métodos, principalmente em tarefas com restrições de distâncias.

Por sua vez, foram propostas mais três especializações de CHOCOLATE. As especializações incluem novos sensores e/ou modificação do controlador de baixo nível dos robôs. As três novas especializações foram desenvolvidas para observar se a modificação do poder de representação do método é suficiente para abordar tarefas nas quais a coesão de robôs é relevante para a correta solução da tarefa coletiva.

A primeira especialização, chamada de CHOCO LLC, utiliza uma modificação da especialização de CHOCOLATE. Esta modificação propõe a mudança do controlador de baixo nível do robô. O controlador de baixo nível que será utilizado no robô foi proposto em [Turgut et al.\[46\]](#). Este controlador tem a vantagem que permite a rotação do robô no seu próprio eixo, diferentemente da especialização de CHOCOLATE, que impede realizar este tipo de movimentos. Assim, no método CHOCO LLC, os vetores de direção dos comportamentos primitivos são enviados para este controlador de baixo nível, que fará a redução do erro de ângulo de guinada e enviará os comandos para os atuadores do robô.

A segunda especialização, chamada de CHOCO CAM, modifica o sensor que mede a proximidade com os robôs vizinhos. É proposta a utilização da câmera omnidirecional do e-puck (Seção 3.5.1) para a captura da presença dos robôs vizinhos. Esta câmera tem a vantagem de possuir medidas mais precisas, estáveis e independentes do estado da bateria do robô, em relação às medidas do sensor de proximidade relativa com os vizinhos. Assim, no

método CHOCOCAM, cada um dos comportamentos primitivos, ao invés de utilizar o sensor de proximidade relativa entre robôs proposto em [Gutierrez et al.](#); [Gutierrez et al.](#)[103, 104], utilizam a câmera omnidirecional do robô e-puck.

A terceira especialização, chamada de CHOCAMLLC, modifica tanto o sensor de presença de robôs vizinhos como o controlador de baixo nível do robô. Esta modificação foi realizada para observar os efeitos combinados de um controlador que permite rotação no próprio eixo do robô com um sensor ainda não utilizado nas especializações do método AUTOMODE.

Com estas três novas especializações do método AUTOMODE, se observará se estas modificações no poder de representação do controlador são suficientes para permitir comportamentos de coesão nos robôs.

A hipótese de trabalho é que os métodos AUTOMODE e EVOSTICK possuem limitações para resolver tarefas, com restrições de distâncias, definidas *a priori* na bibliografia, especialmente aquelas em que as distâncias relativas entre os robôs é de relevância para obter coesão e padrões de formação entre os membros do enxame. Tarefas nas quais a coesão possui relevância podem ser aquelas que são resolvidas com comportamentos coletivos tais como os comportamentos de **formação de padrões**, como foi observado na Seção 3.4.1.

Do estudo feito da bibliografia é esperado que o método EVOSTICK obtenha um melhor desempenho que o método AUTOMODE na realização das tarefas, pois o EVOSTICK é um método de projeto automático com um controlador de maior poder de representação. Este maior poder de representação está relacionado com o tipo de arquitetura imposto no método, que produz menos restrições nas interações entre os robôs e dos robôs com o ambiente. O método EVOSTICK possui uma arquitetura monolítica em rede neuronal que conecta todas as entradas com todas as saídas de forma direta. No método EVOSTICK não são feitas suposições que restrinjam fortemente as interações entre os robôs e dos robôs com o ambiente. Porém, no método AUTOMODE as restrições feitas no que diz respeito a essas interações são explícitas nos módulos *a priori* definidos pelo usuário. Pelo anterior, as relações de entrada-saída definidas nos módulos do método AUTOMODE, podem não gerar o comportamento coletivo que resolva a tarefa quando aconteça a composição dos controladores individuais que foram instanciados nos robôs. Assim, é esperado que o método AUTOMODE não consiga realizar algumas das tarefas coletivas definidas pelos usuários. Por sua vez, pode ser observado um desempenho baixo na métrica que mede a qualidade da

solução.

O presente capítulo está estruturado da seguinte forma: Na Seção 4.1 será apresentado o robô utilizado com seu modelo. Por sua vez, serão apresentados os modelos de sensores e atuadores do robô. Logo após, na Seção 4.3, será apresentada a metodologia para realizar os experimentos. Os experimentos são ordenados tendo em conta o tipo de restrição de distância colocado na tarefa coletiva. Assim, na Seção 4.4, serão apresentados os resultados experimentais em seis tarefas coletivas. Dentre essas tarefas, na Seção 4.4.1, será considerada uma tarefa coletiva que deve ser resolvida com comportamento de agrupação de objetos. Nas seguintes tarefas as restrições de distâncias se fazem ainda mais presente. Na Seção 4.4.2, será apresentado um conjunto de tarefas coletivas que devem ser resolvidas com agregação dos indivíduos do enxame. Na Seção 4.4.3, serão colocadas maiores restrições de distâncias nas tarefas coletivas. Nestas últimas duas tarefas coletivas, unicamente a coesão dos robôs permite a correta solução da tarefa. Finalmente, na Seção 4.5 serão apresentadas as considerações finais do capítulo.

4.1 Modelo de robô e sensores

Modelos são necessários no intuito de controlar um sistema, esses modelos devem ser gerais, para poder ser implementados em outros robôs. Por sua vez, os modelos devem ser simples e relevantes o suficiente para especificar o fenômeno que se deseja modelar sem adicionar complexidade desnecessária. Neste capítulo será utilizado o robô e-puck apresentado na Seção 3.5.1.

4.1.1 Modelo cinemático do robô

O modelo cinemático de uma plataforma consiste na relação entre as posições, velocidades e acelerações. O modelo de robô utilizado nos experimentos da tese é o modelo cinemático do uniciclo da Equação (3.5), apresentado novamente na Equação (4.1).

$$\begin{cases} \dot{x} = v \cos(\theta) \\ \dot{y} = v \sin(\theta) \\ \dot{\theta} = \omega \end{cases} \quad (4.1)$$

A velocidade $v \in \mathbb{R}$ é a velocidade linear do robô e $\omega \in \mathbb{R}$ a velocidade angular do robô. A pose do robô é representada com as variáveis x , y e θ representada respeito ao sistema de coordenadas O_w . Este modelo cinemático permite definir vários tipos de robôs de forma simples. Por exemplo, os robôs apresentados que possuem tração diferencial na Seção 3.5, podem ser modelados utilizando a Equação (4.1), obtendo o modelo de estados da Equação (4.2) quando $v = (v_r + v_l)/2$ e $\omega = (v_r - v_l)/L$.

$$\begin{cases} \dot{x} = (v_r + v_l) \cos(\theta) / 2 \\ \dot{y} = (v_r + v_l) \sin(\theta) / 2 \\ \dot{\theta} = (v_r - v_l) / L \end{cases}, \quad (4.2)$$

em que $L = 0,053 \text{ m}$ é o comprimento do eixo das rodas, v_r e v_l são as velocidades lineares das rodas direita e esquerda, respectivamente. Uma forma simples e intuitiva de realizar o controle no robô é usar o modelo do uniciclo (Equação (4.1)) para o controle da velocidade linear v e angular ω do robô. Logo, utilizar a Equação (4.3) na implementação no robô diferencial.

$$\begin{cases} v_r = (2v + \omega L) / 2 \\ v_l = (2v - \omega L) / 2 \end{cases}. \quad (4.3)$$

Por isto, realiza-se um mapeamento entre $\begin{bmatrix} \omega & v \end{bmatrix} \rightarrow \begin{bmatrix} v_r & v_l \end{bmatrix}$ na Equação (4.3), que foi obtida como uma igualdade da Equação (4.1) e Equação (4.2).

4.1.2 Modelo de sensores e atuadores do robô

Os modelos de sensores descrevem o processo de formação da medida gerada no mundo físico. Existem diversos sensores que mensuram diferentes magnitudes, tais como sensores de proximidade, posição, luz, som, olfativos, cor, entre outros. Na Figura 4.1 é apresentada a distância d_{ij} e ângulo ϕ_{ij} relativo entre dois robôs $i \neq j$ especificadas em função do sistema de coordenadas O_c do robô i , para o caso planar.

Os sensores de distância obtém a medida entre dois robôs por intermédio da Equação (4.4),

quando estão no campo de visibilidade do robô, definida como a distância euclidiana d_{ij} .

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \quad (4.4)$$

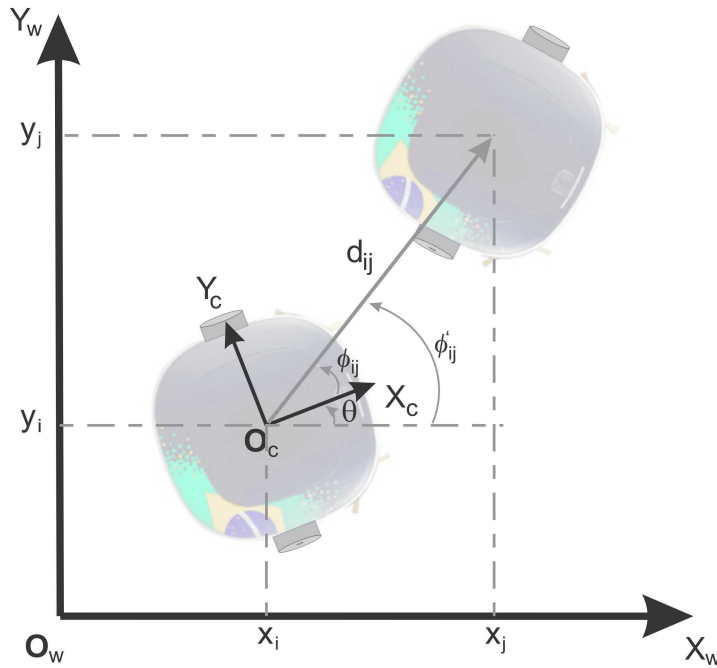
em que (x_i, y_i) é a posição do robô com relação a O_w . A posição (x_j, y_j) é a posição do robô j em relação ao sistema O_w . O ângulo $\phi_{ij} \in [-\pi, \pi]$ (Equação (4.5)) é a diferença entre o ângulo θ_i do robô i e o ângulo de detecção ϕ'_{ij} da entidade j (Equação (4.6)).

$$\phi_{ij} = \phi'_{ij} - \theta_i, \quad (4.5)$$

com ϕ'_{ij} definido como na Equação (4.6), em que $\phi'_{ij} \in [-\pi, \pi]$.

$$\phi'_{ij} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right), \quad (4.6)$$

Figura 4.1: Robô i apresentando a distância relativa d_{ij} e ângulo ϕ_{ij} aos vizinhos, $j \neq i$.



Fonte: Elaborada pelo autor.

Sensores de proximidade

O sensor de proximidade é um módulo de reflexão fotoelétrico que integra um emissor e um receptor infravermelho. Geralmente apresentam interferência com a luz visível. Este sensor transforma a reflexão do feixe em um valor de proximidade. Estes sensores permitem medir a distância de entidades próximas ao robô, geralmente os robôs possuem vários destes sensores instalados na plataforma. O valor medido do sensor de proximidade $i \in \mathbb{N} : [1,8]$ é $v_{i_{prox}} \in \mathbb{R} : [0,1]$, modelado por intermédio da Equação (4.7).

$$v_{i_{prox}} = ad_{ij}^2 + bd_{ij} + c + \sigma_{prox}, \quad (4.7)$$

em que $a = 298,7$; $b = -36,9$ e $c = 1,1$. Estas constantes foram obtidas de dados empíricos do sensor de proximidade ajustados a um modelo de segundo grau. A distância d_{ij} (em metros) neste caso é entre o sensor $i \in \mathbb{N} : [1,8]$ e o obstáculo mais próximo do sensor na linha de visão. Esta distância é calculada como na Equação (4.4) considerando que $d_{ij} \leq 0,05 m$. De acordo com dados experimentais, o ruído é considerado um ruído aleatório uniforme definido no intervalo $\sigma_{prox} \in \mathbb{R} : [-0,05; 0,05]$.

Sensores de luminosidade

O sensor de luminosidade do robô permite medir a intensidade de luz de uma fonte luminosa. Neste sensor, são consideradas oclusões da fonte luminosa com obstáculos ou robôs. Caso existissem mais de uma fonte luminosa o cálculo é feito somando o valor de cada fonte separadamente sem considerar a interferência entre fontes luminosas. O valor medido do sensor de luminosidade $i \in \mathbb{N} : [1,8]$ é $v_{i_{luz}} \in \mathbb{R} : [0,1]$, é modelado por intermédio da Equação (4.8).

$$v_{i_{luz}} = \left(e^{-\frac{2d_{ij}}{I}} \right) s_{\theta} + \sigma_{luz}, \quad (4.8)$$

em que d_{ij} é a distância, neste caso entre o sensor $i \in \mathbb{N} : [1,8]$ e o obstáculo mais próximo do sensor na linha de visão. A distância é calculada como na Equação (4.4) considerando que $d_{ij} \leq 2,5 m$. No simulador, a intensidade da fonte luminosa é medida em metros e foi definida num valor de $I = 2,5 m$ considerando a distância máxima de percepção do estímulo

luminoso no sensor físico. A ponderação $s_\theta \in \mathbb{R} : [0,1]$ permite ponderar linearmente caso os sensores não estejam diretamente posicionados na fonte luminosa. Por exemplo, um sensor diretamente posicionado em direção à fonte luminosa ($\theta_{dif} = 0$) possui um valor de $s_\theta = 1$ e um sensor a mais de $\theta_{dif} \geq \pi/2$ da fonte luminosa possui um valor $s_\theta = 0$. De acordo com dados experimentais, o ruído é considerado um ruído aleatório uniforme definido no intervalo $\sigma_{luz} \in \mathbb{R} : [-0,05; 0,05]$.

Câmera omnidirecional

A câmera omnidirecional do robô é um sensor de visão composto por uma câmera e um espelho hiperbólico. Este sensor de visão permite medir a presença de características na imagem, tais como cores de pixels, que ordenados em conjuntos de pixels permitem determinar os LEDs, led_i , de cada robô. A distância $v_{icam} \in \mathbb{R} : [0,05; 0,30] m$ do robô a cada LED é calculada utilizando a Equação (4.10).

$$v_{icam} = d_{ij} + \sigma_{cam}. \quad (4.9)$$

A modelagem da distância d_{ij} do robô para cada LED dos robôs vizinhos utiliza a Equação (4.4). Segundo dados experimentais, a modelagem considera ruído aleatório gaussiano com média nula e desvio padrão $\sigma_{cam} \in \mathbb{R} : [-0,04; 0,04] m$, considerando o sensor físico do robô e-puck. O sensor oferece uma lista de conjunto de pixels, com a cor de cada conjunto. Assim, é possível identificar a cor dos LEDs dos robôs no alcance do sensor.

Sensor de proximidade relativa

O sensor de proximidade relativa permite realizar a medida de ângulo e distância relativa com robôs da vizinhança. O sensor foi modelado da mesma forma que a câmera omnidirecional.

$$v_{iR\&B} = d_{ij} + \sigma_{R\&B}. \quad (4.10)$$

A modelagem da distância d_{ij} do robô para os robôs vizinhos utiliza a Equação (4.4). Pode ser observado das medições com os robôs físicos que o sensor possui menos precisão de medida

que o sensor da câmera omnidirecional. De acordo com dados experimentais, a modelagem considera ruído aleatório gaussiano com média nula e desvio padrão $\sigma_{R\&B} \in \mathbb{R} : [-0,1; 0,1] m$, considerando o sensor físico do robô e-puck.

Tração do robô

Como o robô possui tração diferencial, foi considerado um modelo que possui uma flutuação aleatória em cada uma das velocidades lineares das rodas do robô. Na Equação (4.11) se observa o modelo de ruído para ambos atuadores do robô.

$$\begin{cases} v_r = v'_r + v'_r \sigma_{atu} \\ v_l = v'_l + v'_l \sigma_{atu} \end{cases} . \quad (4.11)$$

Segundo dados experimentais, foi considerado um ruído aleatório uniforme definido no intervalo $\sigma_{atu} \in \mathbb{R} : [-0,05; 0,05]$.

4.2 Especialização dos métodos de projeto

Como foi estudado no Capítulo 3, os métodos automáticos para projeto de controladores precisam ser especializados. As especializações de CHOCOLATE e EVOSTICK foram definidas na Seção 3.8. A especialização de CHOCOCAM é similar à especialização de CHOCOLATE, mas substitui o sensor que realiza a medição da proximidade relativa com os robôs vizinhos. Assim, CHOCOCAM utiliza a câmera omnidirecional do robô e-puck para realizar a medida de distâncias entre os robôs. A especialização de CHOCOLLC é similar à especialização de CHOCOLATE, mas substitui o controlador de baixo nível do robô. O controlador de baixo nível será explicado na Seção 4.2.1. Finalmente, a especialização de CHOCAMLLC realiza modificações na especialização de CHOCOLATE e combina as modificações de CHOCOLLC e CHOCOCAM.

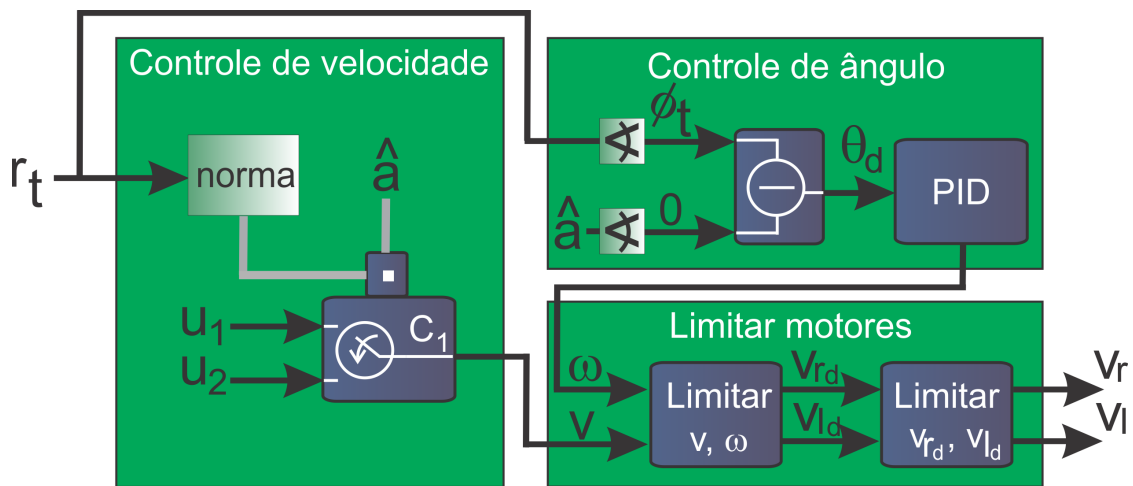
Como mencionado, tanto CHOCOLLC quanto CHOCAMLLC utilizam um novo controlador de baixo nível que permite a rotação do robô no seu próprio eixo sem necessidade de modificar sua posição atual. Por ser especializações do método AUTOMODE, o controlador supervisor de CHOCOCAM, CHOCOLLC e CHOCAMLLC é uma máquina de estados finita

probabilística como descrito na Seção 3.8.2. Na Figura 4.2 observa-se o controlador de baixo nível para CHOCOLLC e CHOCAMLLC. A saída de cada estado da máquina de estados é o vetor r_t . Este vetor r_t depende do estado particular da máquina de estados. O valor de esse vetor r_t para cada estado particular da máquina de estados se encontra definido na Seção 3.8.2.

4.2.1 Controlador de baixo nível do robô

O controlador de baixo nível permite controlar o ângulo de guinada do robô e a velocidade do robô para atingir a direção e velocidade desejada. Na Figura 4.2 observa-se a entrada do controlador como sendo o vetor resultante r_t . Este controlador basicamente realiza uma transformação $r_t \rightarrow \begin{bmatrix} v_r & v_l \end{bmatrix}$. Na Figura 4.2 apresenta-se o controle de baixo nível do robô para realizar o controle das velocidades lineares das rodas do robô.

Figura 4.2: Controle de baixo nível do robô, apresentando o controle de velocidade, ângulo e ao módulo para limitar motores.



Fonte: Elaborada pelo autor.

O módulo de velocidade do robô permite controlar a velocidade linear v do robô a partir do vetor r_t . Na Figura 4.2 observa-se a entrada r_t a qual é normalizada para $r'_t = \frac{r_t}{|r_t|}$ se $r_t > 0$, logo se realiza o produto escalar com o vetor \hat{a} que é definido como o vetor unitário que descreve a orientação do robô em relação ao sistema de coordenadas da base do robô O_c .

Logo a saída do bloco C_1 é v de valor expressado na Equação (4.12).

$$v = \begin{cases} u_1 = k_v(\hat{a} \bullet r'_t)v_{max} & \hat{a} \bullet r'_t > 0 \\ u_2 = 0 & \text{cc} \end{cases} . \quad (4.12)$$

O controle de velocidade implementado foi anteriormente definido em [Turgut et al.\[46\]](#). O produto interno permite modular a velocidade caso r'_t ou \hat{a} não tenham a mesma direção. Este controle permite que os robôs consigam manobrar com maior margem para obter a orientação de r_t de forma segura. Assim, à medida que o robô precise de mais manobrabilidade o produto interno tende a zero e $v = 0$. Quando a direção desejada r'_t é a mesma que a atual \hat{a} , não existe necessidade de manobrar e $v = k_v v_{max}$. Se esse produto interno entre a direção desejada e a direção atual é negativo, o robô é detido ($u_2 = 0$) permitindo somente rotações puras sem translações.

O módulo para controle de ângulo recebe como entrada os ângulos dos vetores r_t e \hat{a} em relação ao sistema de coordenadas O_c e é calculado o erro de ângulo θ_d que deve ser reduzido. A redução desse erro vai ser realizada por intermédio de um controlador PID (Seção 3.6.1) que possui unicamente o componente proporcional ativado. Na Figura 4.2 observa-se que a entrada do controlador PID é $\theta_d = \sphericalangle r_t - \sphericalangle \hat{a}$, tendo em conta que $\sphericalangle \hat{a} = 0$ porque esta expressado em relação ao sistema de coordenadas O_c .

Implementa-se a Equação (3.7) para os controladores PID da velocidade linear e angular do robô. O tempo de ciclo de ambos os controladores é de $T_a = 0,1 s$ e os ganhos dos controladores PID foram ajustados ativando unicamente o componente proporcional, como sendo $k_\omega = 0,7 s^{-1}$ para o controle proporcional de velocidade angular e $k_v = 1,0$ para o controle proporcional de velocidade linear do robô, pois apresentam uma boa resposta e pouco sobrepasso. A saída dos controladores é a velocidade angular e linear do robô (ω e v) que tende a corrigir o erro de guinada do robô θ_d para atingir a direção especificada por r_t .

Finalmente, na Figura 4.2 apresenta-se o módulo para limitar a rotação dos motores ao máximo valor permitido. Pode ser necessário que o robô tenha que fazer uma manobra muito brusca para satisfazer a especificação gerada pelo vetor r_t e para isso seja necessário que o motor tenha que rotacionar a uma velocidade maior que a permitida. Os motores possuem restrições que devem ser respeitadas tanto para não danificá-los assim como também para

satisfazer a especificação da melhor forma possível. O módulo permite sempre garantir a velocidade rotacional do robô ω em detrimento da sua velocidade linear v , assegurando que as velocidades lineares das rodas do robô v_r e v_l não ultrapassem a velocidade de rotação máxima.

O módulo aceita como entradas as variáveis de controle v e ω as quais serão limitadas nos valores $-v_{max} \leq v_d \leq v_{max}$ e $-\omega_{max} \leq \omega_d \leq \omega_{max}$. Logo, estas duas velocidades são transformadas em velocidades lineares das rodas v_{r_d} e v_{l_d} por intermédio da Equação (4.3). Caso alguma destas duas velocidades continue excedendo o limite imposto pelos motores de v_{max} ($-v_{max}$), se subtrai (aumenta) a diferença do excedente de velocidade em cada roda para obter v_l e v_r . Finalmente, cabe mencionar que este módulo de restrição da velocidade rotacional dos motores pode ser implementado em outros robôs com tração diferencial, sendo específico e projetado para plataformas com este tipo de restrições cinemáticas.

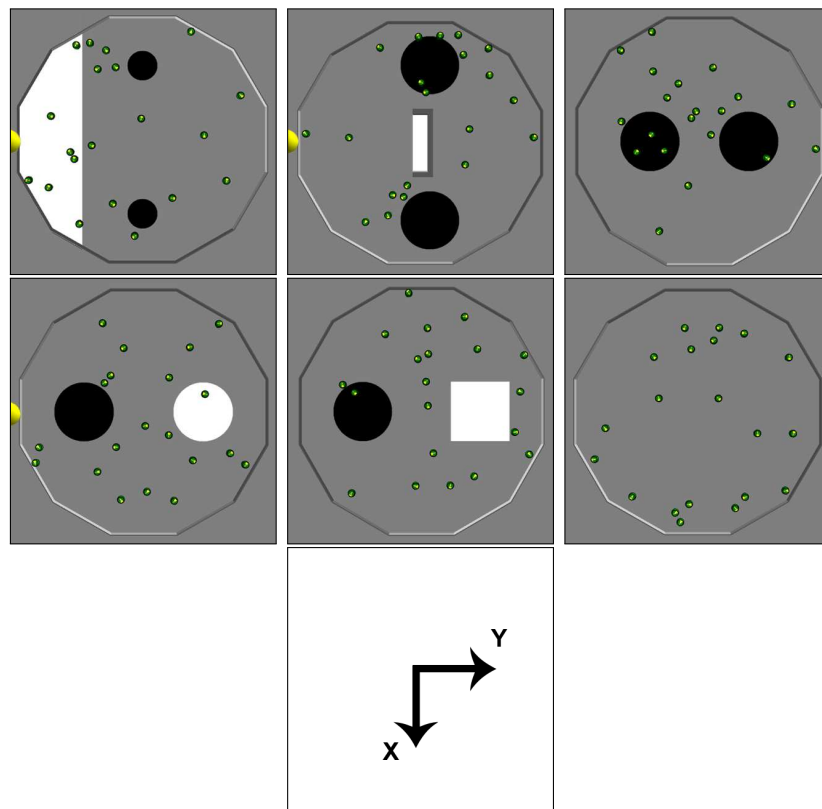
4.3 Metodologia

As missões nas quais os métodos são comparados foram apresentadas na bibliografia previamente nos artigos de [18] e [19]. Estas missões apresentam tarefas que são consideradas referência na robótica de enxame. Cada missão descreve por intermédio de uma função objetivo uma tarefa coletiva que os robôs do enxame devem realizar num determinado cenário de trabalho. A Figura 4.3 apresenta os cenários com a pose inicial dos robôs numa instância particular de cada experimento. Por sua vez é apresentado o sistema de coordenadas usado para descrever os cenários. Cabe esclarecer, que esse sistema de coordenadas somente é utilizado para descrever geometricamente os cenários e não como forma de localização global dos robôs.

As missões são: **procura de objetos, acesso restrito ao refúgio, agregação, agregação com informação no ambiente, cobertura e monitoramento de região e cobertura da maior rede do enxame**. Pode ser observado que estas missões têm por objetivo a resolução de tarefas usando principalmente comportamentos com restrições de distâncias, tais como agrupação de objetos, agregação de robôs e formação de padrões. Estes comportamentos coletivos foram especificados na Seção 3.4.1. Os cenários destas missões são modelados com o simulador ARGoS (Seção 3.8.3) e vão ser utilizados os robôs e-puck

(Seção 3.8.1).

Figura 4.3: Imagem superior dos cenários simulados. De cima para baixo e de esquerda à direita: **procura de objetos**, **acesso restrito ao refúgio**, **agregação**, **agregação com informação no ambiente**, **cobertura** e **monitoramento de região e cobertura da maior rede do enxame**. As imagens mostram também a pose inicial dos 20 robôs para cada missão. A imagem inferior apresenta o sistema de coordenadas localizado no centro de cada cenário.



Fonte: Elaborada pelo autor.

Neste capítulo são consideradas missões que priorizam o desempenho de como foi realizada a tarefa coletiva sobre a funcionalidade do comportamento individual do robô. Este tipo de função objetivo foi descrito na Seção 3.8.4. Segundo a classificação de [121] estas funções são chamadas de funções objetivo comportamental. Por outro lado, segundo a classificação de [31] estas funções são chamadas de funções objetivo de agregação de dados.

Os experimentos são ordenados tendo em conta o tipo de restrição de distância imposto na tarefa coletiva. Na primeira missão **procura de objetos**, as restrições de distâncias não são explícitas na função objetivo. Nas três missões seguintes: **agregação**, **agregação**

com informação no ambiente e acesso restrito ao refúgio as restrições são colocadas indiretamente, pois, para a missão ser resolvida é necessário que os robôs se organizem no cenário, minimizando a distância entre eles, para agrupar a maior quantidade de robôs nas regiões de agrupamento. Nas últimas duas missões, são colocadas restrições de distâncias para a existência de coesão no enxame. Na missão **cobertura e monitoramento de região**, as restrições de distâncias na função objetivo exigem que os robôs mantenham coesão, e especificamente na última missão (**cobertura da maior rede do enxame**) as restrições de distâncias exigem que os robôs mantenham conectividade antes de ser atingida a coesão.

Das missões é esperado que os métodos EVOSTICK e AUTOMODE-CHOCOLATE consigam resolver as missões que não exigem coesão entre os robôs. Porém, é esperado observar que ambos os métodos possuam dificuldades para resolver tarefas que exigem a coesão entre robôs, no que diz respeito a manter distâncias específicas entre os robôs. Para superar essas possíveis dificuldades, a comparação com outras especializações do método AUTOMODE tem o objetivo de colocar uma melhoria na especialização do método para abordar estas tarefas coletivas.

Os robôs operam num cenário dodecagonal de $4,91 m^2$. O cenário, chamado de arena, é restrito por paredes que limitam a extensão da mesma. Algumas das regiões do chão da arena são sinalizadas pela sua cor, dependendo da missão. É definido um sistema de coordenadas com a origem no centro da arena, unicamente para fins de descrição da arena. Os eixos x^+ e y^+ apontam para acima e para a esquerda da Figura 4.3, respectivamente. O tempo para resolver as missões são de $T_m = 120 s$ para os métodos comparados.

No começo do experimento, os robôs são instanciados aleatoriamente no cenário. O enxame é composto de 20 robôs e-puck. Cada método de projeto é executado 10 vezes obtendo 10 controladores para cada tarefa. Logo para cada tarefa, os controladores são avaliados unicamente em simulação e uma única vez. São utilizados 200.000 experimentos para projetar cada controlador.

No caso do método EVOSTICK são feitos a mesma quantidade de experimentos distribuídos da seguinte forma: são utilizados 100 candidatos de controle aleatoriamente gerados e avaliados cada um 10 vezes em diferentes instâncias do cenário e repetido este procedimento 150 vezes com as futuras gerações da população inicial. Logo uma pós-avaliação é feita para

obter o melhor controle. A pós-avaliação é aplicada nos 100 melhores candidatos de controle avaliando cada um 500 vezes.

A síntese dos controladores feita por cada método é realizada *off-line*. O método AUTOMODE e especializações sintetizam o controlador utilizando a metodologia da Seção 3.7.2. O método EVOSTICK sintetiza controladores como explicitado na Seção 3.7.3. Por sua vez, a síntese dos controladores é realizada num conjunto de computadores remotos, ou *cluster*, chamado de MAJORANA [124]. O *cluster* é composto de 1152 núcleos e 5 *racks*. Para uma tarefa coletiva específica, na qual queira ser sintetizado o controlador do robô, o tempo de projeto de cada controlador depende, principalmente, do *rack* utilizado e de outros trabalhos que estejam sendo executados nesse mesmo *rack* no momento da síntese. A utilização do *cluster* de computadores permite que os projetos de controlador sejam realizados em menor tempo. O mesmo controlador projetado é carregado em todos os robôs do enxame. Os experimentos são realizados por um enxame de robôs que possui as características mencionadas na Seção 3.2.1: o sistema possui controladores descentralizados, os robôs que compõem o enxame são autônomos, homogêneos e utilizam informações locais do ambiente.

Para apresentar graficamente os resultados em cada missão, são usados diagramas de caixas nos quais cada amostra representa o desempenho de um controlador num cenário amostrado aleatoriamente de um conjunto de possíveis cenários. Os diagramas de caixas permitem observar a mediana e a dispersão dos resultados por intermédio dos quartis. É realizado o teste estatístico de Wilcoxon pareado para comparar os métodos em cada missão, a significância do teste é 95 % [123]. O teste é um teste não paramétrico, pois não é necessário assumir nenhuma hipótese sobre a distribuição de probabilidade da população da qual são retiradas as amostras. É utilizado o teste de Wilcoxon, pois de cada método foram obtidas pequenas quantidades de amostras (10 controladores por método). Nestas situações não é possível trabalhar com a abordagem paramétrica [123]. Por sua vez, este teste permite ser menos sensível a valores atípicos ou extremos, pois utiliza classificação de dados (*ranks*), sendo a hipótese do teste realizada em função da mediana dos dados. É utilizado o teste pareado pois cada método é avaliado, em pares, nas mesmas condições iniciais.

4.4 Resultados experimentais

4.4.1 Comportamento de agrupação de objetos

Procura de objetos

A missão tem o objetivo de realizar agrupamento de objetos encontrados em regiões específicas da arena. A função objetivo que deve ser maximizada é

$$C(\theta) = n_o, \quad (4.13)$$

em que n_o é a quantidade de objetos que são depositados corretamente na região branca no final do experimento no instante de tempo T_m . A função objetivo que deve ser maximizada é a quantidade de objetos depositados na região branca da arena. Assim, quanto maior o valor, maior a quantidade de objetos coletados nessa região. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 1 Cálculo da função objetivo para a missão **procura de objetos**

```

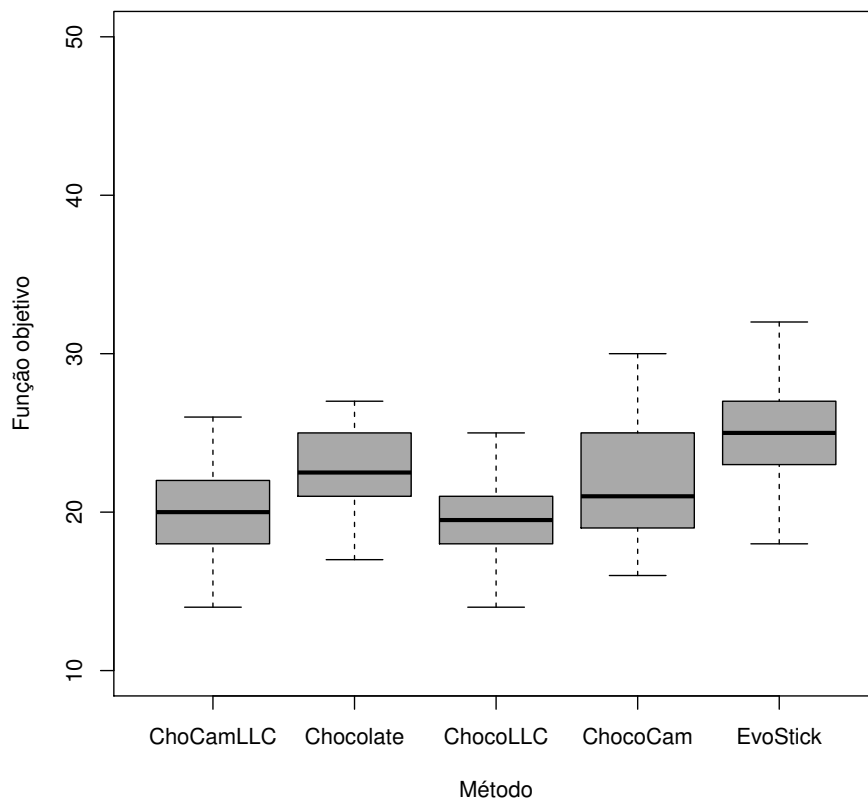
1:  $I_{n_{ins}} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $n_o \leftarrow 0$ 
4:  $n_{robos} \leftarrow 20$ 
5:  $T_m \leftarrow 1200$ 
6: for  $t = 1, t \leq T_m, t++$  do
7:   for  $i = 1, i \leq n_{robos}, i++$  do
8:     if Item coletado por robôi then
9:        $n_o \leftarrow n_o + 1$ 
10:  $C(\theta) \leftarrow n_o$ 
11: return  $C(\theta)$ 

```

Na linha 1 são definidas as condições iniciais dos robos (distribuição uniforme no cenário) e a definição do cenário, representado como um cenário dodecagonal com duas regiões circulares pretas de diâmetro $0,3\text{ m}$ localizadas em $(-0,75\text{ m}; 0,0\text{ m})$ e $(0,75\text{ m}; 0,0\text{ m})$, uma região branca em $y \leq -0,6\text{ m}$ e rodeada das paredes da arena. A arena possui uma fonte luminosa próxima da área onde os objetos são depositados localizada em $(0,0\text{ m}; -1,5\text{ m})$ e a $0,4\text{ m}$ de altura. Na linha 4 e linha 5 se define a quantidade de robôs e tempo de experimento (120 s), respectivamente. Na linha 6-9 o experimento é executado e a cada

ciclo de amostragem e para todos os robôs se realiza a contagem de objetos coletados (linha 9). Os robôs devem coletar a maior quantidade de objetos possíveis das regiões circulares pretas para logo depositar esses objetos na região branca. Como os robôs não possuem efetadores para coletar os objetos, se considera que um robô coleta o objeto quando ingressa nas regiões pretas. Por sua vez, é considerado que o robô deposita o objeto na região branca da arena, quando ingressa nela e se antes ingressou na região preta. Na linha 10 é finalizado o experimento com a devolução do valor da função objetivo na linha 11.

Figura 4.4: Diagrama de caixas apresentando o desempenho dos métodos CHOCOLATE e EVOSTICK na missão **Procura de objetos**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o desempenho quantitativo da missão na Figura 4.4, pode ser concluído que os controladores do método EVOSTICK possuem um melhor desempenho que os do método CHOCOLATE de acordo com o teste de Wilcoxon 95%. A diferença de medianas de desempenho é de 5 objetos coletados e o intervalo de confiança sinaliza que no pior dos casos

o desempenho do método EVOSTICK será melhor que o método CHOCOLATE em 4,5 objetos coletados. A mediana de desempenho do método CHOCOLATE é de aproximadamente 20 objetos e do EVOSTICK é de 25.

Observando o comportamento emergente produzido por ambos os métodos, no método CHOCOLATE se observa que os robôs começam explorar a arena e quando um objeto é encontrado em alguma das regiões pretas os robôs começam seguir a fonte luminosa até chegar na região branca. Assim começa o processo de exploração novamente. O processo de busca e depósito dos objetos não tem uma organização aparente. Porém, o método EVOSTICK possui um comportamento coletivo em que os robôs realizam uma rotação pela arena que permite aos robôs ingressarem geralmente nas duas áreas pretas e chegar na região branca, reiniciando ciclicamente o processo. Um comportamento emergente de seguidor é produzido, assim os robôs vão juntos de forma cíclica pelas regiões, uns seguindo os outros, mas sem nenhum líder que comanda o enxame.

A respeito das especializações de CHOCOLATE, nenhuma delas obteve um desempenho superior ao método CHOCOLATE de acordo com o teste de Wilcoxon 95%. Nesta tarefa específica, o controlador de baixo nível de CHOCOLATE parece ser mais apropriado para o movimento dos robôs. O controlador de baixo nível de CHOCOLLIC e CHOCAMLLIC coloca limitações na velocidade linear do robô no intuito de satisfazer a velocidade angular do robô. Por isto, uma quantidade menor de objetos são coletados em cada experimento. A respeito a CHOCOCAM, não existe significância estatística para concluir que possui desempenhos diferentes a CHOCOLATE. A respeito do comportamento coletivo qualitativo das especializações, não se observou diferença, no que diz respeito ao comportamento coletivo de CHOCOLATE. Os controladores individuais tem semelhanças quanto aos comportamentos primitivos escolhidos pelo projeto automático.

4.4.2 Comportamento de agregação

Este comportamento coletivo possui o objetivo de reunir o enxame de robôs numa região determinada. Este comportamento é avaliado em três tarefas: **Acesso restrito ao refúgio**, **agregação** e **agregação com informação no ambiente**.

Acesso restrito ao refúgio

A missão tem o objetivo de agrupar a maior quantidade de robôs no refúgio que possui chão branco. A função objetivo que deve ser maximizada é

$$C(\theta) = \sum_{t=0}^{T_m} N(t), \quad (4.14)$$

em que $N(t)$ é a quantidade de robôs no refúgio no instante de tempo t . A função objetivo é avaliada a cada $0,1 s$ (*time step*) e o experimento tem a duração de $T_m = 120 s$. A função objetivo permite calcular a presença dos robôs no refúgio branco. Assim, quanto maior o valor, maior a presença de robôs na região. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 2 Cálculo da função objetivo para a missão **acesso restrito ao refúgio**

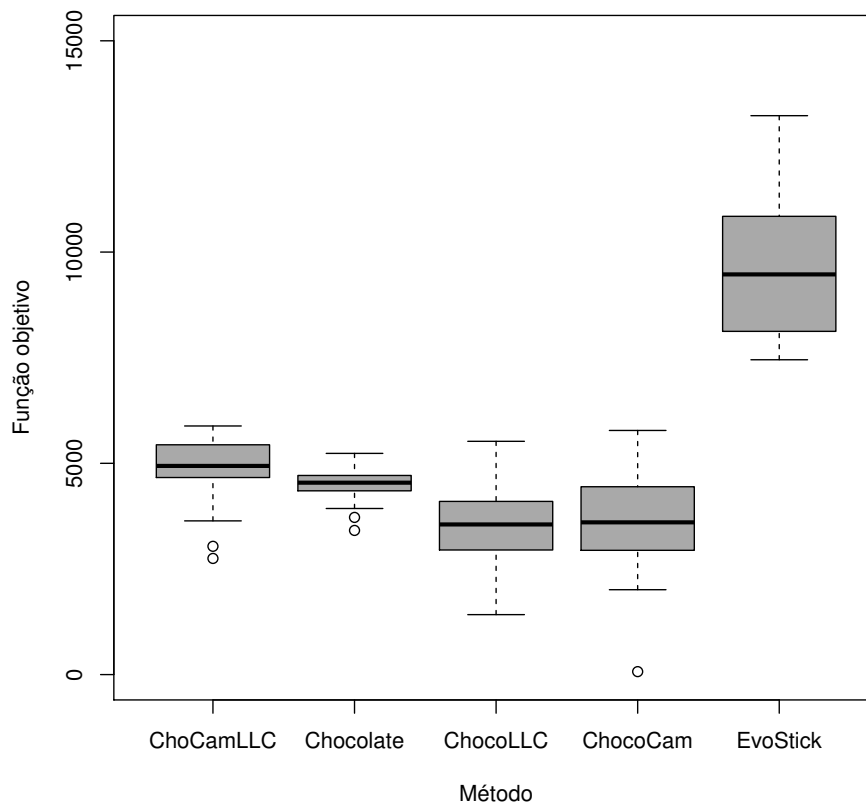
```

1:  $I_{n_{ins}} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $N \leftarrow 0$ 
4:  $n_{robos} \leftarrow 20$ 
5:  $T_m \leftarrow 1200$ 
6: for  $t = 1, t \leq T_m, t++$  do
7:   for  $i = 1, i \leq n_{robos}, i++$  do
8:     if robôi no refúgio then
9:        $N \leftarrow N + 1$ 
10:  $C(\theta) \leftarrow N$ 
11: return  $C(\theta)$ 

```

Na linha 1 são definidas as condições iniciais dos robos (distribuição uniforme no cenário) e a definição do cenário, representado como um cenário dodecagonal com um refúgio branco. O refúgio tem as dimensões de $0,15 m \times 0,6 m$ e possui um único acesso livre com os outros três acessos bloqueados por paredes, localizado na origem de coordenadas. Por sua vez, o cenário também possui uma fonte luminosa que serve como orientação dos robôs, da mesma forma que duas regiões circulares pretas de $0,6 m$ de diâmetro localizadas em $(-0,8 m; 0,1 m)$ e $(0,8 m; 0,1 m)$. A lâmpada esta localizada em $(0,0 m; -1,5 m)$ e a $0,4 m$ de altura. Na linha 1-5 é inicializado o experimento. Na linha 6-9 é executado o experimento e a cada ciclo de amostragem e para todos os robôs se realiza a contagem de robôs no refúgio (linha 9). Na linha 10 é finalizado o experimento com a devolução do valor da função objetivo na linha 11.

Figura 4.5: Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão **Acesso restrito ao refúgio**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o desempenho quantitativo da missão na Figura 4.5, pode ser concluído que os controladores do método CHOCOLATE possuem um menor desempenho que os do EVOSTICK de acordo com o teste de Wilcoxon 95%. Em cada instante de tempo, o comportamento coletivo dos controladores produzidos pelo método CHOCOLATE agrega uma quantidade menor de robôs que o comportamento coletivo do EVOSTICK. A diferença entre medianas de desempenho é de 4707 presenças de robôs e o intervalo de confiança sinaliza que no pior dos casos o desempenho dos controladores do método EVOSTICK será melhor que os controladores do método CHOCOLATE em 4151 presenças de robôs. Isto equivale a uma diferença aproximada de 3,5 presenças de robôs na região preta a cada *time step*. Comparando as medianas de desempenho dos controladores, a mediana do método EVOSTICK dobra a do método CHOCOLATE, sendo de aproximadamente 8,2 e 4,1 presenças

de robôs em cada ciclo de controle, respectivamente.

Observando o comportamento coletivo dos 10 controladores individuais de CHOCOLATE, geralmente os robôs possuem dificuldades de ingressar no refúgio devido ao que o comportamento emergente do enxame tenta evitar colisões entre os robôs e com o ambiente. Porém, as interações entre robôs produzidas pelo método EVOSTICK são menos repulsivas, permitindo menor distâncias entre robôs e maior quantidade de colisões entre eles. Por isto, a evolução dos controladores no método EVOSTICK produz controladores em que uma maior quantidade de robôs conseguem se agregar no refúgio. Observa-se também, que a fonte luminosa como informação é utilizado em ambos métodos, os robôs são guiados pela luz até encontrar a parede da arena, quando uma certa quantidade de robôs se agrupa nesta parte da arena, o comportamento de *antifototaxis* é ativado em e leva eventualmente os robôs ao refúgio.

A respeito às novas especializações de CHOCOLATE, de acordo com o teste de Wilcoxon 95 % existe significância estatística entre CHOCOLATE e CHOCOLLC, e entre CHOCOLATE e CHOCOCAM. Em ambas comparações, CHOCOLATE possui um melhor desempenho. Porém, da comparação entre CHOCOLATE e CHOCAMLLC observa-se que CHOCAMLLC possui um melhor desempenho que CHOCOLATE. Estes resultados mostram, nesta tarefa específica, que a modificação do controlador de baixo nível e da câmera isoladamente não parecem ter efeitos positivos no desempenho. Porém, a combinação de ambas modificações parece favorecer o desempenho dos controladores. Com relação ao comportamento coletivo, qualitativamente não se observam diferenças significativas entre as especializações na comparação com CHOCOLATE.

Agregação

A missão tem o objetivo de agrupar os robôs numa das duas regiões pretas do cenário. A função objetivo que deve ser maximizada é

$$C(\theta) = \frac{\max(N_a, N_b)}{N}, \quad (4.15)$$

em que N_a e N_b são a quantidade de robôs nas regiões pretas "a" e "b" no final do experimento no tempo T_m . N é a quantidade total de robôs utilizados no experimento. A função objetivo que deve ser maximizada é a quantidade de robôs em uma das regiões pretas. Assim, quanto

maior o valor, maior a concentração de robôs numa das duas áreas. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 3 Cálculo da função objetivo para a missão **agregação**

```

1:  $I_{n_{ins}} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $N_a \leftarrow 0, N_b \leftarrow 0, N \leftarrow 20$ 
4:  $n_{robos} \leftarrow 20$ 
5:  $T_m \leftarrow 1200$ 
6: for  $t = 1, t \leq T_m, t++$  do
7:   if  $t = T_m$  then
8:     for  $i = 1, i \leq n_{robos}, i++$  do
9:       if robôi na area "a" then
10:         $N_a \leftarrow N_a + 1$ 
11:      if robôi na area "b" then
12:         $N_b \leftarrow N_b + 1$ 
13:  $C(\theta) \leftarrow \max(N_a, N_b) / N$ 
14: return  $C(\theta)$ 

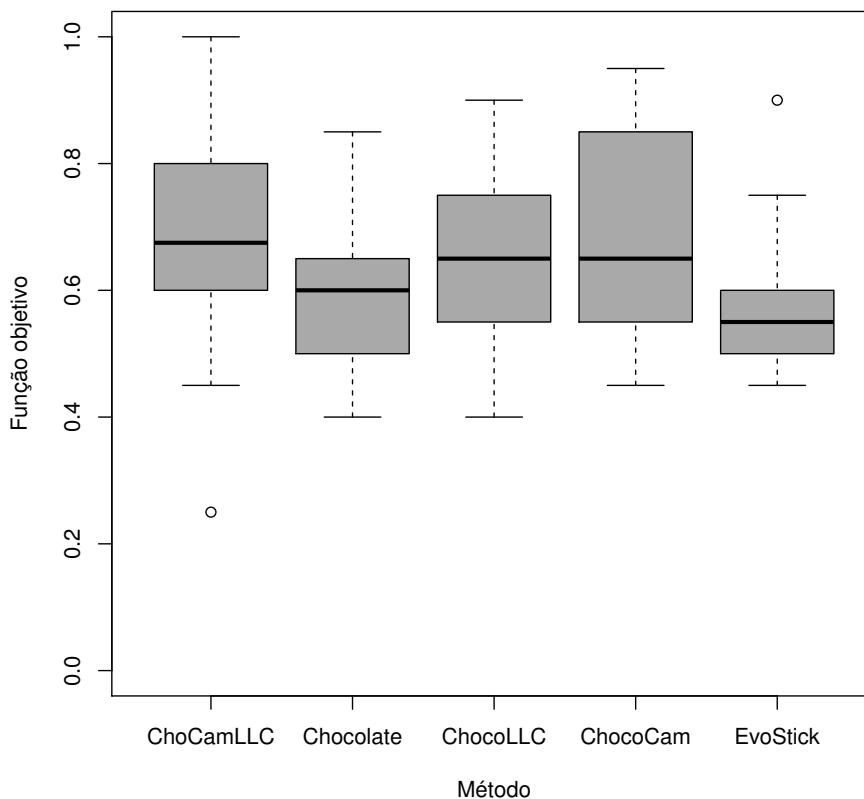
```

O chão da arena é cinza com duas regiões pretas de diâmetro $0,6\text{ m}$ centradas em $(0,0\text{ m}; 0,6\text{ m})$ e $(0,0\text{ m}; -0,6\text{ m})$. Os robôs são inicializados uniformemente no cenário. Na linha 6-12 o experimento é executado e no final do experimento (linha 7) e para todos os robôs (linha 8) se realiza a contagem de robôs nas areas. Na linha 13 é finalizado o experimento com a devolução do valor da função objetivo na linha 14.

Observando o desempenho quantitativo da missão na Figura 4.6, pode ser concluído que os controladores do método CHOCOLATE possuem um melhor desempenho que os controladores do método EVOSTICK de acordo com o teste de Wilcoxon 95%. Nos experimentos se observa que o método CHOCOLATE agrupa mais robôs que o método EVOSTICK em alguma das duas regiões pretas. A diferença entre medianas de desempenho é de 0,15 (3 robôs) e o intervalo de confiança sinaliza que no pior dos casos o desempenho da especialização CHOCOLATE será melhor que o método EVOSTICK em 0,07 (1 robô).

Observando o comportamento coletivo dos 10 controladores individuais do método CHOCOLATE, observa-se que 14 robôs são agrupados na região preta de maior agrupamento. Porém, no método EVOSTICK observa-se um número menor, 11 robôs, distribuindo os robôs nas regiões aproximadamente em partes iguais quando todos os robôs conseguem ser agrupados em alguma região. Geralmente os robôs se atraem entre eles formando pequenos

Figura 4.6: Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão **Agregação**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

conjuntos de robôs locais, eventualmente quando eles encontram o chão preto, eles param ficando estáticos nessa região. Assim, estes robôs servem para atrair mais robôs. No caso do método EVOSTICK observa-se que os robôs quando encontram o chão preto, eles começam rotacionar no seu próprio eixo, mantendo a posição e atraindo mais robôs na região.

Observando o desempenho das novas especializações de CHOCOLATE, existe significância estatística entre as especializações e o próprio CHOCOLATE de acordo com o teste de Wilcoxon 95%. Os resultados sinalizam que CHOCO LLC, CHOCOCAM e CHO CAM LLC possuem melhor desempenho que CHOCOLATE. Dos resultados dos comportamentos individuais dos robôs, não são observadas diferenças significativas entre as máquinas de estado obtidas.

Agregação com informação no ambiente

A missão tem o objetivo de agrupar a maior quantidade de robôs na região preta da arena. A função objetivo que deve ser maximizada é

$$C(\theta) = \sum_{t=0}^{T_m} N(t), \quad (4.16)$$

em que $N(t)$ é a quantidade de robôs na região circular preta no instante de tempo t . A função objetivo é avaliada a cada $0,1 s$ (*time step*) e o experimento tem a duração de $T_m = 120 s$. A função objetivo calcula a presença dos robôs na região preta. Assim, quanto maior o valor, maior a presença de robôs na região. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 4 Cálculo da função objetivo para a missão **agregação com informação no ambiente**

```

1:  $I_{n_{ins}} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $N \leftarrow 0$ 
4:  $n_{robos} \leftarrow 20$ 
5:  $T_m \leftarrow 1200$ 
6: for  $t = 1, t \leq T_m, t++$  do
7:   for  $i = 1, i \leq n_{robos}, i++$  do
8:     if robô $i$  na area preta then
9:        $N \leftarrow N + 1$ 
10:  $C(\theta) \leftarrow N$ 
11: return  $C(\theta)$ 

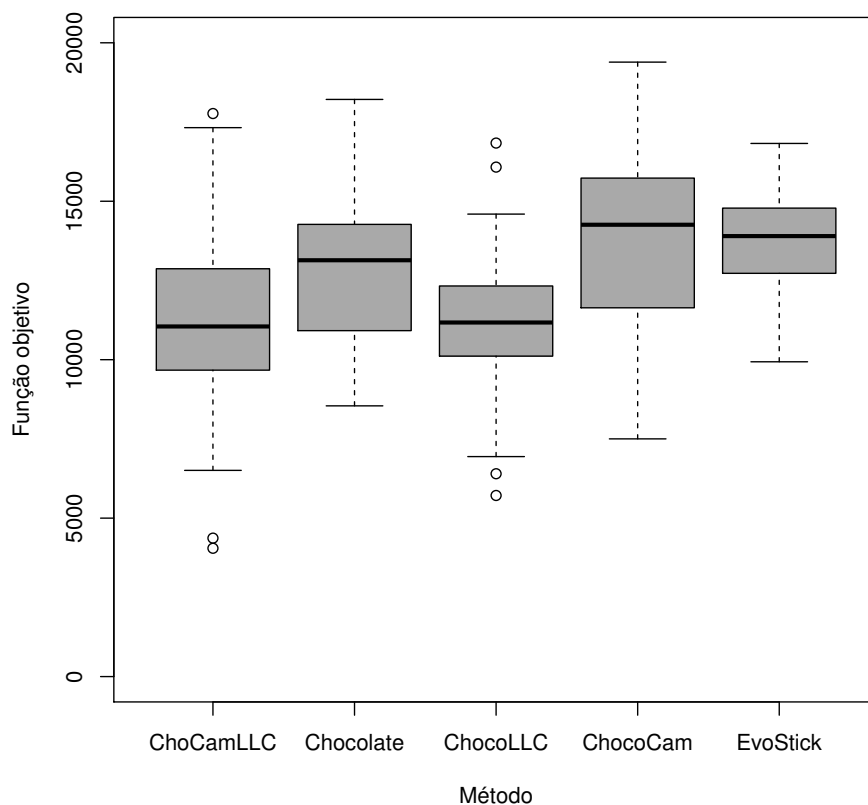
```

Na linha 1 é definida as condições iniciais dos robos (distribuição uniforme no cenário) e a definição do cenário, representado como um cenário dodecagonal cinza que possui duas regiões circulares de diâmetro $0,6 m$. A região circular preta está localizada em $(0,0 m; -0,6 m)$ e a branca em $(0,0 m; 0,6 m)$. Uma fonte luminosa foi colocada próxima da região preta e pode servir como orientação dos robôs, localizada em $(0,0 m; -1,5 m)$ e a $0,4 m$ de altura. Na linha 6-9 o experimento é executado e a cada ciclo de amostragem e para todos os robôs se realiza a contagem de robos na regioao preta (linha 9). Na linha 10 é finalizado o experimento com a devolução do valor da função objetivo na linha 11.

Observando o desempenho quantitativo da missão (Figura 4.7) pode ser concluído que os controladores do método CHOCOLATE possuem um menor desempenho que os do método

EVOSTICK de acordo com o teste de Wilcoxon 95%. Em cada instante de tempo os controladores do método CHOCOLATE agregam uma quantidade menor de robôs que os controladores do método EVOSTICK. A diferença entre medianas de desempenho é de 2778 presenças de robôs e o intervalo de confiança sinaliza que no pior dos casos o desempenho do método EVOSTICK será melhor que o desempenho do método CHOCOLATE em 1632 presenças de robôs. Isto equivale a uma diferença aproximada de 1,4 presenças de robôs na região preta a cada *time step*.

Figura 4.7: Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão **Agregação com informação no ambiente**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o comportamento coletivo dos 10 controladores individuais do método CHOCOLATE e do método EVOSTICK, os robôs aproveitam o uso da fonte luminosa para ser atraídos mais rapidamente na região de interesse. No caso dos controladores do método CHOCOLATE, eles ficam estáticos uma vez que encontram a região preta. Outros robôs

eventualmente perto destes robôs, serão atraídos na região de interesse. No caso do método EVOSTICK observa-se que os robôs quando encontram o chão preto, eles começam rotacionar no seu próprio eixo, mantendo a posição e atraindo mais robôs na região.

A respeito às novas especializações de CHOCOLATE, de acordo com o teste de Wilcoxon 95 %, o método CHOCOLATE possui um melhor desempenho que as especializações CHOCOLLIC e CHOCAMLLIC. A respeito da comparação entre as especializações CHOCOCAM e CHOCOLATE, não é possível concluir que exista significância estatística de desempenhos entre os métodos. De igual forma que na tarefa **Procura de objetos**, o controlador de baixo nível do método CHOCOLATE parece aproveitar de melhor forma a fonte luminosa. Respeito aos comportamentos individuais dos robôs as máquinas de estado possuem os mesmos padrões de transições e comportamentos utilizados.

4.4.3 Comportamento de formação de padrões

Este comportamento coletivo possui o objetivo de produzir formas de organização regulares em que os robôs devem respeitar certas distâncias entre eles para atingir o padrão global. Este comportamento é avaliado em duas tarefas: **cobertura e monitoramento de região** e **Cobertura da maior rede do enxame**.

Cobertura e monitoramento de região

A missão tem o objetivo de realizar cobertura de áreas e monitoramento de perímetro simultaneamente. A função objetivo que deve ser minimizada é

$$C(\theta) = \frac{E[d_A]}{c_A} + \frac{E[d_P]}{c_P}, \quad (4.17)$$

em que $E[d_A]$ é a distância esperada entre qualquer ponto aleatório da região quadrada branca e o robô mais próximo nessa região quadrada. Por sua vez, $E[d_P]$ é a distância esperada entre qualquer ponto aleatório no perímetro da região circular preta e o robô mais próximo nessa região circular. As constantes $c_A = 0,08 m$ e $c_P = 0,06 m$ são fatores de escala para normalizar a medida e foram calculadas na condição de 9 robôs igualmente distribuídos na região branca e outros 9 no perímetro da região preta como apresentado no artigo [19]. A função objetivo é avaliada no final do experimento no tempo T_m . O pseudocódigo para o

cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 5 Cálculo da função objetivo para a missão **cobertura e monitoramento de região**

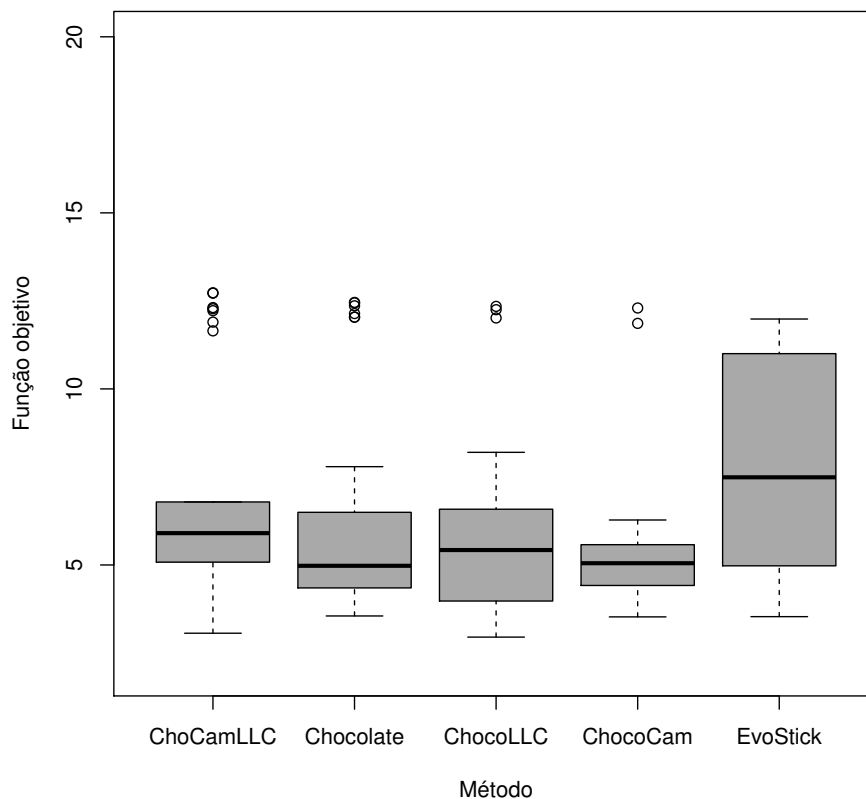
```

1:  $I_{nins} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $E[d_A] \leftarrow 0$ ,  $E[d_P] \leftarrow 0$ 
4:  $c_A \leftarrow 0,08$ ,  $c_P \leftarrow 0,06$ 
5:  $num_p \leftarrow 1000$ 
6:  $n_{robos} \leftarrow 20$ 
7:  $T_m \leftarrow 1200$ 
8: for  $t = 1$ ,  $t \leq T_m$ ,  $t++$  do
9:   if  $t = T_m$  then
10:     for  $l = 1$ ,  $l \leq num_p$ ,  $l++$  do
11:        $d_{min} \leftarrow 0,85 m$ 
12:       for  $i = 1$ ,  $i \leq n_{robos}$ ,  $i++$  do
13:         if  $rob\hat{o}_i$  na area branca then
14:            $d_A \leftarrow dist(p_l, rob\hat{o}_i)$ 
15:           if  $d_A < d_{min}$  then
16:              $d_{min} \leftarrow d_A$ 
17:            $E[d_A] \leftarrow E[d_A] + d_{min}$ 
18:        $E[d_A] \leftarrow E[d_A]/num_p$ 
19:     for  $l = 1$ ,  $l \leq num_p$ ,  $l++$  do
20:        $d_{min} \leftarrow 0,6 m$ 
21:       for  $i = 1$ ,  $i \leq n_{robos}$ ,  $i++$  do
22:         if  $rob\hat{o}_i$  na area preta then
23:            $d_P \leftarrow dist(p_l, rob\hat{o}_i)$ 
24:           if  $d_P < d_{min}$  then
25:              $d_{min} \leftarrow d_P$ 
26:            $E[d_P] \leftarrow E[d_P] + d_{min}$ 
27:        $E[d_P] \leftarrow E[d_P]/num_p$ 
28:  $C(\theta) \leftarrow E[d_A]/c_A + E[d_P]/c_P$ 
29: return  $C(\theta)$ 

```

Os robôs devem cobrir a região branca e permanecer no perímetro da região preta. A arena contém uma região quadrada de cor branca de $0,6 m$ de comprimento centrada em $(0,0 m; 0,6 m)$. Por sua vez, existe uma região circular preta de $0,6 m$ de diâmetro e centrada em $(0,0 m; -0,6 m)$. No final do experimento é calculado $E[d_A]$ ($E[d_P]$) de acordo às linhas 10-18 (19-27). Na linha 10 são escolhidos aleatoriamente 1000 pontos na região branca e na linha 19 são escolhidos outros 1000 no perímetro da região preta. Na linha 28 é finalizado o experimento com a devolução do valor da função objetivo (linha 29).

Figura 4.8: Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão **cobertura e monitoramento de região**, quanto menor a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o desempenho quantitativo da missão na Figura 4.8, pode ser concluído que, de acordo com o teste de Wilcoxon 95%, não existe evidência estatística suficiente para rejeitar a hipótese nula do teste que assegura que as medianas de desempenho são iguais entre os métodos CHOCOLATE e EVOSTICK. Da figura, se observam em ambos os métodos desempenhos similares com medianas muito próximas. Os controladores fornecidos pelo método CHOCOLATE oferecem soluções que são mais próximas do desempenho da sua mediana, sem muita dispersão. Porém, o controlador do método EVOSTICK possui mais variabilidade no desempenho. Isto é devido a que o comportamento coletivo do sistema não sempre é similar entre as instâncias de controladores. Os controladores do método CHOCOLATE apresentam maior homogeneidade na solução.

Observando o comportamento emergente produzido pelo método CHOCOLATE, pode

ser observado que os robôs exploram a arena e uma vez que cada robô ingressa na região preta, atravessam ela com um comportamento alternado entre parar e se movimentar. Quando o robô está saindo da região e encontra novamente a região cinza, finalmente para até o final do experimento, permanecendo na borda da região preta. Porém, na região branca não é observada uma coesão entre os robôs para manter distâncias e cobrir a região apropriadamente. Os robôs ingressam na região branca e param, observando-se agregados de robôs sem coesão de distâncias e por sua vez a quantidade de robôs é menor que a observada a região preta. Também, observa-se em alguns controladores que os robôs depois de certo tempo abandonam a região branca. Por sua vez, no método EVOSTICK a única diferença radica em que os robôs exploram o ambiente e quando ingressam nas regiões começam rotacionar no seu próprio eixo, e também se formam alguns agregados fora das regiões de cobertura.

A respeito das especializações de CHOCOLATE, de acordo com o teste de Wilcoxon 95 %, não existe significância estatística que indique que os controladores produzidos com o método CHOCOLATE possuam desempenhos diferentes dos controladores produzidos com as especializações CHOCOCAM, CHOCOLLIC e CHOCAMLLIC. Os comportamentos coletivos observados não possuem diferenças entre os métodos.

Como conclusão da missão pode ser dito que a coesão entre os robôs não é observada em nenhum dos métodos na região branca. Os métodos projetaram controladores que priorizam mais a cobertura do perímetro da região preta que a área da região branca. Isto é devido a que os métodos não conseguem desenvolver controladores que produzam interações entre os robôs com um certo grau de coesão para realizar coberturas de áreas. O comportamento coletivo para obter coesão realizando cobertura de área exige maior esforço, pois é um comportamento que exige sincronização dos robôs nas suas interações para manter distâncias e também o controlador deve ter em conta as interações dos robôs com o ambiente, para determinar a cor do chão e determinar se essa é a região correta para permanecer.

Cobertura da maior rede do exame

A missão tem o objetivo de maximizar a cobertura da arena com a maior rede conectada possível de robôs do exame. Assim, quanto maior o valor, maior a área que vai ser coberta.

A função objetivo que deve ser maximizada é

$$C(\theta) = A_{N_B}, \quad (4.18)$$

em que N_B é definida como a rede com maior quantidade de robôs. Esta rede N_B cobre a área A_{N_B} . O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 6 Cálculo da função objetivo para a missão **cobertura da maior rede do enxame**

```

1:  $I_{n_{ins}} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $d_r = 0,35 m$ ,  $d_c = 0,25 m$ 
4:  $A_{N_B} \leftarrow 0$ 
5:  $n_{robos} \leftarrow 20$ 
6:  $T_m \leftarrow 1200$ 
7: for  $t = 1$ ,  $t \leq T_m$ ,  $t++$  do
8:   if  $t = T_m$  then
9:      $N \leftarrow$  DefinirRedes(robôs, $d_r$ )
10:     $N_B \leftarrow$  DefinirRedeComMaiorQtRobôs( $N$ )
11:     $A_{N_B} \leftarrow$  area $_{N_B}(N_B, d_c)$ 
12:  $C(\theta) \leftarrow A_{N_B}$ 
13: return  $C(\theta)$ 

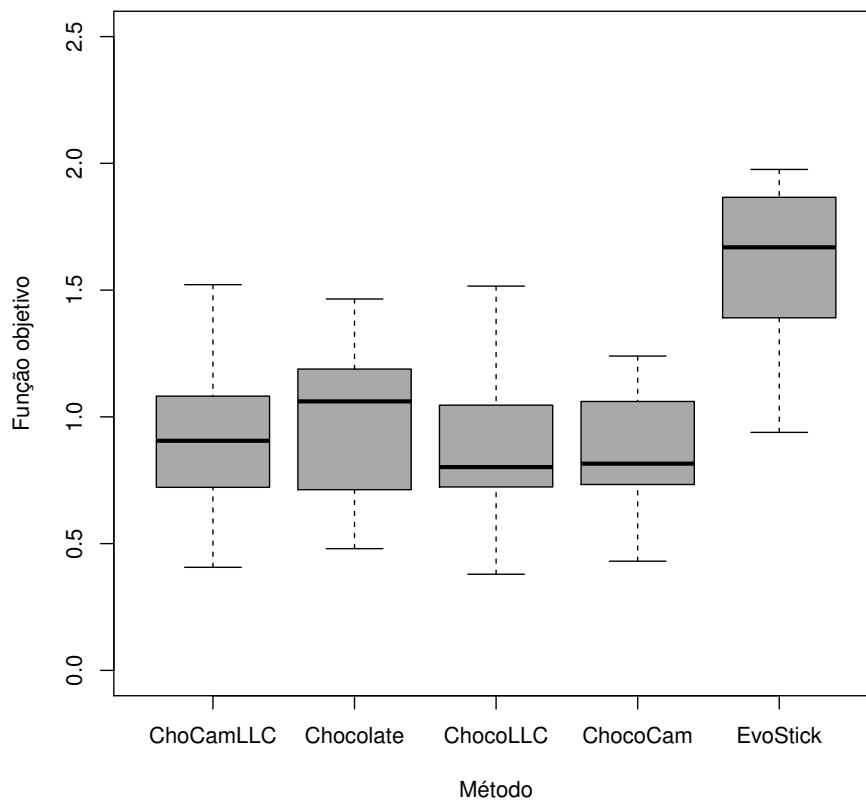
```

Na linha 1 são definidas as condições iniciais dos robos (distribuição uniforme no cenário) e a definição do cenário, representado como um cenário dodecagonal com chão totalmente cinza e sem obstáculos. Na linha 7-11 o experimento é executado. No final do experimento se definem as redes N de robos existentes (linha 9) e se encontra a rede de maior quantidade de robos N_B a partir de N (linha 10). Um robô é considerado em N_B se possui conectividade de pelo menos de 1 vizinho. Os vizinhos de um robô são aqueles que possuem uma distância de $d_r \leq 0,35 m$. Na linha 11 se calcula a area coberta pela rede N_B : O alcance de cobertura de cada robô é de $d_c \leq 0,25 m$, permitindo também redes que podem cobrir sem solapamento entre as regiões de cobertura de cada robô. Na linha 12 é finalizado o experimento com a devolução do valor da função objetivo na linha 13.

Observando o desempenho quantitativo da missão na Figura 4.9, pode ser concluído que os controladores produzidos pelo método EVOSTICK possuem um melhor desempenho que os controladores produzidos pelo método CHOCOLATE de acordo com o teste de Wilcoxon 95 %. No final de cada experimento a maior rede de robôs pelo método EVOSTICK consegue cobrir

uma área maior comparada com a produzida pelo método CHOCOLATE. A diferença entre medianas de desempenho é de $0,67 m^2$ e o intervalo de confiança sinaliza que no pior dos casos o desempenho do método EVOSTICK será melhor que o método CHOCOLATE em $0,55 m^2$. A mediana de cobertura da arena é de aproximadamente $1,0 m^2$ e $1,7 m^2$ para os métodos CHOCOLATE e EVOSTICK, respectivamente. Isto representa que o método EVOSTICK cobre aproximadamente 14% a mais da área do cenário de trabalho quando comparado com o método CHOCOLATE.

Figura 4.9: Diagrama de caixas apresentando o desempenho dos controladores para diferentes métodos de projeto na missão **Cobertura da maior rede do enxame**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o comportamento qualitativo produzido por ambos os métodos, CHOCOLATE é incapaz de gerar redes de robôs conectadas. Os robôs não mantêm distâncias específicas com os outros robôs do enxame. Os enlaces entre robôs não são mantidos. Assim, a coesão entre os robôs não é observada. Eles parecem ser atraídos entre si e as interações de atração

e repulsão são as predominantes, mas a coesão não é mantida, gerando que conjuntos de menor tamanho sejam formados, não tendo bom desempenho na resolução da tarefa. Por sua vez, no método EVOSTICK a rede de robôs é geralmente maior que a produzida pelo método CHOCOLATE, mas não é o suficientemente rígida como para ser mantida ao longo do experimento. Como conclusão pode se dizer que o ajuste das restrições de distâncias realizado pelo método EVOSTICK foi mais adequado que a realizada pelo método CHOCOLATE no que diz respeito a cobrir a área da arena. Porém, o ajuste das interações entre robôs não conseguiu desenvolver uma coesão mantida no tempo.

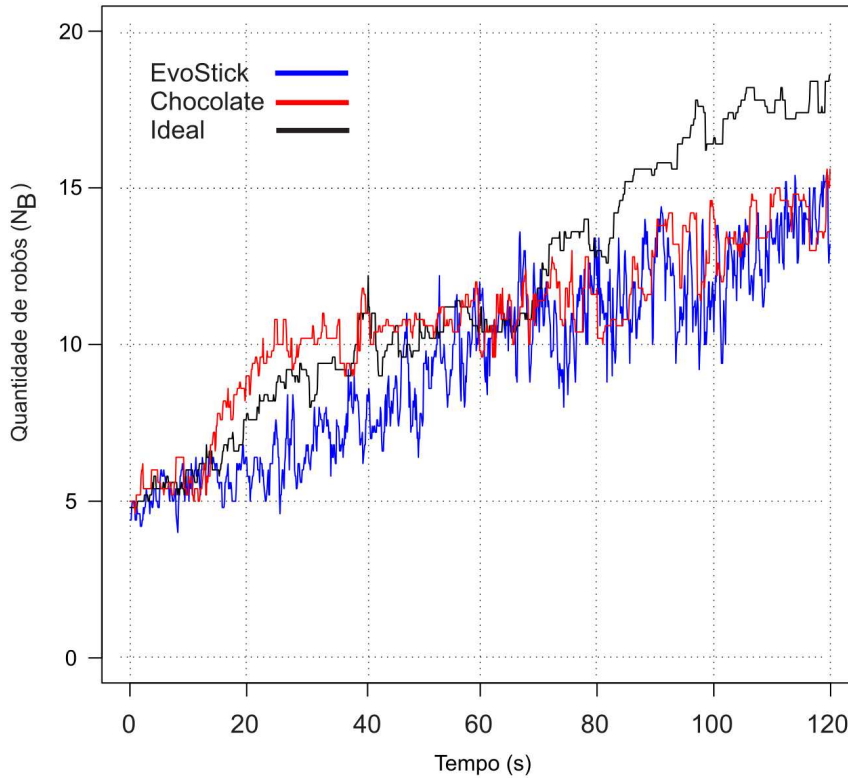
Respeito as especializações de CHOCOLATE, de acordo com o teste de Wilcoxon 95 %, não existe significância estatística que indique que o método CHOCOLATE possui desempenhos diferentes que as especializações CHOCOCAM, CHOCOLLIC e CHOCAMLLIC. Os resultados são concordantes à tarefa anterior (**Cobertura e monitoramento de região**). Os comportamentos coletivos observados entre os métodos não possuem diferenças relevantes. Nenhuma das três especializações conseguiu atingir nem manter a conectividade e coesão entre os robôs do enxame.

Na Figura 4.10 observa-se a quantidade de robôs da maior rede de robôs N_B em função do tempo, para a missão **Cobertura da maior rede do enxame**. Os controladores comparados são controladores do método EVOSTICK (curva na cor azul) e do método AUTOMODE-CHOCOLATE (curva na cor vermelha). Por sua vez, é realizada a comparação com uma curva ideal (curva na cor preta) obtida de projetos manuais, nos quais foi incluída a restrição de distância entre robôs para eles permanecer a 20 cm de distância com os vizinhos a $d \leq 0,35$ cm. Todas as curvas de N_B foram obtidas da ponderação de 10 controladores.

Como observado da figura, o tamanho de N_B aumenta em função do tempo para as três curvas. O valor mínimo é obtido no instante inicial, próximo de 5 robôs. O valor máximo de robôs é de 15 robôs conectados do conjunto para os métodos automáticos e próximo de 18 para o projeto manual de controladores. Pode ser observado também que o método AUTOMODE-CHOCOLATE possui maior número de robôs conectados até aproximadamente a metade do experimento, logo ambos métodos possuem conjuntos similares.

Na Figura 4.11 são observadas a media e desvio padrão da distância entre robôs de N_B para os métodos de projeto automático AUTOMODE-CHOCOLATE, EVOSTICK e para o método manual. As curvas das medias se observam na parte superior da figura e as curvas de desvio

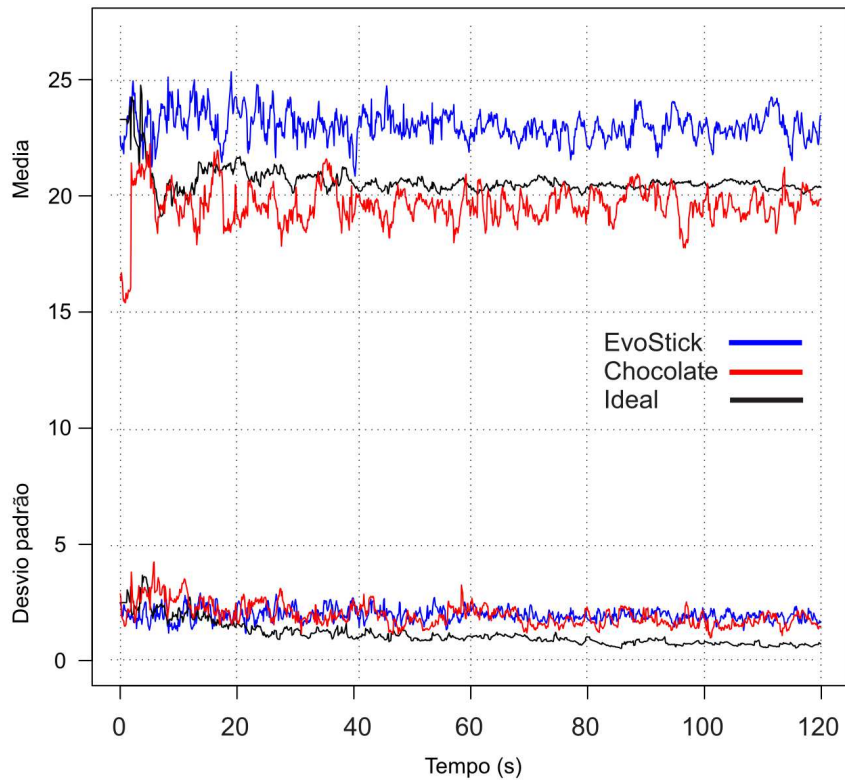
Figura 4.10: Tamanho do maior conjunto de robôs (N_B) em função do tempo para a missão Cobertura da maior rede do enxame.



Fonte: Elaborada pelo autor.

padrão se observam na parte inferior da mesma figura. Para cada método, estas curvas estão ponderadas sobre os 10 controladores projetados. A media é calculada como $\bar{x} = \sum_i \sum_j d_{ij}/n$, d_{ij} é a distância do robô $i \in N_B$ com os robôs vizinhos, $j \neq i, j \in N_B$. Um robô é considerado vizinho se $d_{ij} \leq 0,35 m$. Por sua vez, n é a quantidade de valores d_{ij} utilizados no cálculo da media. O desvio padrao σ é calculado como $\sigma = \sqrt{\sum_i \sum_j (d_{ij} - \bar{x})^2 / (n - 1)}$. Da figura se observa que a distância media entre robôs de N_B produzida pelos controladores obtidos por intermédio do método EVOSTICK é maior às obtidos por intermédio do método AUTOMODE-CHOCOLATE. Este fato é traduzido em maior área coberta utilizando o método EVOSTICK (Figura 4.9). Respeito ao desvio padrão, ambos os métodos projetam controladores com desvio padrão similar. Este desvio padrão sinaliza que as distâncias entre robôs vizinhos possuem variabilidade e possuem valores altos de dispersão respeito a media de distâncias. Estes resultados indicam que existe dificuldades para os robôs do enxame manter coesão, no

Figura 4.11: Media e desvio padrão em função do tempo para a missão **Cobertura da maior rede do enxame**.



Fonte: Elaborada pelo autor.

que diz respeito a manter distâncias específicas. As distâncias entre pares de robôs possuem variabilidade e a coesão não é mantida no tempo. Por sua vez, da solução ideal pode ser observado que os robôs mantêm a coesão na distância projetada, com uma media próxima de 20 *cm* que é verificada da análise do desvio padrão. O desvio padrão é menor quando é comparado com as curvas de projeto automático, indicando que os robôs mantêm a coesão desejada.

4.5 Considerações finais

Neste capítulo realizou-se a comparação de dois métodos de projeto automático de controladores para enxames de robôs em diferentes tarefas previamente definidas na bibliografia. O objetivo foi observar o desempenho dos métodos AUTOMODE-CHOCOLATE e EVOSTICK,

principalmente em tarefas com restrições de distâncias, para observar se os métodos possuem limitações para resolver tarefas coletivas em que a coesão é relevante para a solução da tarefa.

Foi considerada uma abordagem baseada em robótica evolutiva, chamada de EVOSTICK [18], em que o processo de ajuste dos parâmetros do controlador foi realizado por um algoritmo artificial evolutivo que ajustou os parâmetros de uma rede neuronal. A rede neuronal foi definida sem nós ocultos em que todos os neurônios da entrada são conectados com os de saída.

Foi apresentado o método de projeto de controladores para robôs de enxames denominado de CHOCOLATE [18], que realizou a síntese de controlador para cada robô. AUTOMODE-CHOCOLATE é um método de projeto automático que sintetiza o controlador de um robô individual por intermédio de um algoritmo de otimização que maximiza uma função objetivo que descreve a tarefa que os robôs do enxame devem executar. Este processo de otimização permite ajustar os módulos e seus parâmetros para a obtenção do controlador.

Foram apresentadas três novas especializações do método CHOCOLATE, chamadas de CHOCOCAM, CHOCOLLC e CHOCAMLLC. CHOCOCAM substitui o sensor para medir distâncias entre robôs. CHOCOLLC substitui o controlador de baixo nível do robô e CHOCAMLLC inclui as duas modificações conjuntamente. Estas novas especializações tiveram o propósito de observar se estas modificações permitiam produzir comportamentos de coesão entre os robôs do enxame.

Foi apresentado o robô utilizado com o seu modelo. Após, os modelos dos atuadores e sensores para esse robô foram apresentados. Logo, foi definida a metodologia dos experimentos, tendo em conta o tipo de restrição de distância colocado na tarefa coletiva. Foram apresentadas seis missões divididas em três grandes grupos: *Comportamento de agrupação de objetos*, *Comportamento de agregação* e *Comportamento de formação de padrões*. Entre esses grupos se definiram missões para agrupar objetos encontrados em regiões específicas da arena (*Procura de objetos*), para agrupar a maior quantidade de robôs em um refúgio (*Acesso restrito ao refúgio*), agrupar os robôs numa das duas regiões do cenário (*Agregação*, agrupar a maior quantidade de robôs numa região que possui sinalização luminosa (*Agregação com informação no ambiente*), cobertura de área com a maior rede conectada possível de robôs (*Cobertura da maior rede do enxame*) e, finalmente, cobertura de áreas e monitoramento de

perímetro simultaneamente (*Cobertura e monitoramento de região*).

Foram comparados os métodos de projeto automático nestas missões para verificar se as abordagens de projeto automático de controladores para enxames de robôs possuem limitações na resolução de alguma dessas tarefas. Particularmente observam-se limitações nas duas missões de *Comportamento de formação de padrões*, nas quais uma solução correta destas missões exige que robôs devem manter certas distâncias entre eles.

Considerando os resultados da seção anterior, pode ser concluído que a abordagem de robótica evolutiva avaliada foi capaz de realizar uma variedade ampla de tarefas robóticas no ambiente simulado e de obter soluções relevantes na maioria das missões, com desempenhos melhores aos obtidos pelo método CHOCOLATE. Isto é devido a que foi definida uma arquitetura sem limitações nas relações de entrada e saída que permitiu um ajuste granular das interações entre os robôs e dos robôs com o ambiente. Da avaliação qualitativa das missões, foi observado que as missões em que a solução requer de coesão, mantendo distâncias específicas entre os robôs, não conseguem ser resolvidas corretamente.

A abordagem modular de projeto, CHOCOLATE, apresentou bons desempenhos e soluções para as três tarefas de agregação e a tarefa de agrupar os objetos. Os desempenhos no geral foram abaixo dos do método EVOSTICK. Isto pode ser explicado, pois a arquitetura possui mais restrições nas relações de entrada e saída comparadas com as restrições impostas no método EVOSTICK. O EVOSTICK possui mais liberdade para ajustar essas interações. No entanto, e da mesma forma que o método EVOSTICK, o método CHOCOLATE não conseguiu resolver corretamente as duas tarefas que visam coesão com fortes restrições de distâncias.

Pode ser observado que os métodos de projeto CHOCOLATE e EVOSTICK foram avaliados sobre as mesmas condições experimentais seguindo o mesmo protocolo que em [Francesca et al.\[19\]](#): Foi utilizado o mesmo robô entre os métodos, sensores, atuadores e foram empregados os mesmos modelos para o robô; Foi empregada a mesma definição do problema utilizando a mesma função objetivo e a mesma definição do ambiente para cada tarefa; Foi utilizado o mesmo simulador para representar os robôs e cenário; Foi definida a mesma quantidade de experimentos em ambos métodos para projetar os controladores. A única diferença entre os métodos EVOSTICK e CHOCOLATE é o algoritmo de otimização utilizado e à arquitetura empregada de controlador. Em relação às especializações de CHOCOLATE, as diferenças se encontram no hardware escolhido para medir as distâncias entre robôs e o controlador de

baixo nível utilizado para comandar os atuadores do robô.

No tocante ao algoritmo de otimização e sua configuração, o espaço de busca inicial de controladores é o suficiente abrangente em ambos métodos para conter controladores promissores, no que diz respeito às tarefas propostas nesta seção. A respeito à definição da topologia de máquina de estados definida no método AUTOMODE-CHOCOLATE, pode ser dita que a quantidade de estados máxima definida na arquitetura foi suficiente para representar os potenciais controladores que abordem as tarefas deste capítulo. Por sua vez, a quantidade de transições por estado é razoável para conectar um estado com seus sucessores caso seja necessário. Isto nos fornece uma indicação de que soluções para comportamentos coletivos que requerem coesão não vão surgir do ajuste de parâmetros e módulos unicamente, incluso se é aumentada a quantidade de estados da máquina de estados. No que diz respeito a arquitetura e topologia escolhida no método EVOSTICK, pode ser dito que o tipo de arquitetura é o adequado para abordar o problema e maiores estudos devem ser realizados para analisar a incidência das definições das entradas e saídas da rede neuronal.

A respeito à definição da função objetivo, pode ser dito que várias funções objetivo poderiam abordar o mesmo problema para encontrar o comportamento coletivo que resolve a tarefa. Embora este raciocínio seja válido, pode-se observar que foram definidas duas instâncias de uma mesma classe de tarefa. As missões *Cobertura da maior rede do enxame* e *Cobertura e monitoramento de região* são duas instâncias da classe *Comportamento de formação de padrões*. Estas funções objetivo são funções que colocam diferentes tipos de pressão na busca da solução [31]. A missão *Cobertura e monitoramento de região* procura cobertura e monitoramento simultâneo, produzindo uma pressão distribuída em dois objetivos. O raciocínio anterior pode ser a justificativa da falta de coesão do time. Porém, foi considerada uma outra função objetivo na missão *Cobertura da maior rede do enxame*. Nesta função objetivo foi colocada uma pressão maior nos algoritmos de otimização no que diz respeito à coesão do time. Logo, pode ser dito que empiricamente tentou-se atacar com ambos os métodos esta classe de tarefas. Como conclusão pode ser dito que em ambas as missões o comportamento coletivo encontrado não é o apropriado para resolver a tarefa.

De igual forma, as novas especializações de CHOCOLATE, não conseguiram resolver estas duas últimas tarefas corretamente. Das especializações, pode ser concluído que não foi possível desenvolver comportamentos coletivos de coesão entre os robôs do enxame. A

resolução de tarefas coletivas que requerem coesão no enxame não foram resolvidas com a modificação do controlador de baixo nível, nem com a inclusão de um sensor de distâncias mais preciso e menos propenso à interferência. Por sua vez, a combinação destas características não favoreceu o projeto tendente a resolver este tipo de tarefas coletivas.

Finalmente, pelo protocolo utilizado nos experimentos e pelos resultados obtidos pode ser concluído que soluções que precisam coesão em enxames, especificamente aquelas que os robôs precisam manter distâncias específicas, foram comprometidas pela definição dos controladores. As restrições das relações de entrada e saída da arquitetura nos métodos propostos, não permitem que no enxame emergja o comportamento coletivo de coesão. O anterior nos leva a concluir que as restrições de distâncias incluídas nos métodos de projeto não permitiram obter o comportamento coletivo capaz de obter a coesão nas interações entre os robôs.

Capítulo 5

Comportamentos com restrições de distâncias

Neste capítulo será apresentada a viabilidade e necessidade de utilizar comportamentos com restrições de distâncias para permitir coesão nos robôs de um enxame quando realizam movimento coletivo. O intuito é mostrar a relevância do comportamento de coesão por intermédio de experimentos no ambiente de simulação.

Os experimentos foram realizados em simulação com cenários sem obstáculos para observar o funcionamento de cada comportamento separadamente. Após, foram realizados experimentos em cenários com obstáculos para observar o funcionamento conjunto de todos os comportamentos projetados. É realizada uma análise quantitativa e qualitativa do comportamento coletivo do enxame observado.

Primeiramente, na Seção 5.1 será realizada a modelagem do robô, com a modelagem do seus sensores e atuadores. Logo, na Seção 5.2 definida e justificada a escolha da arquitetura utilizada. Logo, será explicitada a lei de controle que vai ser utilizada nessa arquitetura e serão apresentados os pontos fortes e as limitações da abordagem. Depois disso, na Seção 5.3 será apresentado controlador do robô que codifica uma série de comportamentos com restrições de distâncias. Na Seção 5.4 serão apresentados os resultados experimentais. Finalmente, na Seção 5.5 serão apresentadas as considerações finais do capítulo.

5.1 Modelo de robô e sensores

Modelos são necessários no intuito de controlar um sistema, esses modelos devem ser gerais, para poder ser implementados em outros robôs; eles devem ser simples e relevantes o suficiente para especificar o fenômeno que se deseja modelar sem adicionar complexidade desnecessária.

Para o desenvolvimento dos experimentos deste capítulo foram escolhidos os robôs Foot-bot. Os robôs Foot-bot possuem tração diferencial composto por um sistema de lagartas e rodas. Para maiores detalhes, o robô Foot-bot foi anteriormente apresentado na Seção 3.5.2. Os robôs Foot-Bot oferecem um conjunto de sensores e atuadores apropriados para realizar comportamentos com restrições de distâncias.

5.1.1 Modelo cinemático do robô

O robô Foot-bot compartilha várias similitudes com o robô e-puck. Ambos os robôs possuem tração diferencial e o modelo cinemático que representa melhor esses robôs é o modelo apresentado na Seção 4.1.1. O robô Foot-bot possui um comprimento do eixo das rodas de $L = 0,14 m$.

5.1.2 Modelo de sensores e atuadores do robô

Sensores de proximidade

No caso do Foot-bot, o robô possui 24 sensores de proximidade e o valor medido é $v_{i_{prox}} \in \mathbb{R} : [0,1]$, $i \in \mathbb{N} : [1,24]$. A equação que modela as medições do sensor em função da distâncias é a Equação (5.1).

$$v_{i_{prox}} = \begin{cases} \frac{a}{d_{ij}+b} + \sigma_{prox} & d_{ij} \geq 0,00989 m \\ 1 & cc \end{cases} \quad (5.1)$$

em que $a = 0,0100527 m$ e $b = 0,000163144 m$. Estas constantes foram obtidas de dados empíricos do sensor de proximidade. A distância d_{ij} neste caso é entre o sensor $i \in \mathbb{N} : [1,24]$ e o obstáculo mais próximo do sensor na linha de visão. Esta distância é calculada em

metros como na Equação (4.4). De acordo com dados experimentais, o ruído é considerado um ruído aleatório definido no intervalo $\sigma_{prox} \in \mathbb{R} : [-0,05; 0,05]$.

Sensores de luminosidade

O sensor de luminosidade do Foot-bot foi modelado como na Equação (4.8) da Seção 4.1.2. A distância d_{ij} entre o sensor $i \in \mathbb{N} : [1,24]$ e o obstáculo mais próximo do sensor na linha de visão, é calculada em metros como na Equação (4.4). No simulador, a intensidade da fonte luminosa é medida em metros e foi definida num valor de $I = 25 m$. Esta distância é 10 vezes a distância máxima medível pelo sensor físico. Isto foi feito para simular uma bússola que permite medir a direção do movimento do enxame. O experimento tem o objetivo de fornecer o espaço suficiente para os robôs se movimentarem na direção da fonte luminosa e se estabelecerem num padrão de coesão, antes e depois de obstáculos. Assim, na condição inicial os robôs medem a fonte luminosa. Segundo dados experimentais, o ruído é considerado um ruído aleatório definido no intervalo $\sigma_{luz} \in \mathbb{R} : [-0,05; 0,05]$.

Câmera omnidirecional

A câmera omnidirecional do Foot-bot permite, da mesma forma que a do e-puck, a captura de conjunto de pixels com certas características da imagem. A calibração permite a captura de LEDs, led_i , dos robôs vizinhos na distância de captura do sensor. A distância $v_{i_{cam}} \in \mathbb{R} : [0,10; 1,35] m$ do robô a cada LED é calculada utilizando a Equação (4.10), a mesma que para modelar a câmera do e-puck da Seção 4.1.2. De acordo com dados experimentais, é considerado um ruído aleatório gaussiano com média nula e desvio padrão $\sigma_{prox} \in \mathbb{R} : [-0,04; 0,04] m$.

Tração do robô

O modelo do atuador inclui ruído aleatório em cada velocidade das rodas, como realizado na Equação (4.11) da Seção 4.1.2. Foi considerado um ruído aleatório definido no intervalo $\sigma_{atu} \in \mathbb{R} : [-0,05; 0,05]$.

5.2 Arquitetura do comportamento *formar*

Na revisão bibliográfica feita no Capítulo 2, foi corroborado que existem vários métodos de projeto de controlador para os robôs do enxame. Os métodos estudados produzem controladores para abordar tarefas coletivas com restrições de distâncias, em que a distribuição no cenário de trabalho deve ser feita com coesão. Estes comportamentos permitem a coesão de robôs sem colisões, manter a conectividade e possibilitam o movimento coletivo do enxame.

A abordagem de campos potenciais artificiais (Seção 3.6.1), aplicada em enxame de robôs, permite controlar os robôs por intermédio de forças virtuais, produto da interação com robôs vizinhos. Esta abordagem de controle leva os robôs do enxame em certas posições de equilíbrio que são considerados pontos de menor energia do sistema [38]. Da definição das interações a nível local entre os robôs vizinhos é obtido um padrão desejado, no qual os robôs mantêm certas posições e coesão entre eles, atingindo a auto-organização do sistema.

Na natureza, entidades tais como moléculas e átomos exercem forças sobre outras entidades e respondem a forças de outras entidades. Geralmente as únicas forças relevantes são aquelas de entidades próximas. Da mesma forma, na abordagem de campos potenciais para enxame de robôs o cálculo é realizado via **interações locais** [38]. Cada robô é modelado como uma partícula que exerce forças virtuais sobre outros robôs e são exercidas forças sobre cada robô no alcance do seu sensor de detecção de proximidade com outros robôs.

A escolha da abordagem de campos potenciais artificiais se deve principalmente pelos pontos citados a seguir:

- Localidade das interações.
- Independência da plataforma robótica.
- Abordagem minimalista apropriada para a robótica de enxame.
- Tolerância a falhas.
- Geração de padrões com coesão.
- Invariabilidade do controlador pela escalabilidade do sistema.
- Implementável em robôs físicos.

- Análise de propriedades do enxame.
- Facilidade de implementação em arquiteturas modulares.

Como foi mencionado em [Spears e Gordon\[38\]](#), campos potenciais artificiais permitem o controle descentralizado de vários tipos de entidades móveis, desde nano robôs, veículos aéreos até satélites. Esta capacidade se deve a que o método é desacoplado da parte de controle de baixo nível (Seção 3.6.1). Esta característica de **independência da plataforma robótica**, permite projetar a lei de controle inclusive antes de saber o robô que vai ser utilizado. Uma vez que a plataforma foi selecionada, outras restrições de distâncias devem ser tidas em conta, como o controle para evitar colisões e o controlador de baixo nível para controlar os movimentos da plataforma [29]. Por isto, a abordagem de campos potenciais é fácil de ser implementada e transferida para outros tipos de plataformas.

Tendo em conta as limitações dos robôs que compõem enxames de robôs, no que diz respeito a seus sensores e atuadores, a abordagem de campos potenciais artificiais é uma abordagem propícia para o controle desses robôs, pois unicamente são necessárias informações locais do entorno do robô que podem ser obtidas com sensores de baixo custo. Pelo anterior, a abordagem de campos potenciais artificiais é uma **abordagem minimalista** pois são necessários um conjunto mínimo de sensores e atuadores para ser implementada [29]. Frente as limitações dos sensores e atuadores em enxames de robôs, a abordagem se apresenta **tolerante a falhas**, pois em caso de falha de algum robô a abordagem permite a reparação da estrutura do enxame: o enxame se readapta e as forças calculadas por cada robô farão com que a região ocupada pelo robô avariado seja novamente ocupado por outro robô do enxame. Esta característica de tolerância a falhas faz a abordagem ser robusta no que diz respeito a avaria de robôs e inclusive a ruído dos sensores e atuadores, como observado em [Spears e Gordon\[38\]](#).

Utilizando esta abordagem de campos potenciais artificiais, cada robô pode manter distâncias arbitrárias com os robôs da sua vizinhança, formando diversos tipos de **padrões com coesão**. Por exemplo, a manipulação das restrições de distâncias entre robôs vizinhos pode produzir padrões, tais como hexágonos, quadrados, linhas, dentre outros. Os padrões gerados com esta abordagem são padrões que não são perfeitos no que diz respeito a que são unicamente locais. Normalmente, não são colocadas restrições globais no padrão [38]. Caso

sejam colocadas restrições globais com relação a estrutura do padrão, podem ser esperados padrões perfeitos tal como os apresentados em [Spears e Gordon](#); [Kazadi](#)[38, 48].

Como a abordagem é escalável no que diz respeito à quantidade de robôs no enxame, as técnicas de controle são projetadas independentemente da quantidade de robôs no enxame [38]. Por isto, o **controlador é invariável** com relação a quantidade de robôs no enxame. Por sua vez, a abordagem de campos potenciais não requerem de um poder de computo excessivo para calcular a lei de controle em cada instante de tempo, pois não são utilizados modelos para realizar previsões de nenhum tipo sobre os robôs vizinhos nem o ambiente no qual o robô esta imerso. Pelo anterior, o controlador pode ser facilmente **implementável em robôs reais**. O controlador se caracteriza por ser um controlador reativo que responde em base a estímulos observados no instante de tempo atual.

A utilização de campos potenciais permite maior previsibilidade do sistema, como trabalhos realizados em [Spears et al.](#); [Spears et al.](#)[125, 126]. Por sua vez, podem ser realizadas análises quantitativas da energia e da transição entre estruturas de formação, como realizado em [Spears e Gordon](#); [Spears et al.](#); [Kazadi](#)[29, 38, 48]. Finalmente, podem ser derivadas conclusões quanto à estabilidade do padrão e convergência [16], como em [Shucker, Murphey e Bennett](#)[47]. A abordagem de campos potenciais artificiais pode ser facilmente incorporada em **arquiteturas modulares**, pois, permite a composição vetorial cooperativa e competitiva de comportamentos (Seção 3.3.1). Finalmente, a abordagem é amplamente utilizada na bibliografia, para tarefas coletivas como cobertura [6, 43, 108], monitoramento e vigilância [29, 43, 47], movimento coletivo [29, 46, 49, 89], entre outros.

5.2.1 Escolha da lei de controle

A função de potencial de Lennard-Jones (Seção 3.6.1) é uma das funções dentro da abordagem da teoria de campos potenciais artificiais. Foi inicialmente proposta para modelar interações entre moléculas e átomos. A função de Lennard-Jones mostrou empiricamente que possui convergência para produzir comportamentos relevantes na robótica de enxame, em especial comportamentos com restrições de distâncias. Trabalhos em que foi estudada esta lei podem ser encontrados em [Spears, Thayer e Zarzhitsky](#); [Ferrante et al.](#); [Pinciroli et al.](#); [Pinciroli et al.](#); [Hettiarachchi e Spears](#)[4, 65, 68, 89, 127], entre outros.

A escolha da função de Lennard-Jones nesta tese é justificada pelos seguintes motivos:

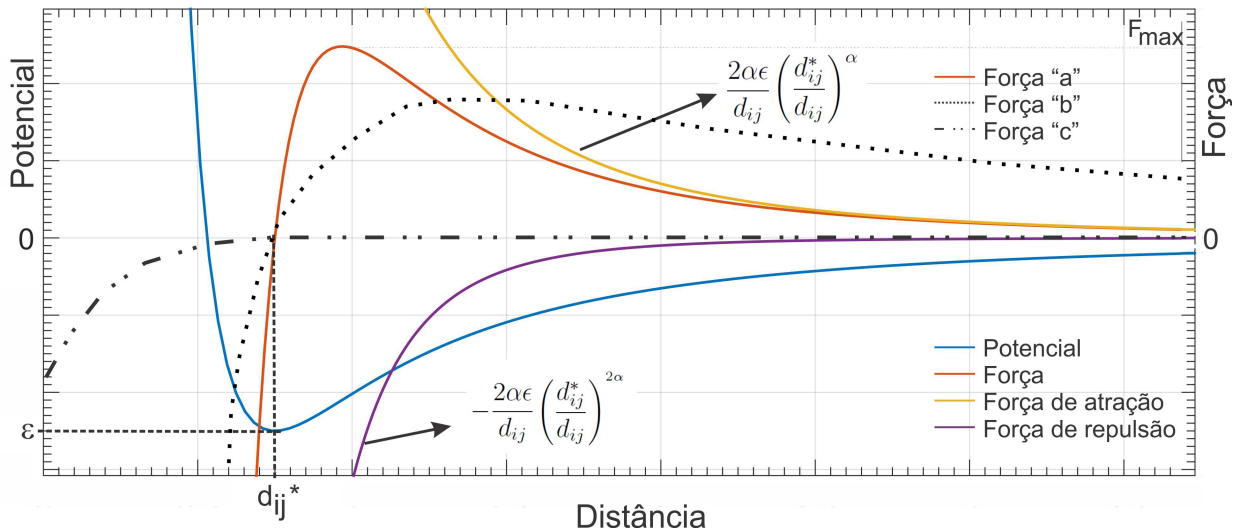
- Fortemente parametrizável.
- Parâmetros intuitivos de serem ajustados.
- Função contínua na região de operação.
- Permite modelar formações com diferentes tipos de rigidez.
- Permite previsibilidade da realização de tarefas coletivas.

A função de Lennard-Jones pode ser expressa de várias formas, como por exemplo em [Spears, Thayer e Zarzhitsky\[127\]](#) é apresentada a sua forma generalizada, que é fortemente parametrizável, em que pode ser ajustadas a intensidade da força de atração e de repulsão independentemente, a inclinação da curva na zona de operação, dentre outros parâmetros. A zona de operação se define como a região que inclui a distância desejada da coesão que os robôs devem manter. Uma forma simplificada da Função de Lennard-Jones é apresentada em [Ferrante et al.\[89\]](#) (Equação (3.12)). Nesta expressão, é possível ajustar de forma granular a distância desejada d_{ij}^* entre pares de robôs, a intensidade das forças de atração e repulsão de forma conjunta utilizando o parâmetro ϵ . Finalmente, a lei permite o ajuste do parâmetro α , que define a inclinação da curva na zona de operação.

O ajuste desses parâmetros oferecem diferentes tipos de coesão entre os robôs [5]. Primeiramente, a manipulação da distância desejada entre cada par de robôs permite definir vários tipos de padrões diferentes. Por exemplo, podem ser produzidos padrões hexagonais quando as distâncias d_{ij}^* entre pares de robôs são todas iguais, padrões em forma de quadrado quando se tem dois tipos de distâncias entre robôs [29]. Por sua vez, regiões assimétricas da vizinhança de cada robô permitem obter padrões estreitos até cadeias de robôs, exemplos da manipulação da região podem ser observados em [Spears; Maxim; Mendiburu, Morais e Lima\[49, 128, 129\]](#).

Na Figura 5.1 se observa três ajustes de parâmetros diferentes para a função de Lennard-Jones. A diferença entre as três curvas são os valores de α e ϵ . A curva chamada de força "a", esta ajustada para que as relações de atração e repulsão realizem uma coesão rígida. Uma coesão rígida permite manter as interações entre robôs firmes. Assim, na região de

Figura 5.1: Diferentes ajustes de parâmetros para a função de Lennard-Jones. Curva "a" é uma coesão rígida, curva "b" é uma coesão líquida e curva "c" os robôs não possuem coesão.



Fonte: Elaborada pelo autor.

operação, pequenas variações na distância produziram grandes mudanças na força. A curva chamada de força "b" é obtida reduzindo o valor de α e aumentando o de ϵ , mantendo a mesma intensidade da força máxima de atração que a força "a", mas as interações entre os robôs são mais flexíveis (coesão líquida). Esta configuração é mais apropriada para que o enxame consiga se adaptar mais facilmente às características do ambiente. Por exemplo, coesão mais flexível é apropriada para o enxame evitar obstáculos do ambiente no movimento coletivo [5,44]. Finalmente a curva chamada de força "c" é obtida da força "a" mas reduzindo ϵ . Assim, a força "c" possui unicamente a componente de repulsão. Nesta situação não existe coesão entre os robôs, formando um enxame gasoso, no qual os robôs se repelem suavemente para evitar colisões, mas não existe um componente para atrair os robôs vizinhos. Como pode ser visto do exemplo anterior, as relações de interação utilizando a função de Lennard-Jones podem ser modificadas facilmente e de forma intuitiva.

Por sua vez a função de Lennard-Jones é uma função contínua na região de operação. Que seja uma função contínua tem uma série de vantagens no que diz respeito à implementação da lei de controle nos robôs físicos. Por exemplo, que a função seja contínua na região de operação evita instabilidade e movimentos bruscos no robô [130]. Este resultado afeta

diretamente a qualidade da coesão entre os robôs. Funções que são descontínuas, vão ter variações grandes e imediatas na lei de controle entre o componente atrativo e repulsivo que degrada o desempenho da coesão entre os robôs [130].

A função de Lennard-Jones permite previsibilidade da realização de tarefas coletivas, como apresentado em Spears et al.; Spears et al.[125, 126]. Os autores mostraram que a utilização da função de Lennard-Jones permite avaliar de forma mais precisa o risco de realização de certas tarefas coletivas de enxame, baseados em pequenas amostras do enxame. Outros trabalhos apresentados na bibliografia comparam a função de Lennard-Jones com outras leis de controle disponíveis na bibliografia. Foi mostrado empiricamente que a função de Lennard-Jones tem desempenho significativamente melhor do que a força gravitacional para a realização de movimento coletivo em ambientes desestruturados [64]. A função de Lennard-Jones apresentou baixos níveis de colisão com obstáculos e os robôs atingiram o objetivo em tempos razoáveis e com alta conectividade. Finalmente, a função de Lennard-Jones, por possuir interações nas quais podem ser de pouca rigidez, permitem atenuar alguns dos problemas dos campos potenciais artificiais, como são o movimento coletivo em corredores [131]. Outras funções rígidas, por possuir pouca flexibilidade nas interações podem enfrentar problemas e os robôs não conseguirem passar no corredor.

Pelas justificativas anteriores, a função de Lennard-Jones é selecionada como lei de controle para atingir a coesão entre robôs. Na Equação (3.12) foi modelada a interação entre dois robôs. Para interações entre um robô com mais de um vizinho, a força aplicada no robô se define como a soma das forças aplicadas pelos robôs vizinhos de forma individual. Assim, o módulo *formar* é definido a seguir, na Equação (5.2).

$$r_{cam} = \begin{cases} \frac{1}{n} \sum_{i=1}^n \frac{4\epsilon}{v_{i_{cam}}} \left[\left(\frac{d^*}{v_{i_{cam}}} \right)^4 - \left(\frac{d^*}{v_{i_{cam}}} \right)^2 \right] e^{-j\phi_{i_{cam}}} & n \geq 1 \\ e^{j0} & n = 0 \end{cases} . \quad (5.2)$$

A lei de controle utiliza a câmera omnidirecional do robô para realizar as medidas de distâncias com os robôs vizinhos. O vetor r_{cam} é o vetor resultante de cada robô devido as interações produzidas com os robôs vizinhos. A distância medida entre os $i \in \mathbb{N} : [0, n]$ LEDs dos robôs vizinhos é $v_{i_{cam}}$, e o ângulo medido é $\phi_{i_{cam}}$. O parâmetro α foi ajustado em $\alpha = 2$, que apresenta melhor desempenho nos robôs físicos [89]. O parâmetro d^* representa a

distância desejada entre robôs, ajustado pelo projeto manual, ou ajustado automaticamente pelo projeto automático. Para simplificar o tipo de coesão produzida, esta distância d^* é considerada a mesma para todos os pares de robôs ($d_{ij}^* = d^*$). Na bibliografia, é mostrado que este ajuste de distâncias produz uma configuração de robôs efetivamente distribuída em padrão de grade, com forma de hexágono [38]. Por sua vez, na expressão da Equação (5.2), é possível ajustar a intensidade das forças de atração e repulsão de forma conjunta utilizando o parâmetro ϵ .

5.2.2 Limitações da abordagem

A escolha da função de campos potenciais artificiais possui limitações conhecidas e apontadas na bibliografia, como por exemplo, o problema de mínimos locais [132]. O problema de mínimos locais acontece tanto para robôs individuais como para enxames de robôs resolvendo tarefas coletivas. No movimento coletivo, cada robô está sujeito a forças de atração e repulsão. Considerando o caso de que as forças de interação entre os robôs são nulas (os robôs possuem coesão ideal no deslocamento), os robôs possuem unicamente forças de atração e repulsão produzidas pelos objetivos e pelos obstáculos do ambiente, respectivamente. O problema surge quando a superposição dessas forças em cada robô se anulam numa região do ambiente de trabalho diferente da região na qual se encontra o objetivo a ser atingido. Esta situação produz que os robôs fiquem presos numa região do ambiente de trabalho diferente da região desejada.

Várias abordagens foram propostas para tratar o problema de mínimos locais. Uma delas é a modificação dos parâmetros livres da lei de controle de forma *online* que permite que cada robô manipule a força resultante aplicada nele. Utilizando a abordagem anteriormente mencionada, a modificação dos parâmetros da lei de controle de forma *online* é realizada em Mabrouk e McInnes; Mabrouk e McInnes[133, 134]. Nesta abordagem, quando os robôs estão presos num mínimo local, existe transformação da região para um máximo local que permite os robôs escapar da região. Normalmente, neste tipo de abordagem, os robôs aprendem por intermédio do progresso que eles estão fazendo em direção a um objetivo. Assim, caso não existir progresso em direção ao objetivo é sinônimo que os robôs estão num mínimo local e o mecanismo se ativa para permitir modificar as forças aplicadas no robô.

Finalmente, os robôs conseguem escapar do mínimo local produto de modificação da força resultante e, por sua vez, da interação com outros robôs que já conseguiram escapar do mínimo local [133].

Outras técnicas para escapar do mínimo local são aplicadas em robôs individuais [135], que medem o progresso que é feito em direção ao objetivo e utilizam seguimento de paredes para escapar do mínimo local. Uma vez que o robô está fora do mínimo local, a abordagem muda de estado e o robô deixa de seguir a parede e continua o movimento para o objetivo, evitando obstáculos. Existem também outras limitações apontadas em Koren e Borenstein[132], um dos problemas é causado por instabilidades no movimento devido a oscilações do movimento do robô quando o robô encontra obstáculos. Estes problemas são encontrados normalmente em robôs se movimentando em alta velocidade em ambientes com obstáculos. Outra limitação é encontrada quando existem corredores ou lugares estreitos para os robôs passarem. Neste último caso, para evitar estas limitações podem ser utilizadas as técnicas de modificação de potencial anteriormente citadas e a utilização de formações flexíveis que permitem comprimir o enxame na passagem de corredores [131]. Uma dessas leis de controle flexível é a função de Lennard-Jones, que se mostrou uma lei de controle apropriada para superar as limitações dos campos potenciais artificiais em ambientes não estruturados [64].

Embora as limitações de campos potenciais artificiais não sejam abordadas nesta tese, o uso de projeto automático *offline* de controladores para robôs de enxame consegue lidar parcialmente com algumas das limitações dos campos potenciais artificiais em ambientes estruturados (os quais não possuem variações no tempo). Por exemplo, suponha que o enxame de robôs deve resolver uma tarefa utilizando movimento coletivo nesses ambientes com obstáculos fixos. O controlador deve ser projetado para cada robô produzir o comportamento coletivo desejado que resolva a tarefa coletiva. No projeto do controlador, o controlador projetado que resolve a tarefa é escolhido sobre outros de menor desempenho. Alguns desses controladores de menor desempenho, podem ter ajustes de parâmetros específicos que podem levar aos robôs do enxame num mínimo local e o enxame não resolver a tarefa. Estes controladores que levam os robôs a mínimos locais foram, eventualmente, descartados no processo de avaliação dos controladores, por terem desempenhos inferiores a outros que conseguiram superar esse mínimo local. Assim, o ajuste de parâmetros *offline* permite regular as interações entre robôs e dos robôs com o ambiente e tendem a resolver essas limitações

observadas na abordagem de campos potenciais artificiais.

Em outras situações, nas quais o cenário de trabalho não é estruturado, podem ser produzidos mínimos locais nos quais o controlador escolhido (de melhor desempenho) no projeto *offline* não consegue se adaptar a essas mudanças no ambiente. A escolha da função de Lennard-Jones permite amenizar as limitações anteriores. A função possui utilidade, pois permite modelar formações com coesão flexível. Esta característica, somada a que a função pode ser ajustada com mecanismos *offline* e reajustada de forma *online*, permite resolver as limitações da abordagem de campos potenciais artificiais. Assim, a escolha da função de Lennard-Jones nesta tese permite a futura expansão do trabalho para superar as limitações da abordagem de campos potenciais artificiais.

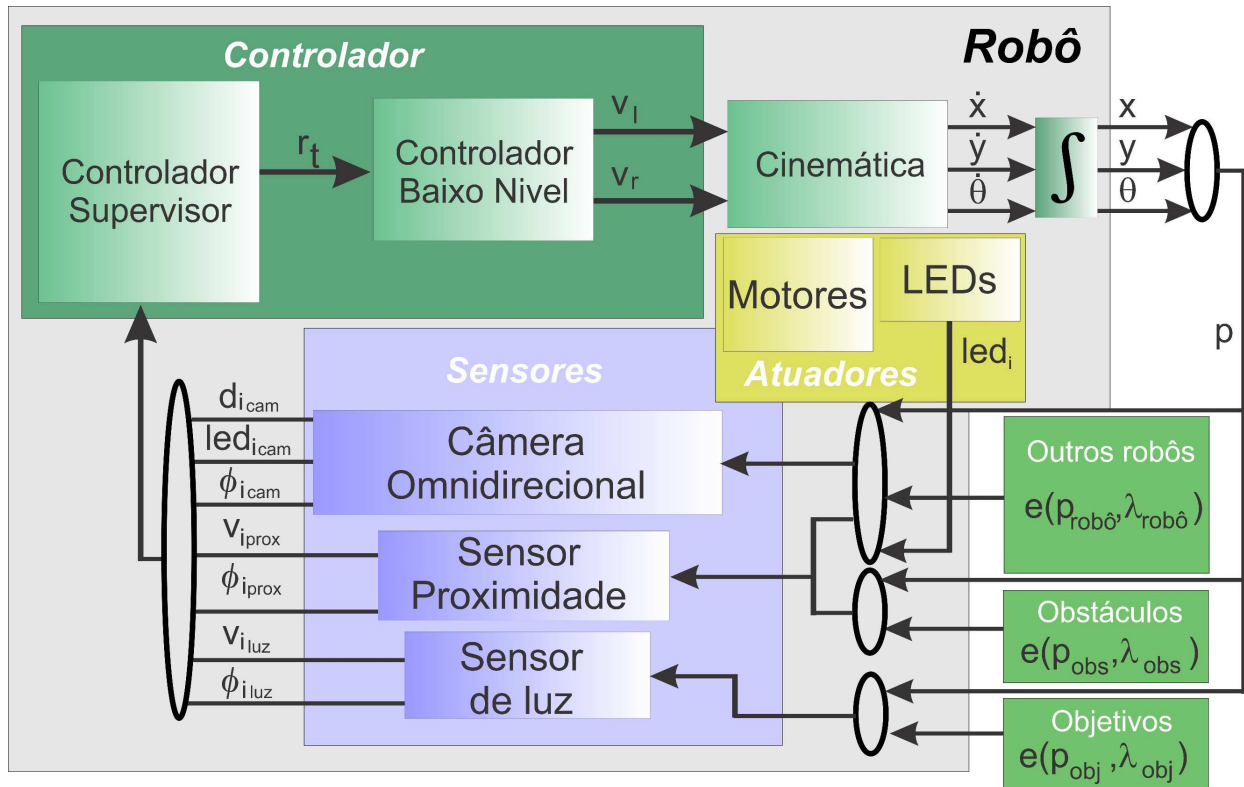
Nesta tese as limitações anteriormente mencionadas não aparecem explicitamente na maioria dos experimentos. Especificamente, os cenários onde os robôs desenvolvem as tarefas são estruturados, os robôs se deslocam a baixa velocidade e alguns dos experimentos utilizam projeto automático *offline* de controladores para os robôs do enxame. Estas características juntas atenuam fortemente as limitações do método.

5.3 Sistema de controle do robô

Nos experimentos que serão realizados nas subseqüentes seções deste capítulo utilizaram o sistema de controle definido nesta seção. O sistema de controle do robô se encarrega de definir e coordenar os movimentos do robô. Este sistema permite que o enxame de robôs realize movimento coletivo com coesão. O sistema é composto por um controlador de baixo nível e outro supervisor como observado na Figura 5.2.

O controlador de baixo nível permite realizar a tradução das informações do controlador supervisor em comandos para as rodas do robô, este controlador se comunica com o hardware e é dependente da plataforma, ou seja, se é utilizado outro tipo plataforma não compatível com o controlador, este controlador deverá ser modificado. O controlador recebe o ângulo θ_d que deverá ser minimizado para o robô seguir a trajetória de movimento. A saída do controlador de baixo nível são as velocidades angulares das rodas v_l e v_r que permitem o deslocamento do robô. Na Seção 4.2.1 é detalhado este controlador por intermédio do diagrama de blocos e as equações matemáticas.

Figura 5.2: Sistema de controle apresentando o controlador do robô e os sensores e atuadores utilizados.



Fonte: Elaborada pelo autor.

O controlador supervisor (Seção 5.3.1) se encarrega da composição dos comportamentos num vetor direção que define o sentido de movimentação do robô. Este controlador é independente da plataforma, não exigindo modificação caso seja escolhida outro robô. Este controlador recebe como entrada variáveis que são as medidas obtidas pelos sensores de proximidade, sensores de luminosidade e a câmera omnidirecional. O sistema de controle atua sobre os motores e sobre os LEDs do robô.

Os sensores realizam o sensoriamento de estímulos no cenário de trabalho do enxame de robôs. Os estímulos detectados são estímulos de outros robôs no cenário, $e(p_{robô}, \lambda_{robô})$, estímulos de obstáculos $e(p_{obs}, \lambda_{obs})$ e estímulos de alvos $e(p_{obj}, \lambda_{obj})$. As medições adquiridas pela câmera omnidirecional são as distâncias relativas entre robôs ($v_{i_{cam}}$), orientação relativa entre robôs ($\phi_{i_{cam}}$) e valores de cor dos LEDs dos robôs vizinhos ($led_{i_{cam}}$). A modelagem destes sensores foi realizada na Seção 5.1.2.

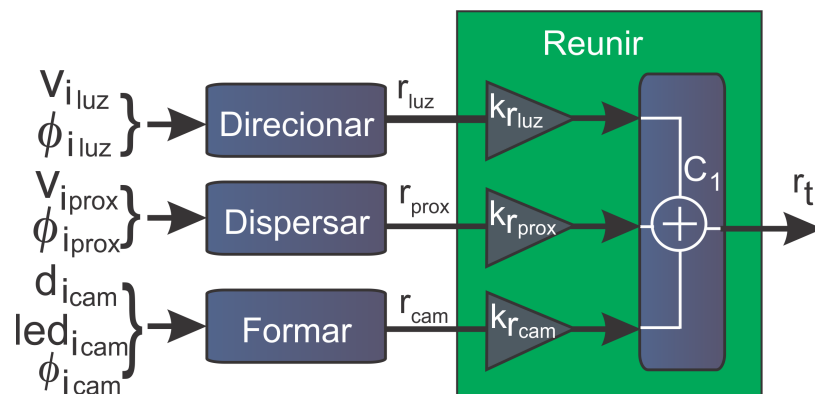
Por sua vez, o sensor de proximidade captura a presença de objetos em curtas distâncias. Estes objetos podem ser obstáculos ou outros robôs no cenário. As variáveis medíveis pelo sensor são a presença de objetos ($v_{i_{prox}}$) e orientação relativa de obstáculos ($\phi_{i_{prox}}$). Finalmente, o sensor de luminosidade detecta a presença de alvos luminosos no cenário. Assim, o sensor mede a presença de alvos ($v_{i_{luz}}$) e orientação relativa do alvo ($\phi_{i_{cam}}$).

Na Figura 5.2 observa-se também o bloco de modelagem cinemática do robô que recebe as velocidades lineares das rodas do robô e obtém as variáveis de estado pose do robô em relação ao sistema de coordenadas O_w . Este bloco foi previamente definido na Seção 5.1.1.

5.3.1 Controlador supervisor

O controlador supervisor do robô é o encarregado de especificar os comportamentos do robô para se locomover e interagir com o ambiente. Na Figura 5.3 observa-se o controlador supervisor do robô dividido em quatro comportamentos chamados: *direcionar*, *dispersar*, *formar* e *reunir*.

Figura 5.3: Controlador supervisor apresentando os comportamentos *direcionar*, *dispersar*, *formar* e *reunir* com seus vetores de entrada e saída para um único robô.



Fonte: Elaborada pelo autor.

Comportamento para manter coesão: *formar*

O comportamento *formar* permite que os robôs mantenham uma distância particular de seus vizinhos com o intuito de evitar colisões entre si e assegura a coesão a nível de enxame.

O sensor que permite capturar os dados do comportamento é uma câmera omnidirecional. Neste comportamento foi utilizada a função de Lennard-Jones previamente explicitado na Seção 5.2.1.

Na Figura 5.3 se observa as entradas e saídas desse bloco. As entradas do comportamento são a cor $led_{i_{cam}}$ dos LEDs dos robôs vizinhos, a distância $v_{i_{cam}}$ e o ângulo $\phi_{i_{cam}}$ aos LEDs dos robôs vizinhos. A a cor dos LEDs é comparada com a cor escolhida do enxame para verificar se o robô pertence ao enxame. Neste experimento a cor escolhida do enxame foi vermelha. A ganancia foi escolhida em $\epsilon = 25$ e a distância desejada entre robôs em $d^* = 0,75 m$. A saída do bloco é o vetor r_{cam} com a orientação e magnitude que o robô deve manter para atingir a coesão entre vizinhos.

Comportamento para evitar colisões com obstáculos: *dispersar*

O comportamento *dispersar* permite que o robô se afaste de obstáculos com o intuito de evitar colisões. Embora o comportamento *formar* permita evitar colisões entre robôs, o mesmo não permite evitar colisões com obstáculos. Então, é necessário definir esta nova restrição de distância no controlador. A saída do comportamento é o vetor r_{prox} expressado na Equação (5.3).

$$r_{prox} = \begin{cases} \frac{1}{24} \sum_{i=1}^{24} v_{i_{prox}} e^{-j\phi_{i_{prox}}} & |r_{prox}| > 0,1 \\ 0e^{j0} & cc \end{cases} . \quad (5.3)$$

Os sensores usados no comportamento *dispersar* são os 24 sensores de proximidade do robô. Cada medida dos sensores de proximidade $i \in \mathbb{N} : [1,24]$ esta limitada no intervalo $0 \leq v_{i_{prox}} \leq 1$. A variável $v_{i_{prox}}$ define o grau de proximidade do sensor i com a entidade mais próxima: obstaculo ou robô. Os ângulos $\phi_{i_{prox}}$ fornecem a orientação da entidade mais próxima. A Equação (5.1) anteriormente apresentada modela as medições $v_{i_{prox}}$ do sensor.

Comportamento para atingir objetivos: *direcionar*

O comportamento *direcionar* permite definir uma orientação objetivo para o robô se movimentar. O robô esta equipado com $j = 24$ sensores de luz. O comportamento captura por intermédio desses sensores a intensidade de cada fonte luminosa que serão a entrada do

bloco e r_{luz} será a saída do bloco. O vetor orientação r_{luz} é definido na Equação (5.4).

$$r_{luz} = \begin{cases} \frac{1}{24} \sum_{i=1}^{24} v_{i_{luz}} e^{j\phi_{i_{luz}}} & |r_{luz}| > 0,1 \\ 0e^{j0} & \text{cc} \end{cases}. \quad (5.4)$$

Os sensores usados neste comportamento *direcionar* são os 24 sensores de luz presentes no robô. Cada medida dos sensores de proximidade $i \in \mathbb{N} : [1,24]$ esta limitada no intervalo $0 \leq v_{i_{luz}} \leq 1$. A variável $v_{i_{luz}}$ foi definida na Equação (4.8) e os valores das constantes foram definidos na Seção 5.1.2.

Comportamento de coesão para movimento coletivo: *reunir*

O comportamento *reunir* é a composição ponderada dos comportamentos acima citados obtida por intermédio do coordenador C_1 . Esta composição permite a orientação do robô no sentido de um objetivo, evitando obstáculos e mantendo uma coesão (distância) com os vizinhos. Este controlador permite a navegação coesa e ordenada seguindo um padrão de formação espacial hexagonal produto do comportamento emergente obtido quando o controlador é implementado em todos os robôs do enxame. Possui como entrada os comportamentos acima mencionados com os respectivos vetores r_{luz} , r_{prox} e r_{cam} . O vetor resultante r_t se define como na Equação (5.5) e é a saída do comportamento *reunir*.

$$r_t = k_{r_{luz}} r_{luz} + k_{r_{prox}} r_{prox} + k_{r_{cam}} r_{cam}. \quad (5.5)$$

Este controlador permite a navegação coesa e ordenada seguindo um padrão de formação espacial hexagonal produto do comportamento emergente obtido quando o controlador é implementado em todos os robôs do enxame. As constantes foram definidas em $k_{r_{luz}} = 0,4$, $k_{r_{prox}} = 0,5$ e $k_{r_{cam}} = 0,9$.

5.3.2 Controlador de baixo nível

O controlador de baixo nível permite controlar o ângulo de guinada do robô e a velocidade do robô para atingir a direção e velocidade desejada. O controlador de baixo nível utilizado neste capítulo foi definido na Seção 4.2.1. O ciclo do controlador foi definido em $T_a = 0,1$ s.

Os ganhos do PID foram ajustados em $k_v = 8$ para o controle proporcional de velocidade linear e $k_\omega = 1,1 s^{-1}$ para o controle proporcional de velocidade angular. Estes valores apresentam uma resposta suave nos robôs físicos.

5.4 Resultados experimentais

Nesta Seção será realizada uma série de experimentos com o controlador projetado manualmente (Seção 5.3) para cada robô do enxame. O propósito é avaliar a capacidade da função de Lennard-Jones de produzir coesão entre os robôs. Logo disto, é avaliada a capacidade do robô de se movimentar no ambiente de forma conjunta e coesa para atingir uma orientação predefinida (o alvo a ser atingido e mantido pelo enxame). A relevância destes experimentos é mostrar a viabilidade da definição das restrições de distâncias feitas no controlador *formar* projetado neste capítulo. Este controlador será logo incluído numa arquitetura modular e em outra monolítica, em capítulos subsequentes desta tese.

As tarefas propostas neste capítulo abstraem tarefas de relevância na robótica, tais como o resgate de vítimas em ambientes hostis para seres humanos, monitoramento de perímetro, busca de químicos que são tóxicos, desminagem, dentre outras tarefas coletivas. A diferença do trabalho realizado anteriormente [49], no qual é abordado o movimento coletivo desde a perspectiva de manter unicamente a conectividade no sistema, nesta seção se busca que os robôs se desloquem mantendo distâncias específicas, ou seja, se desloquem com coesão. Por sua vez, no trabalho em Mendiburu, Morais e Lima[49], se aborda o problema de movimento coletivo com técnicas de composição temporal de comportamentos (Seção 3.3.1) e leis de controle baseadas também em campos potenciais. A diferença do supervisor do artigo em Mendiburu, Morais e Lima[49] e do apresentado nesta seção é que aqui é um supervisor que realiza a composição cooperativa de comportamentos tal como foi explicado na Seção 5.3.1.

Para a realização de movimento coletivo em sistemas de vários robôs é desejado que o enxame se desloque em formação. Formações permitem que conjuntos de robôs possam navegar de forma coesa embora alguns dos indivíduos do enxame não tenham acesso direto à direção de navegação. Assim, estes robôs somente deverão manter a coesão e indiretamente estariam se deslocando na direção de interesse. Por sua vez, a coesão entre robôs permite uma

forma de deslocamento eficiente reduzindo a interferência física entre os robôs do enxame.

Os algoritmos apresentados nas seções anteriores serão implementados por intermédio de simulação em computador usando o simulador ARGoS [71, 72] e a programação realizada na linguagem C++. O simulador ARGoS é propício para a simulação de enxames de robôs pois oferece escalabilidade no que diz respeito à quantidade de robôs que operam no cenário. Por sua vez, possui flexibilidade no que diz respeito à modularidade do simulador [25]. O ARGoS se diferencia de outros simuladores tais como o Gazebo em que Gazebo prioriza a precisão da simulação com o custo inerente de possuir limitada escalabilidade [25]. Outros simuladores, tais como o Stage, embora possuam alta capacidade de escalabilidade, sofrem com a falta de generalidade das aplicações que podem ser feitas por ele, possuindo limitada flexibilidade [25]. Comparando V-REP com ARGoS, ARGoS prioriza desempenho em troca de modelos complexos do robô e do ambiente [120] e possui mais flexibilidade para gerar experimentos sucessivos por linha de comando ao invés da própria interface gráfica [120]. Além disso, ante simulações exaustivas e com grande quantidade de robôs, como as que são realizadas normalmente no projeto automático de controladores, o simulador ARGoS é o que apresenta melhor desempenho comparado com V-REP e Gazebo [120]. Finalmente, a transferência do controlador projetado para os robôs físicos é realizada de forma direta, sem modificação no código [71, 72].

Os experimentos estão divididos nas seguintes etapas:

- Utilizar o comportamento *formar* no cenário sem obstáculos.
- Utilizar o comportamento *direcionar* e *dispersar* no cenário sem obstáculos.
- Utilizar o comportamento *reunir* no cenário sem obstáculos.
- Utilizar o comportamento *reunir* no cenário com obstáculos.

O intuito destes experimentos é mostrar a viabilidade e necessidade de utilizar um comportamento com restrição de distâncias que permita a coesão entre os indivíduos. A relevância é apresentada em experimentos nos quais os comportamentos atuam de forma separada e logo conjunta. Estes experimentos serão realizados num cenário sem obstáculos e outro com obstáculos. Os experimentos têm duração de 60 s para cenários sem obstáculos e

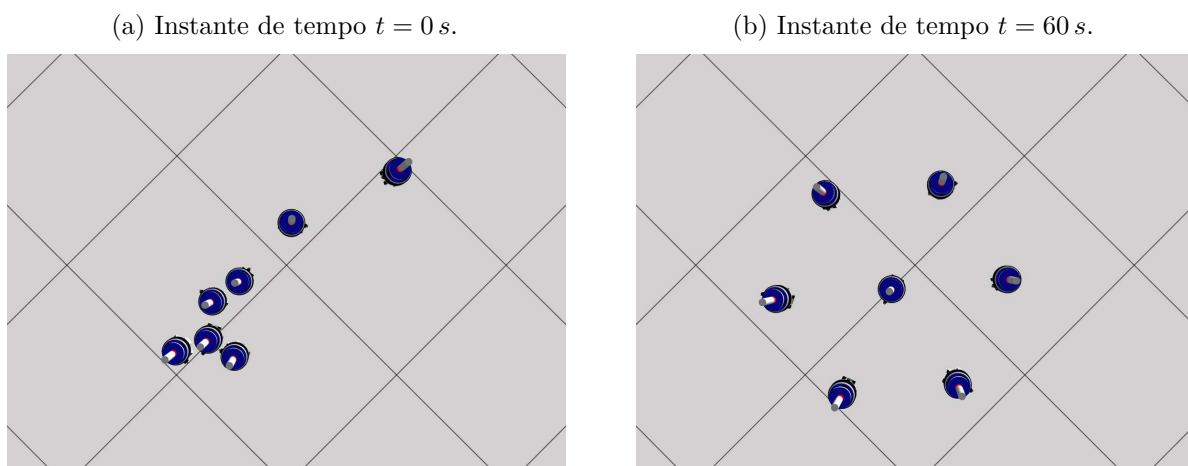
tempos maiores para cenários com obstáculos, como será apresentado nas seguintes seções. Os robôs são inicialmente posicionados com uma densidade de 5 robôs/ m^2 tanto para 7 robôs como para 40 robôs. Foi implementado em 7 e em 40 robôs para observar a escalabilidade do comportamento coletivo.

5.4.1 Comportamento *formar* no cenário sem obstáculos

O comportamento *formar* é implementado, para que os robôs mantenham uma distância específica entre os vizinhos próximos. Os outros comportamentos não foram implementados neste experimento. O intuito é observar o funcionamento do comportamento isoladamente. O experimento consiste em 60 s de simulação, na qual os robôs interagem para finalmente realizarem uma formação com coesão.

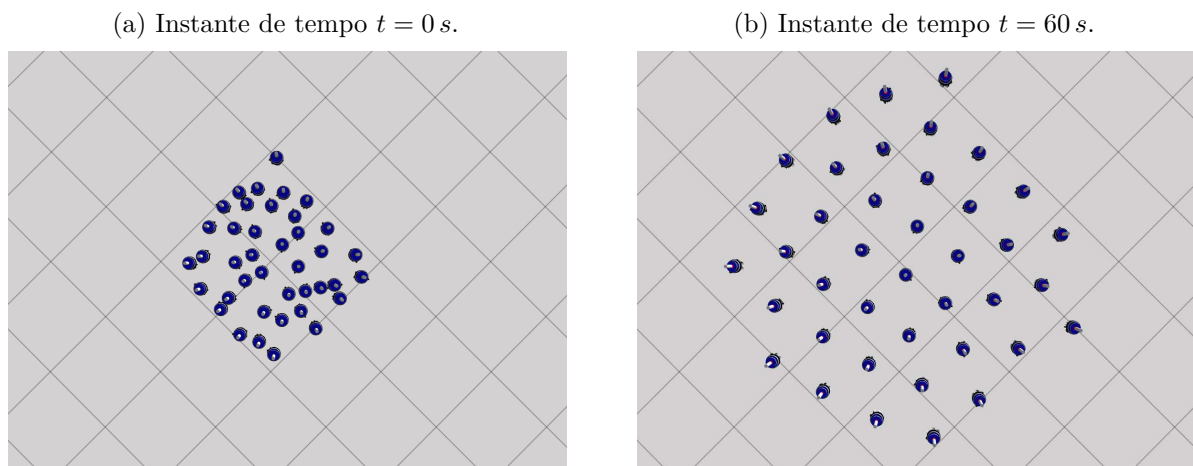
Na Figura 5.4 observa-se o instante de tempo inicial e final que mostra a formação hexagonal com 7 robôs, obtida da implementação do comportamento *formar*.

Figura 5.4: Disposição de 7 robôs que utilizam o comportamento *formar* unicamente.



Fonte: Elaborada pelo autor.

No instante de tempo $t = 0 s$ observa-se o cenário com 7 robôs nas suas posições iniciais. O experimento se inicia e aprecia-se a interação entre os robôs em que forças de dispersão e atração são geradas entre os robôs. A medida que o tempo vai transcorrendo os robôs vão atingindo o ponto de equilíbrio e a intensidade das forças vai diminuindo. Finalmente no instante $t = 60 s$ o enxame de robôs conseguiram atingir a coesão desejada.

Figura 5.5: Disposição de 40 robôs que utilizam o comportamento *formar* unicamente.

Fonte: Elaborada pelo autor.

Observa-se um comportamento similar para 40 robôs na Figura 5.5, mostrando a escalabilidade do comportamento. Foi demorado maior tempo para a estabilização dos movimentos que no caso com 7 robôs. Porém, no final do experimento, observa-se em ambos os casos que é respeitada a distância entre robôs.

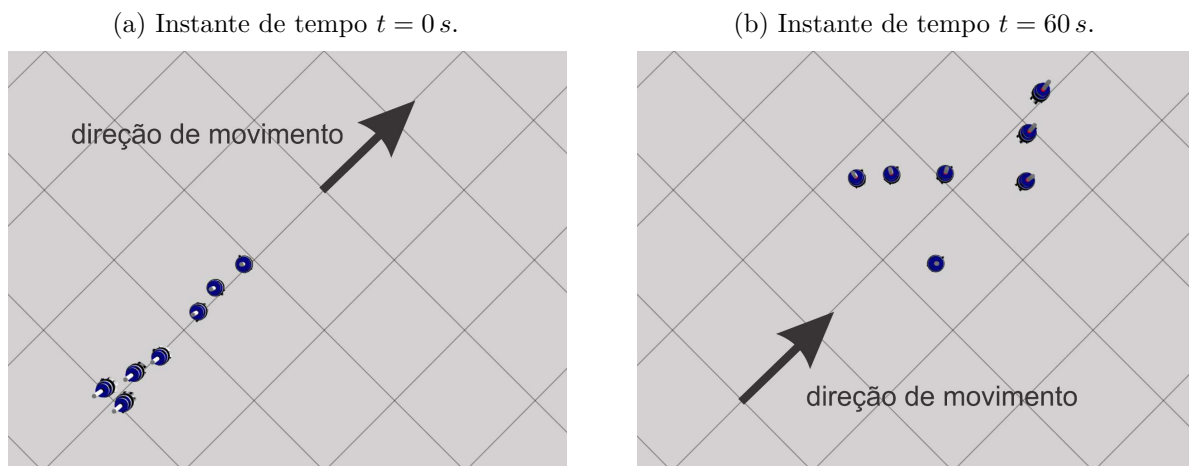
5.4.2 Comportamento *direcionar* e *dispersar* no cenário sem obstáculos

O comportamento *direcionar* e *dispersar* foram implementados conjuntamente, para que os robôs se movimentem para atingir objetivos e por sua vez evitando colisões entre os robôs. O intuito é observar o funcionamento dos comportamentos e mostrar que a coesão entre robôs é necessária. O experimento consiste em 60 s de simulação, na qual os robôs se movimentam em direção a um alvo evitando colisões entre si, mas sem nenhum tipo de coesão aparente.

Na Figura 5.6 observa-se o instante de tempo inicial e final que mostra a formação com 7 robôs, obtida da implementação do comportamento *formar*.

No instante de tempo $t = 0 s$ observa-se o cenário com 7 robôs nas suas posições iniciais. O experimento se inicia e aprecia-se a dispersão entre os robôs que evita os robôs colidirem.

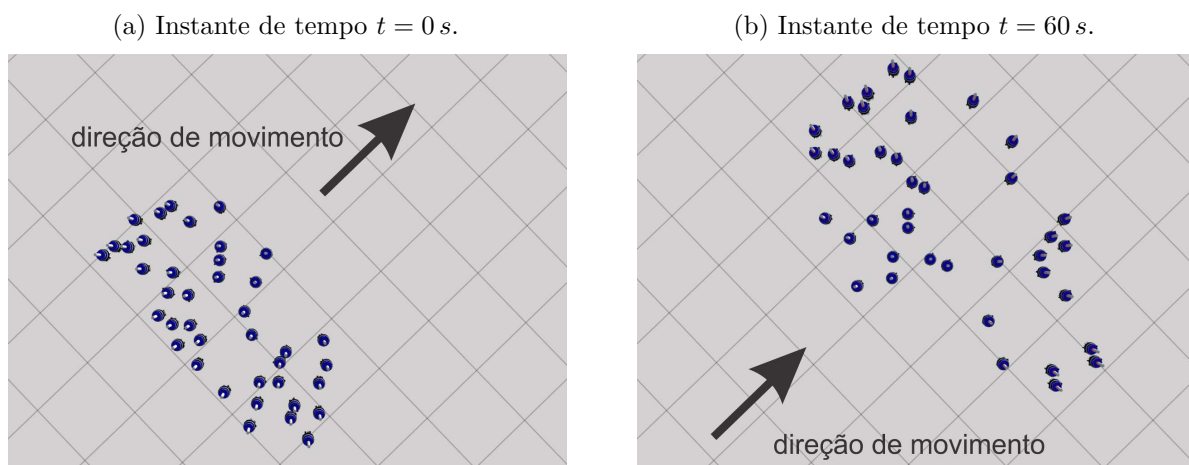
A componente do comportamento *dispersar* vai se reduzindo e a componente do vetor *direcionar* vai aumentando. Logo, uma vez os robôs atingem o equilíbrio no que diz respeito

Figura 5.6: Disposição de 7 robôs que utilizam os comportamentos *direcionar* e *dispersar*.

Fonte: Elaborada pelo autor.

as forças de dispersão, a contribuição de forças se deve unicamente ao comportamento *direcionar*. Finalmente no instante $t = 60 s$ o enxame de robôs conseguiram atingir o alvo desejado.

Observa-se que não existe interação com coesão entre as entidades porque o controle *formar*, que ajusta as interações entre robôs, não foi implementado. Por isto se observa que os robôs não mantêm distâncias determinadas entre si e o movimento é muito flexível, no sentido que os robôs podem perder a conectividade em qualquer momento no movimento.

Figura 5.7: Disposição de 40 robôs que utilizam os comportamentos *direcionar* e *dispersar*.

Fonte: Elaborada pelo autor.

Observa-se um comportamento similar para 40 robôs na Figura 5.7. Os robôs precisaram

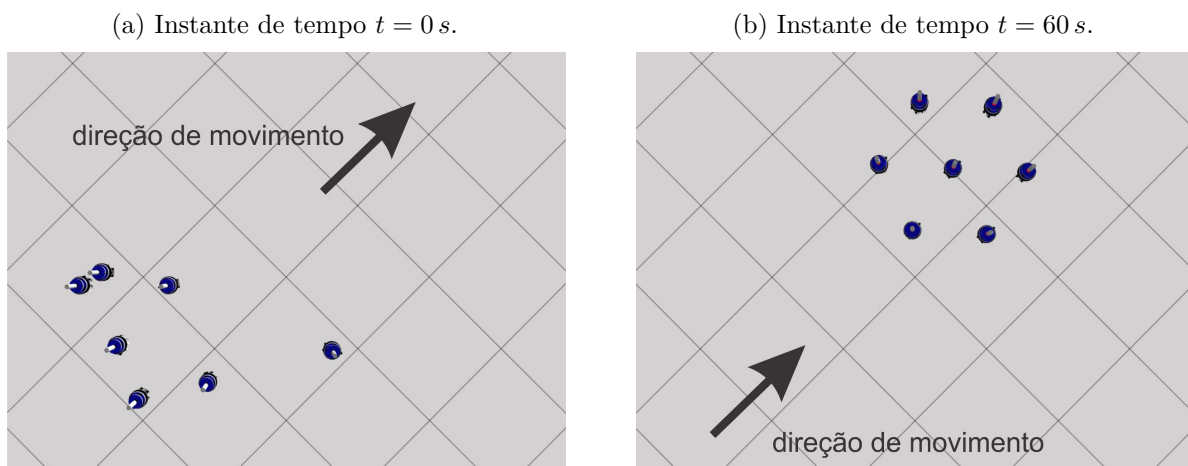
de maior quantidade de tempo para estabilizar as forças de repulsão.

5.4.3 Movimento coletivo com coesão: Comportamento *reunir*

Foi implementado o comportamento *reunir* que combina todos os comportamentos anteriormente analisados. O cenário analisado é um cenário sem obstáculos. O intuito é observar o funcionamento do comportamento no conjunto. O experimento consiste em 60 s de simulação, na qual os robôs se movimentam em direção a um alvo evitando colisões entre si e mantendo coesão entre robôs.

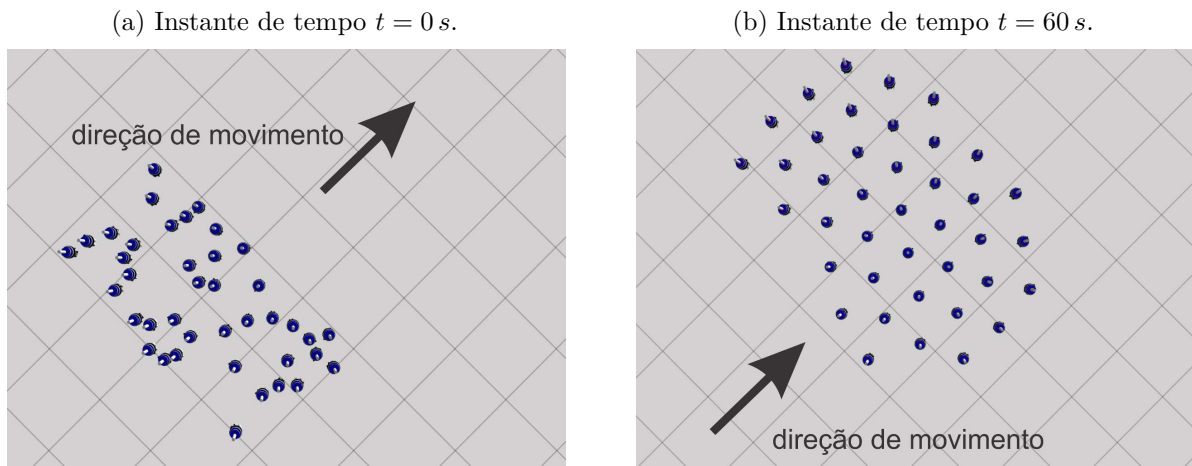
Na Figura 5.8 observa-se o instante de tempo inicial e final que mostra o movimento coletivo com 7 robôs, obtido da implementação do comportamento *reunir*.

Figura 5.8: Movimento coletivo com 7 robôs utilizando o comportamento *reunir*.



Fonte: Elaborada pelo autor.

Primeiramente, na Figura 5.8a observa-se os robôs inicialmente posicionados no cenário, todos os vetores de comportamento tem peso significativo nos cálculos pois os robôs encontram-se em posições aleatórias, longe das posições de equilíbrio para a coesão no movimento. No transcurso do tempo, os robôs vão se acomodando nas suas posições de equilíbrio, os vetores de comportamento r_{prox} e r_{cam} vão reduzindo o valor atingindo valores baixos. Finalmente na Figura 5.8b se atinge o estado de equilíbrio em que se observa claramente a coesão do enxame e ao mesmo tempo se movimentando para o objetivo sem existir colisões entre os robôs.

Figura 5.9: Movimento coletivo com 40 robôs utilizando o comportamento *reunir*.

Fonte: Elaborada pelo autor.

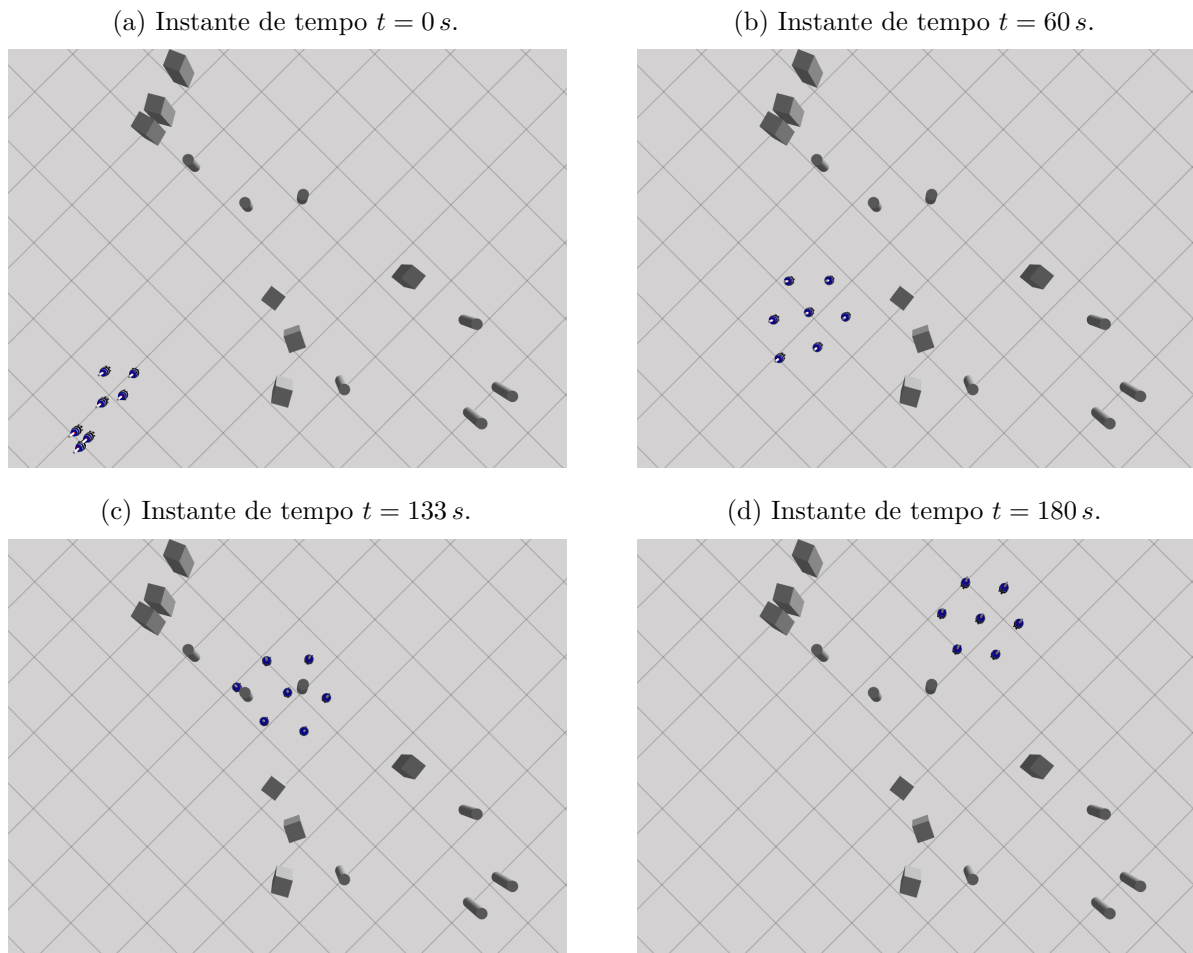
Na Figura 5.9 observa-se o movimento coletivo com coesão para o caso de 40 robôs. Neste experimento é válida a mesma análise que para o caso de 7 robôs. Porém, os robôs precisaram de maior quantidade de tempo para neutralizar as forças de coesão e de repulsão para que o vetor de direção seja preponderante. Analogamente os casos acima citados, pode ser observada a escalabilidade do comportamento coletivo.

Nestes experimentos pode ser observada a relevância da utilização do comportamento *formar*. Nos experimentos com 7 e 40 robôs foi observado que existem oclusões entre a fonte luminosa e alguns robôs do enxame. Esta oclusão é mais frequente com os robôs da parte traseira do enxame, pois os da frente interferem na detecção da fonte luminosa. Quando uma oclusão total acontece em algum robô ($r_{luz} = 0$), o robô perde noção da direção de deslocamento. Porém, as restrições de distâncias colocadas no comportamento *formar* compensam essa oclusão e fazem que o robô mantenha a coesão com os vizinhos e, indiretamente, se desloque em direção ao objetivo.

Nos seguintes experimentos é apresentado o movimento coletivo coeso do enxame de robôs tendo em conta obstáculos no cenário. Por isto, é aumentado o tempo do experimento para 240 s. São posicionados 7 e 40 robôs de forma aleatória com uma densidade de $5 \text{ robôs}/m^2$.

Na Figura 5.10 apresenta-se o movimento coletivo de 7 robôs com obstáculos no ambiente. Os robôs são posicionados no cenário, iniciam seu deslocamento (Figura 5.10a) e atingem a coesão aproximadamente no instante de tempo $t = 60 s$. Na Figura 5.10c os robôs encontram

Figura 5.10: Movimento coletivo para o caso de 7 robôs no cenário com obstáculos.

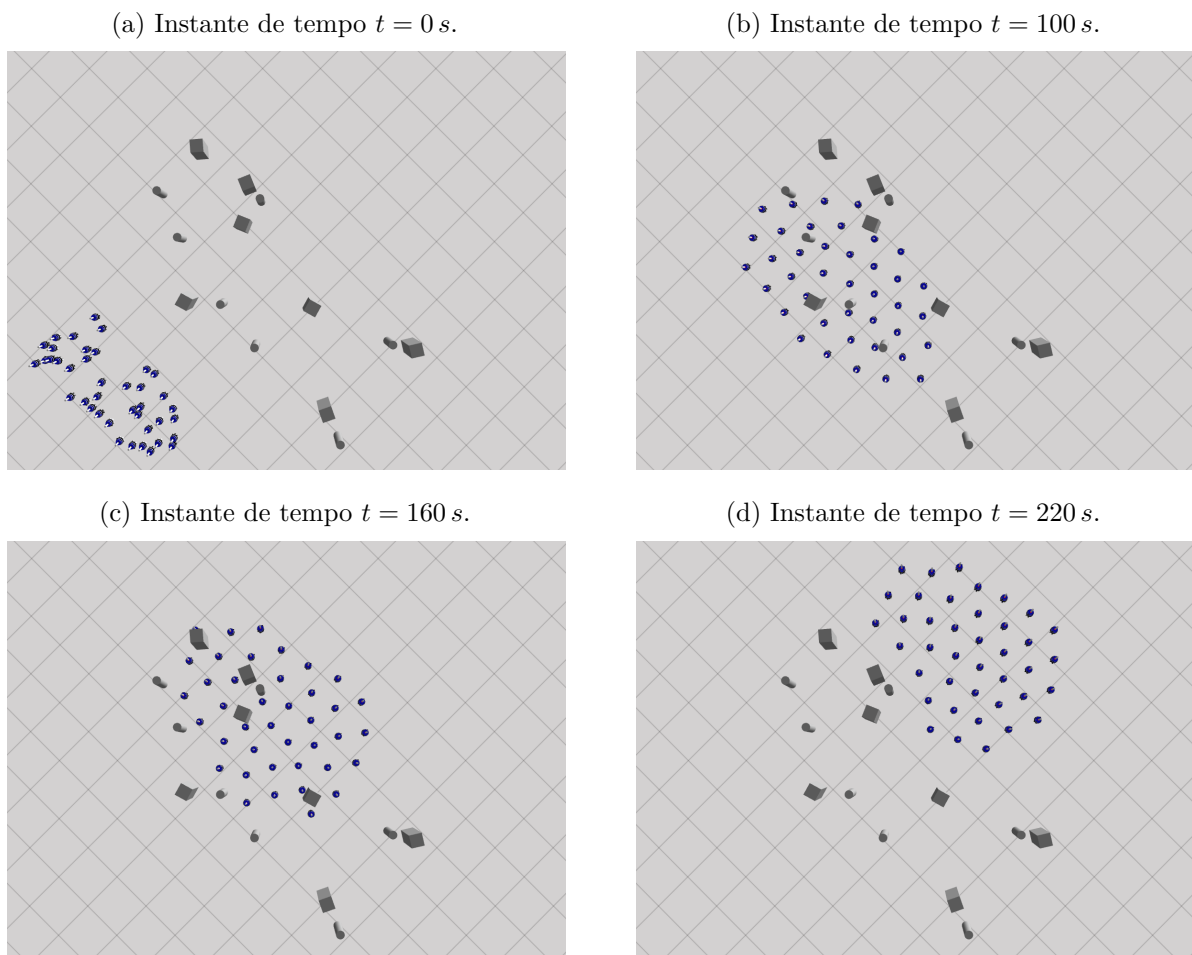


Fonte: Elaborada pelo autor.

os obstáculos e começam realizar as manobras evasivas como observado em $t = 133$ s. Nestes instantes de tempo a influencia do vetor r_{prox} é grande no intuito de evitar as colisões com obstáculos, perdendo levemente o padrão de coesão no propósito de evitar os obstáculos. Finalmente na Figura 5.10d o padrão de formação é atingido novamente ao mesmo tempo que a coesão e deslocamento em direção ao objetivo são bem sucedidos.

Na Figura 5.11 observa-se o movimento coletivo com 40 robôs atuando num cenário com obstáculos. A análise realizada anteriormente para o enxame de 7 robôs é aplicável nesta situação. Porém, neste caso são observadas mais oclusões da fonte luminosa devido aos robôs e obstáculos. Por sua vez, a interferência física entre robôs aumenta o que produz que o destino seja atingido em maior tempo.

Figura 5.11: Movimento coletivo para o caso de 40 robôs no cenário com obstáculos.



Fonte: Elaborada pelo autor.

A relevância do comportamento de coesão permite que a conectividade entre robôs não seja perdida independentemente da oclusão do objetivo. O comportamento é flexível para evitar os obstáculos, manter a conectividade e por sua vez continuar se movimentando em direção ao alvo.

5.4.4 Grau de acurácia $\delta(t)$ no movimento coletivo

Nesta seção são realizados experimentos quantitativos com 100 robôs em cenários com obstáculos para observar o movimento coletivo do enxame com restrições de coesão entre os robôs. O experimento é executado 100 vezes e os robôs são posicionados no cenário com uma densidade de $5 \text{ robôs}/m^2$, de forma aleatória na origem de coordenadas do sistema de

referência O_w . Os obstáculos são posicionados de forma aleatória numa área de $56 m^2$ com centro em $[10; 0] m$ sendo um retângulo de $14 \times 4 m$. Os obstáculos são definidos como 7 caixas de tamanho $0,3 \times 0,3 \times 0,5 m$ e 7 cilindros de tamanho $0,1 m$ de radio e $0,4 m$ de altura que também são posicionados aleatoriamente nesta região retangular. Os 100 experimentos tem duração de $420 s$.

A métrica utilizada para medir a qualidade do movimento coletivo é o grau de acurácia $\delta(t)$ [136]. O grau de acurácia $\delta(t)$ é definido na Equação (5.6) e permite medir o grau de alinhamento do enxame em direção ao objetivo e também ter em conta o grau de alinhamento entre robôs. Este indicador possui valores no intervalo $[0,1]$.

$$\delta(t) = 1 - \frac{1 - \rho_g(t) \cos(\phi_g(t) - \psi_g(t))}{2}, \quad (5.6)$$

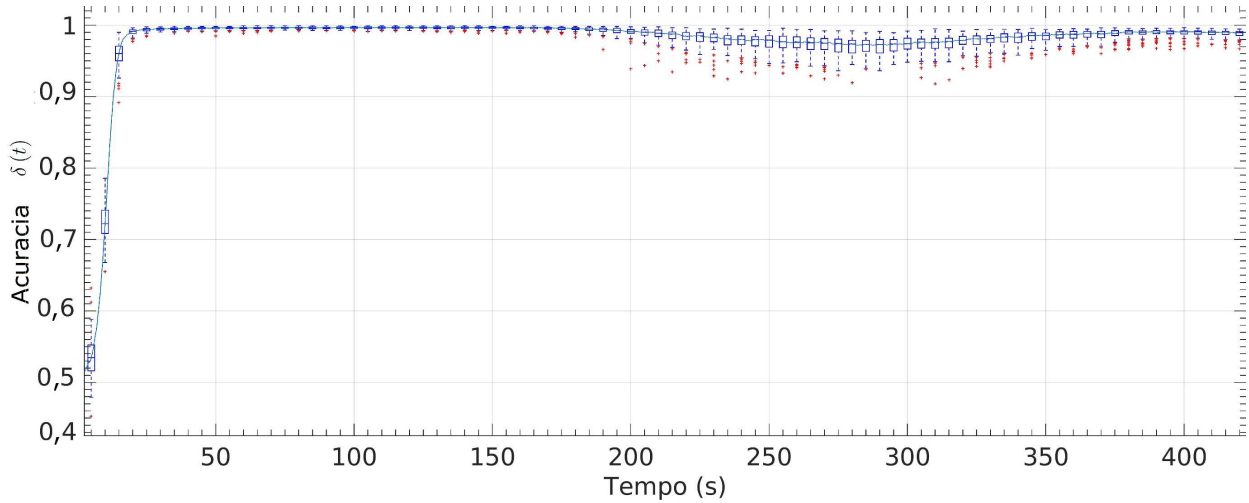
em que $\phi_g(t) = \text{atan2}\left(\sum_1^N o_i(t)\right)$ é o ângulo de guinada do enxame, o_i é o vetor orientação unitário de cada um dos N robôs do enxame expressado no sistema de coordenadas O_w . O ângulo desejado ao objetivo é definido como $\psi_g(t)$. Por sua vez, $\rho_g(t)$ é o grau de orientação dos robôs. O grau de orientação $\rho_g(t)$ mede o alinhamento entre os robôs sendo definido na Equação (5.7). Um grau de orientação alto indica que os robôs do enxame estão alinhados numa orientação comum. O grau de orientação $\rho_g(t)$ possui valores definidos entre $0 \leq \rho_g(t) \leq 1$.

$$\rho_g(t) = \frac{1}{N} \left| \sum_{i=1}^N o_i(t) \right|, \quad (5.7)$$

Na Figura 5.12 observa-se o grau de acurácia $\delta(t)$ para 100 robôs em função do tempo. É apresentado diagramas de caixas que mostram a mediana do grau de orientação, os quartis e os valores atípicos a cada $5 s$.

Observando os dados estatísticos, pode ser constatado que o enxame começa praticamente desordenado sem uma orientação definida ($t = 0 s$). Como o enxame possui vários robôs e obstáculos existem oclusões do alvo para alguns dos integrantes do enxame. A medida que os robôs da frente do enxame vão visualizando o objetivo, vão se alinhando na direção dele. Logo, os robôs que não visualizam o objetivo realizam a coesão com os robôs vizinhos e indiretamente começam se orientar na direção ao objetivo. Uma acurácia alta começa ser

Figura 5.12: Grau de acurácia $\delta(t)$ no movimento coletivo de 100 robôs no cenário com obstáculos.



Fonte: Elaborada pelo autor.

atingida a partir dos 25 s.

Pode-se observar que um alto valor de $\rho_g(t)$ é atingido com dificuldade devido as interações iniciais entre os robôs ($t = 0 s$ a $t = 25 s$). Observa-se também que os robôs da frente do time começam a encontrar os obstáculos em $t = 170 s$ e iniciam-se as manobras para evitar as colisões. Seguido disso os integrantes do time vão passando pela área com obstáculos. A característica flexível do comportamento de coesão permite que a conectividade entre os robôs se reduza no compromisso de evitar os obstáculos. Neste estágio o valor de $\rho_g(t)$ cai pelas interações entre robôs e dos robôs com o ambiente ($t = 170 s$ a $t = 370 s$). Logo que os robôs terminaram de passar pelos obstáculos o grau de orientação começa a aumentar novamente, voltando a ser atingida um $\rho_g(t)$ alto, a partir de $t = 380 s$ aproximadamente.

O comportamento de coesão neste experimento específico foi importante para estabilizar as interações entre o enxame, permitiu que robôs ocluídos não percam conectividade do enxame e permitiu que os robôs na passagem de obstáculos continuem mantendo coesão.

5.5 Considerações finais

Neste Capítulo foi estudada a viabilidade e necessidade do emprego de comportamentos com restrições de distâncias para permitir coesão de robôs no movimento coletivo de um enxame de robôs. O intuito foi mostrar a relevância por intermédio de experimentos no ambiente de simulação.

Primeiramente, foi realizada a modelagem cinemática dos robôs e foram modelados os sensores utilizados: câmera, sensor de proximidade e sensor de luminosidade. Por sua vez, foi modelado os atuadores das rodas. Logo, foi definido um comportamento, chamado de *formar* que permite modelar as interações entre robôs do enxame. No módulo é escolhida a lei de campos potenciais artificiais e justificada a sua escolha. Logo após, foi explicitada a função de Lennard-Jones como lei de controle dos robôs do enxame. Foram apresentados as vantagens e desvantagens da lei de controle. Após, foi apresentado o sistema de controle utilizado: o controlador supervisor do robô que está composto por uma série de comportamentos que possuem restrições de distâncias na sua modelagem. Logo, foi definido o controlador de baixo nível do robô que permite fazer a ponte entre os vetores de direção do comportamento do robô e os comandos que devem ser enviados para os atuadores do robô.

Foram apresentados os resultados experimentais realizados em simulação. Dos experimentos sem obstáculos no ambiente, foi concluído que o comportamento *formar* possui bom desempenho isoladamente e conjuntamente com os outros comportamentos de restrições de distâncias modelados. Por sua vez, foi mostrada que a coesão entre robôs é necessária para manter controladas as interações entre robôs para evitar colisões entre eles. Dos experimentos com obstáculos no ambiente, foram observados todos os comportamentos projetados funcionando conjuntamente. Foi realizada uma análise quantitativa e qualitativa do comportamento coletivo observado do enxame. Foi concluído que a coesão entre robôs permitiu que robôs não informados do objetivo (por oclusões devidas a obstáculos e outros robôs) conseguissem se deslocar seguindo os seus robôs vizinhos do enxame. O comportamento *formar* apresentou a flexibilidade necessária para um movimento coletivo coeso com alta conectividade, e por sua vez, a capacidade de manobrabilidade para evitar colisões.

Capítulo 6

Extensões com restrições de distâncias

Da apresentação feita no Capítulo 4, foi corroborado que tanto a abordagem de robótica evolutiva avaliada, *EVOSTICK*, como a abordagem modular, *AUTOMODE-CHOCOLATE*, apresentam deficiências para resolver problemas com restrições de distância, especificamente aqueles que os robôs devem manter posições relativas entre eles com certa coesão, para por exemplo realizar cobertura de uma região determinada.

Foi concluído que esta deficiência se deve as restrições impostas nas relações de entrada e saída das arquiteturas que faz que o comportamento coletivo não resolva a tarefa. Como apresentado na Seção 3.8.2, em *AUTOMODE* se define os módulos de comportamento individuais, as transições entre eles, a quantidade máxima de estados e transições de saída possíveis na máquina de estados. Por outro lado, em *EVOSTICK* é definido o tipo de topologia de rede sem nós ocultos. Também, são definidas as entradas, as saídas, as suas conexões e a quantidade das mesmas. As entradas foram definidas como apresentado na Seção 3.8.2. A especialização de ambas as arquiteturas não permitiu ajustar corretamente as interações entre robôs e entre os robôs com o ambiente para produzir o comportamento coletivo que resolva as tarefas coletivas que requerem coesão. Como foi observado dos resultados experimentais, em *EVOSTICK* o poder de representação é maior e permitiu ajustar as interações de forma granular que possibilitou um melhor desempenho nas tarefas, porém não conseguiu produzir coesão entre os robôs.

Visando o problema da falta de coesão dos robôs tanto nos controladores produzidos pelos métodos AUTOMODE e EVOSTICK, foi feita uma proposta de viabilidade de um comportamento chamado *formar*. Como foi mostrado, dos resultados experimentais do Capítulo 5, o módulo *formar* permitiu produzir coesão entre os robôs do enxame e permitiu manter controladas as interações entre os robôs do enxame para evitar colisões entre os robôs. Por sua vez, permitiu que robôs não informados do alvo a ser atingido realizem movimento coletivo seguindo a outros robôs do enxame que possuem esse conhecimento. Do Capítulo 5 foi observado que o comportamento *formar* teve bom desempenho isoladamente, assim como também em conjunto com outros comportamentos. O comportamento *formar* foi projetado com uma lei de controle independente da plataforma robótica, flexível, parametrizável, contínua na região de operação e principalmente implementável em robôs físicos. Pelo anterior, esta lei de controle se apresenta apropriada para ser utilizada em outros problemas de robótica de enxame, como o problema da especialização dos métodos AUTOMODE e EVOSTICK, que não permite produzir coesão nos robôs do enxame.

Para abordar as limitações da especialização do método AUTOMODE-CHOCOLATE, neste capítulo é proposta uma extensão chamada de AUTOMODE-MATE, uma abordagem apropriada para problemas em que a coesão dos robôs é de relevância na tarefa coletiva. MATE inclui o comportamento denominado *formar* como uma nova forma de restrição de distâncias no método de projeto automático. Nesta nova abordagem, os robôs podem ser organizados como um agregado de indivíduos em um padrão de rede. Este módulo é construído com base em interações locais que são fundamentais para promover tolerância a falhas e escalabilidade no sistema. Por sua vez, é definido o método EVOSPACE. O EVOSPACE inclui o comportamento *formar* na entrada da rede neuronal para permitir realizar comportamentos de coesão no enxame. EVOSPACE utiliza o mesmo modelo de referência que o método MATE.

Ambas abordagens serão comparadas com o método AUTOMODE-CHOCOLATE. O método foi previamente definido na bibliografia [18] e explicado em detalhe na Seção 3.7 desta tese. São utilizados 20 robôs e-puck para avaliar os controladores de software obtidos dos três métodos em três missões diferentes. Os resultados mostram que o método AUTOMODE-MATE possui controladores com bons desempenhos e significativamente melhores aos controladores dos métodos CHOCOLATE e EVOSPACE, de acordo com o teste estatístico

de Wilcoxon, com 95 % de significância [123]. O método MATE produz comportamentos coletivos relevantes para o problema de coesão nos robôs do enxame.

Este capítulo está dividido em quatro seções. Na Seção 6.1 se descreverá a especialização realizada tanto para MATE como para EVOSPACE. Logo, na Seção 6.2 será especificada a metodologia de como serão realizados os experimentos, como serão avaliados os controladores a projetar e as missões nas quais serão projetados os controladores. Logo após, na Seção 6.3 serão apresentados os resultados experimentais. Finalmente, na Seção 6.4 serão apresentadas as considerações finais do capítulo.

6.1 Especialização de AutoMoDe-Mate e EvoSpace

AUTOMODE-MATE e EVOSPACE, por ser ambos métodos de projeto automático, precisam se especializar como foi explicado na Seção 3.8. Muitas das escolhas realizadas nos métodos MATE e EVOSPACE foram feitas, para fins de comparação, seguindo a metodologia dos métodos CHOCOLATE e EVOSTICK, respectivamente. AUTOMODE se descreve como um método geral de projeto que é independente da tarefa a ser realizada [18]. Como apresentado na bibliografia, a especialização do método AUTOMODE é realizada *a priori* de conhecer o tipo de tarefa a ser abordada. No caso do projeto de MATE a abordagem é a mesma, deve ser projetado tendo uma visão do conjunto de tarefas que poderiam ser abordadas, mais não pensando em alguma dessas tarefas especificamente. Por isto a seguir é definida parte da especialização de MATE e EVOSPACE: *Plataforma robótica, o espaço de busca do controlador, o simulador computacional e o algoritmo de otimização*. As *condições experimentais* e a *função objetivo* fazem parte da tarefa particular definida pelo usuário, serão definidas na Seção 6.2.

Plataforma robótica

Para realizar a especialização do método, primeiramente o especialista define a plataforma robótica que vai utilizar para definir o enxame. Nos experimentos realizados com os métodos MATE e EVOSPACE são utilizados os robôs e-puck (Seção 3.8.1). Os robôs e-puck possuem um conjunto amplo de sensores e atuadores propícios para abordar comportamentos com

restrições de distâncias para os robôs manter coesão e desenvolver tarefas.

Os modelos de robôs, sensores e atuadores utilizados nos métodos MATE e EVOSTICK foram os mesmos que os definidos no Capítulo 4: O modelo cinemático foi definido na Seção 4.1.1 e os modelos de sensores e atuadores foram definidos na Seção 4.1.2.

De acordo como explicado na Seção 3.7.1, deve ser definido o poder de representação da plataforma. Para isto, será definido o modelo de referência tanto para MATE como para EVOSPACE na Tabela 6.1

Tabela 6.1: Modelo de referência para o robô e-puck (RM1.3).

Tipo de entrada (sensor)	Variável	Valores
Proximidade	$[r_{prox} , \angle r_{prox}]$	$v_{i_{prox}} \in \mathbb{R} : [0,1]$ $\phi_{i_{prox}} \in [-\pi, \pi] rad$
Luminosidade	$[r_{luz} , \angle r_{luz}]$	$v_{i_{luz}} \in \mathbb{R} : [0,1]$ $\phi_{i_{luz}} \in [-\pi, \pi] rad$
Chão	gnd	$gnd \in \mathbb{R} : [0,0; 0,5; 1,0]$
Proximidade relativa	$[r_{R\&B} , \angle r_{R\&B}]$ $n_{R\&B}$	$v_{i_{R\&B}} \in \mathbb{R} : (0; 0,4] m$ $\phi_{i_{R\&B}} \in \mathbb{R} : [-\pi, \pi] rad$ $n_{R\&B} = \{0,1,\dots,20\}$
Posição relativa	$[r_{cam} , \angle r_{cam}]$	$v_{i_{cam}} \in \mathbb{R} : [0,05; 0,30] m$ $\phi_{i_{cam}} \in \mathbb{R} : [-\pi, \pi] rad$
Tipo de saída (atuador)	Variável	Valores
Motores das rodas	$v_i, i \in \{l,r\}$	$v_i \in \mathbb{R} : [-0,12; 0,12] m/s$
Leds	$led_{i \in \{1,2,3\}}$	$led_i \in \mathbb{N} : [0,1]$

Ciclo de controle: $T_a = 0,1 s$.

Fonte: Elaborada pelo autor.

Os métodos CHOCOLATE e EVOSTICK utilizam o sensor de proximidade relativa para o cálculo da distância relativa aos vizinhos. Esta medida pode ter flutuações dependendo do nível da bateria, devido a que este sensor utiliza a medição da potência do sinal infravermelho para determinar a distância com os robôs vizinhos [103,104]. Para evitar este tipo de flutuação, nos métodos MATE e EVOSPACE foi adicionada a câmera omnidirecional e os LEDs do e-puck. A adição deste sensor é para garantir uma medida mais precisa da distância e independentemente ao estado da bateria do robô. Por sua vez, utilizar a câmera do robô evita lidar com problemas de interferência inerentes dos sensores de proximidade relativa.

Para minimizar as modificações realizadas nos métodos CHOCOLATE e EVOSTICK, e

tendo em conta que unicamente o módulo *formar* precisa de medidas de distâncias precisas, a adição da câmera é realizada unicamente no módulo *formar* tanto para os métodos MATE e EVOSPACE. Por sua vez, o método MATE utiliza o controlador de baixo nível especificado na Seção 4.2.1. Este controlador permite que o robô rotacione no seu próprio eixo e possa corrigir unicamente a sua orientação sem necessidade de se deslocar da sua posição atual. O ciclo do controlador foi definido em $T_a = 0,1 s$ e os ganhos do PID foram ajustados em $k_v = 1$ para o controle proporcional de velocidade e $k_\omega = 0,7 s^{-1}$ para o controle proporcional de velocidade angular, como anteriormente definido na Seção 4.2.1.

As novas variáveis adicionadas são a ativação dos três LEDs simultaneamente, $led_{i \in \{1,2,3\}}$ com $led_i \in \mathbb{N} : [0,1]$. Por sua vez, $v_{i_{cam}} \in \mathbb{R} : [0,05; 0,30] m$, $i \in \mathbb{N} : [1,8]$, definido como o módulo da distância relativa ao LED do robô vizinho i e $\phi_{i_{cam}} \in \mathbb{R} : [-\pi, \pi]$, $i \in \mathbb{N} : [1,8]$, é o ângulo relativo do robô ao LED do robô vizinho i . Tanto $v_{i_{cam}}$ como $\phi_{i_{cam}}$ são utilizadas para definir o vetor resultante r_{cam} (Equação (5.2)). Neste modelo de referência, os três sensores de chão do e-puck retornam um único valor com o estado do chão, $gnd \in \{0,0; 0,5; 1,0\}$, como sendo preto, branco e cinza, respectivamente.

O espaço de busca do controlador

AutoMoDe-Mate

O espaço de busca do controlador é definido na Equação (3.13) e o controlador é uma máquina de estados probabilística finita. Cada transição entre estados acontece com uma certa probabilidade associada [70]. Esta probabilidade é ajustada pelo projeto automático, obtendo um controlador do robô mais geral, no que diz respeito as tarefas coletivas que podem ser abordadas pelo enxame. A definição das transições da máquina de estados é a mesma que a definida para o método AUTOMODE-CHOCOLATE (Seção 3.8.2). A única diferença é que um novo módulo de comportamento é adicionado. O comportamento é denominado de *formar* e foi detalhado na Seção 5.2.1. Nesta seção será descrito o ajuste de parâmetros realizado nesse comportamento.

Formar: Este comportamento permite aos robôs manter distâncias específicas entre eles. Este comportamento foi representado utilizando a função de Lennard-Jones, que modela a interação entre os robôs do enxame. Como mencionado na Seção 5.2.1, a função

se engloba dentre dos métodos de campos potenciais artificiais [37, 38]. O módulo *formar* foi definido como $r_t = r_{cam} - 5r_{prox}$, com r_{cam} definido na Equação (5.2) e r_{prox} definido na Equação (3.14). Embora o vetor r_{cam} possua um componente repulsivo, foi incluído r_{prox} no cálculo de r_t para evitar colisões com obstáculos e ter redundância, caso os robôs se encontrem muito próximos.

O parâmetro d^* representa a distância desejada entre robôs, ajustada automaticamente pelo projeto automático. Este parâmetro pode ser ajustado no intervalo $0,07 \leq d^* \leq 0,20 m$. Embora o método de síntese possa ajustar todos os parâmetros da função de Lennard-Jones automaticamente, foi definido unicamente d^* como parâmetro ajustável. Com este ajuste da distância d^* entre os robôs é possível que o enxame consiga produzir diferentes tamanhos do padrão de coesão, conseguindo se adaptar a diferentes cenários. O parâmetro ϵ foi ajustado em $\epsilon = 2,5$. Na fase de calibração dos robôs físicos, esta decisão permite especificar uma coesão no enxame que se mostrou adaptável as condições do ambiente, como explicado na Seção 5.2.1.

Cada medida da câmera $v_{i_{cam}}$ esta limitada no intervalo $0,05 \leq v_{i_{cam}} \leq 0,30 m$ para limitar o erro da medida em $4 cm$. O vetor r_{cam} inclui restrições de distância explícitas nas interações entre os robôs.

EvoSpace

Em EVOSPACE os controladores são especificados como na Equação (3.18). A rede tem 14 entradas e 4 saídas e é utilizado o modelo de referência da Tabela 6.1. A escolha dessa quantidade de entradas e de saídas foi feito para que a quantidade de parâmetros da rede neuronal se mantenha próxima a quantidade de parâmetros das redes neurais produzidas pelo método EVOSTICK.

Os vetores de direção do sensor de proximidade, luminosidade e posição relativa foram codificados de acordo a trabalhos realizadas anteriormente na bibliografia [18, 87]. As entradas foram redefinidas para incluir novas restrições de distâncias nas interações entre os robôs para atingir a coesão no enxame.

- 4 entradas do sensor de proximidade para cada projeção do vetor r_{prox} nos vetores $a_i = e^{j\pi \cdot i/4}$, $i = \{-3, -1, 1, 3\}$. Os vetores a_i estão representados em função de O_c . O

vetor r_{prox} foi anteriormente definido na Equação (3.14).

- 4 entradas do sensor de luminosidade para cada projeção do vetor r_{luz} nos vetores $a_i = e^{j\pi \cdot i/4}$, $i = \{-3, -1, 1, 3\}$. O vetor r_{luz} foi anteriormente definido na Equação (3.15).
- 1 entrada do sensor de chão gnd , com $gnd = \{0, 0; 0, 5; 1, 0\}$.
- 4 entradas do sensor de posição relativa para cada projeção do vetor r_{cam} nos vetores $a_i = e^{j\pi \cdot i/4}$, $i = \{-3, -1, 1, 3\}$. O vetor r_{cam} foi anteriormente definido na Equação (5.2).
- 1 entrada do sensor de proximidade relativa para a quantidade de robôs na vizinhança $n_{R\&B}$.
- 2 saídas que representam as velocidades das rodas, escaladas no intervalo $v_i \in \mathbb{R} : [-0,12; 0,12] m/s$, com $i \in \{l, r\}$ as velocidades das rodas esquerda e direita, respectivamente.
- 2 saídas que representam a ativação, ou desativação, simultânea dos três LEDs do e-puck, $led_i \in \mathbb{N} : [0, 1]$ com $i \in \{1, 2, 3\}$.

A rede neural é caracterizada por 60 parâmetros $x \in X$, $x \in \mathbb{R}$ a ser ajustados no intervalo $x : [-5, 5]$. Foi utilizada uma função logística sigmoide em cada saída da rede. As entradas colocam restrições explícitas de distância nas interações entre os robôs e dos robôs com o meio externo. Foram colocadas restrições de distâncias explícitas no vetor de posição relativa com robôs da vizinhança, e em menor grau no vetor de proximidade.

O simulador computacional

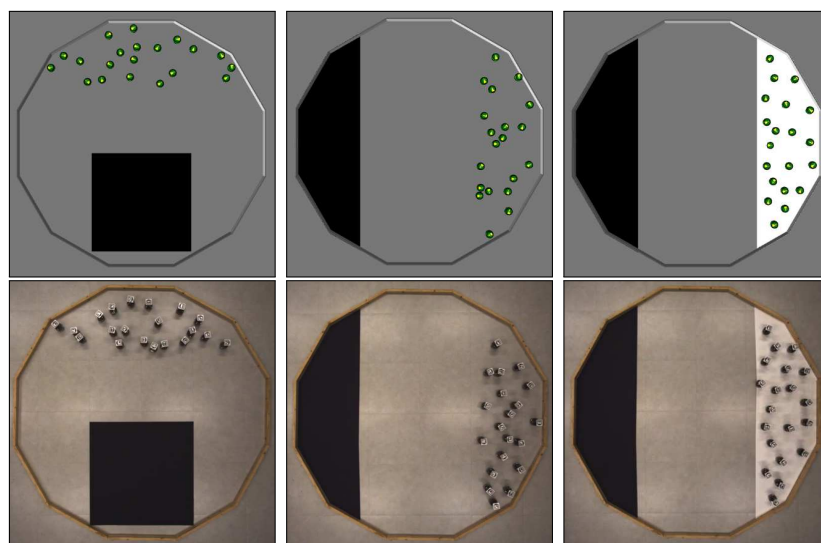
O simulador escolhido para a simulação dos robôs e o ambiente foi o simulador ARGoS (Seção 3.8.3). Como explicitado no Capítulo 5 a escolha do simulador se deve principalmente a que ele é apropriado para ser utilizado em enxames de robôs, por ser escalável na quantidade de robôs que podem ser simulados, flexível por ser um simulador fortemente modular, permite a automatização de experimentos e principalmente permite a execução de simulações exaustivas. Simulações exaustivas são necessárias para realizar o projeto automático de controladores para enxames de robôs no intuito de determinar o controlador do robô.

O algoritmo de otimização

O algoritmo de otimização permite explorar o conjunto de possíveis soluções (controladores) a um problema e decidir qual é a de melhor desempenho. Como são utilizados enxames de robôs para resolver o problema, o problema possui variabilidade no que diz respeito às ações de cada robô individual. Ante este fato, F-Race é apropriado para abordar este tipo de problemas, pois o algoritmo é inerentemente estocástico no que diz respeito a sua evolução e escolhas dos controladores [74]. A escolha do algoritmo permite uma comparação direta entre os métodos CHOCOLATE e MATE. Esses fatos justificam a escolha de F-Race no método MATE. Por sua vez, foi escolhido o algoritmo evolutivo do método EVOSTICK para o método EVOSPACE, pois é um algoritmo padrão utilizado como referencial de comparação na bibliografia e é amplamente utilizado nas abordagens evolutivas [18].

6.2 Metodologia

Figura 6.1: Imagem das arenas simuladas (imagens superiores) e físicas (imagens inferiores): *cobertura simples* (esquerda), *cobertura em rede* (centro) e *cobertura condicional* (esquerda). As imagens mostram também a pose inicial dos 20 robôs para cada missão.



Fonte: Elaborada pelo autor.

As três abordagens, MATE, CHOCOLATE e EVOSPACE, serão comparadas em três missões

que são condicionadas à capacidade dos robôs de se organizar no cenário. A Figura 6.1 apresenta as três arenas e os robôs inicialmente posicionados no começo do experimento. A arena é um cenário dodecagonal de $4,91 m^2$ limitada por paredes de madeira da mesma forma que o experimento anterior. É definido o mesmo sistema de coordenadas que o dos experimentos da Seção 4.3 (imagem inferior da Figura 4.3). O tempo para resolver as missões é de $T_m = 120 s$ para os métodos comparados. As missões se denominam *cobertura simples*, *cobertura em rede* e *cobertura condicional*.

Foram selecionadas tarefas em que se avalia a capacidade dos robôs para se distribuírem na arena. De igual forma que no Capítulo 4, foram escolhidas funções objetivo que se focam diretamente em avaliar o desenvolvimento dessas tarefas coletivas, independentemente do desempenho dos comportamentos individuais dos robôs [31]. Isto deixa graus de liberdade para o projeto automático decidir qual é a melhor configuração de módulos e parâmetros que permite desenvolver a tarefa. Maiores informações dos detalhes de cada missão se encontram na Seção 6.3.

A priori é esperado que tanto na primeira como na segunda missão, o módulo *formar* seja relevante diferentemente da terceira missão: A função objetivo da primeira missão coloca certo grau de pressão no que diz respeito a distância entre robôs e as condições do ambiente, tais como o perímetro da arena, não favorecem para que os robôs permaneçam na área. Na segunda missão a restrição de distância aumenta, é explícita na função objetivo e ainda as condições do ambiente favorecem a que os robôs permaneçam na área. Finalmente, a terceira missão possui uma função objetivo focada a resolver tarefas complexas com mudanças de cenário [31]. Por isto a restrição de distâncias é mais débil que nas anteriores missões. Então, na terceira missão não visualizamos *a priori* qual pode ser a forma de solução da missão.

No começo do experimento, os robôs são posicionados aleatoriamente numa região específica do cenário. O exame é composto de 20 robôs e-puck. Cada método de projeto é executado 10 vezes em cada tarefa, obtendo 10 controladores. Logo para cada tarefa, os 30 controladores obtidos (10 controladores por método) são avaliados em simulação e nos robôs físicos, uma única vez. São utilizados 200.000 experimentos para projetar cada controlador. No caso de EVOSPACE os 200.000 experimentos estão distribuídos da mesma forma como foi feito na Seção 4.3 para o Método EVOSTICK: 100 candidatos de controle aleatoriamente gerados e avaliados cada um 10 vezes em diferentes instâncias do cenário. Logo, com as

seguintes populações obtidas por elitismo e mutação é repetido o procedimento durante 150 gerações para finalmente ser realizado uma pós-avaliação dos 100 melhores candidatos de controle avaliado 500 vezes para obter o melhor controlador para essa missão específica. Os controladores foram sintetizados no cluster MAJORANA [124], como realizado com os controladores nos experimentos do Capítulo 4.

Para apresentar as gráficas de desempenho das abordagens, são utilizados diagramas de caixas. O diagrama de caixa neste contexto permite observar a mediana e quartis dos dados de desempenho de um controlador num cenário amostrado aleatoriamente do conjunto de cenários da missão específica. Cada amostra no diagrama de caixas é o resultado de desempenho de um controlador nesse cenário. São apresentados dois diagramas de caixa para cada abordagem, o diagrama de caixa cumprido faz referência aos dados de desempenho obtidos com os robôs físicos, os outros diagramas de caixa mais conciso são dados dos experimentos simulados. É realizado o teste estatístico de Wilcoxon pareado para comparar ambos os métodos em cada missão, a significância do teste é 95% [123]. A escolha do teste possui os mesmos argumentos que os apresentados na Seção 4.3.

6.3 Resultados experimentais

6.3.1 Cobertura simples

A missão tem o objetivo de que o enxame de robôs cobre a região preta da arena. Os robôs devem se posicionar para minimizar o quadrado da distância esperada entre um ponto aleatório na região preta e o robô mais próximo. A função objetivo que deve ser minimizada é

$$C(\theta) = E[d]^2, \quad (6.1)$$

em que $E[d]^2$ é o quadrado da distância esperada $E[d]$ entre um ponto escolhido aleatoriamente na região preta e o robô mais próximo nessa região no final do experimento em $T_m = 120$ s. A função objetivo deve ser minimizada. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

Algorithm 7 Cálculo da função objetivo para a missão **cobertura simples**

```

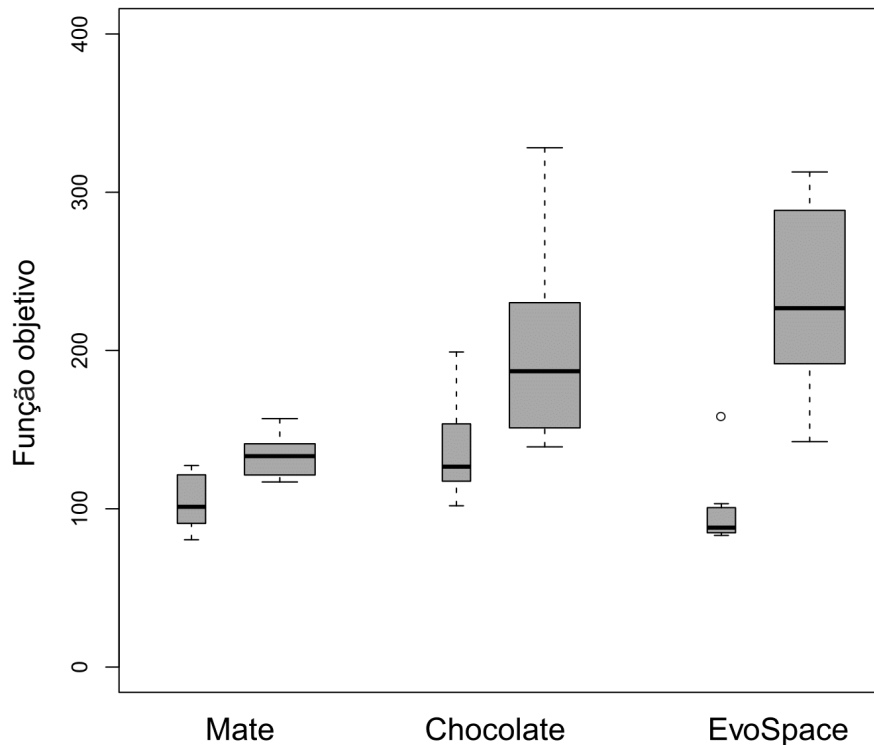
1:  $I_{ins} \leftarrow$  Instância de cenário com condições iniciais dos robôs definidas
2:  $\theta \leftarrow$  Controlador a avaliar
3:  $E[d] \leftarrow 0$ 
4:  $num_p \leftarrow 1000$ 
5:  $n_{robos} \leftarrow 20$ 
6:  $T_m \leftarrow 1200$ 
7: for  $t = 1, t \leq T_m, t++$  do
8:   if  $t = T_m$  then
9:     for  $l = 1, l \leq num_p, l++$  do
10:       $d_{min} \leftarrow 0,67 m$ 
11:      for  $i = 1, i \leq n_{robos}, i++$  do
12:        if robôi na area preta then
13:           $d \leftarrow dist(p_l, robô_i)$ 
14:          if  $d < d_{min}$  then
15:             $d_{min} \leftarrow d$ 
16:           $E[d] \leftarrow E[d] + d_{min}$ 
17:         $E[d] \leftarrow E[d]/num_p$ 
18:  $C(\theta) \leftarrow 10^4 E[d]^2$ 
19: return  $C(\theta)$ 

```

Na linha 1 são definidas as condições iniciais dos robôs e o cenário. Os robôs são posicionados uniformemente dentro do cenário em $x \leq -0,7 m$, como observado na Figura 6.1. Por sua vez, o cenário é representado como uma região dodecagonal que contém uma região quadrada preta centrada em $(0,6 m; 0,0 m)$ e possui $1 m^2$ de área. Esta missão é baseada na missão *cobertura e monitoramento de região*, mas sem o monitoramento do perímetro. O objetivo é cobrir a região preta, distribuindo os robôs da forma mais homogênea possível. Nesta missão um valor esperado grande da distância é penalizado com o quadrado na fórmula, fomentando soluções tendentes a agrupar robôs na região num padrão distribuído. O máximo valor permitido é quando um único robo se encontra na diagonal do quadrado, o qual é penalizado com um valor $d_{min} = 0,67 m$. No final do experimento é calculado $E[d]$ de acordo às linhas 9-17. Na linha 9 são escolhidos aleatoriamente 1000 pontos na região quadrada preta. Na linha 18 é finalizado o experimento e na linha 19 é realizada a devolução do valor da função objetivo em cm^2 .

Resultados quantitativos

Figura 6.2: Diagrama de caixas apresentando o desempenho dos controladores para os métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão **Cobertura simples**, quanto menor a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Observando o desempenho quantitativo da missão na Figura 6.2, pode ser concluído que de acordo com o teste de Wilcoxon 95 % o desempenho dos controladores do método MATE são superiores a os controladores do método CHOCOLATE tanto na simulação como nos robôs físicos. Na simulação, analisando os dados em termos de distância entre o robô mais próximo a pontos aleatórios na região, pode ser dito que a diferença entre medianas de desempenho é de 5,6 cm e o intervalo de confiança sinaliza que no pior dos casos o desempenho dos controladores do método MATE é superior que os do método CHOCOLATE em aproximadamente 3,8 cm. Nos robôs físicos, esta diferença aumenta, existindo uma diferença de medianas de 7,6 cm com o intervalo de confiança sinalizando que no pior dos casos o desempenho dos controladores do método MATE é superior que os do método

CHOCOLATE em aproximadamente 5,0 *cm*. Pode ser observada uma queda no desempenho em CHOCOLATE entre a simulação e os robôs físicos devido a que na simulação os robôs penetram a região e se detêm mais perto do centro da região. Porém, nos robôs físicos, eles se detêm mais próximo da fronteira, não cobrindo corretamente o centro da região. Como nos controladores do método MATE os robôs não se detêm, eles conseguem ajustar a distribuição dentro da região no tempo de execução do experimento.

Da observação dos diagramas de caixas da Figura 6.2, não existe evidencia estatística para concluir que o método MATE possui melhor desempenho que o método EVOSPACE na simulação. Porém, nos robôs físicos o desempenho do método MATE foi significativamente melhor do que o método EVOSPACE em 9,8 *cm* de diferença limitado no pior dos casos a 7,7 *cm*, aproximadamente.

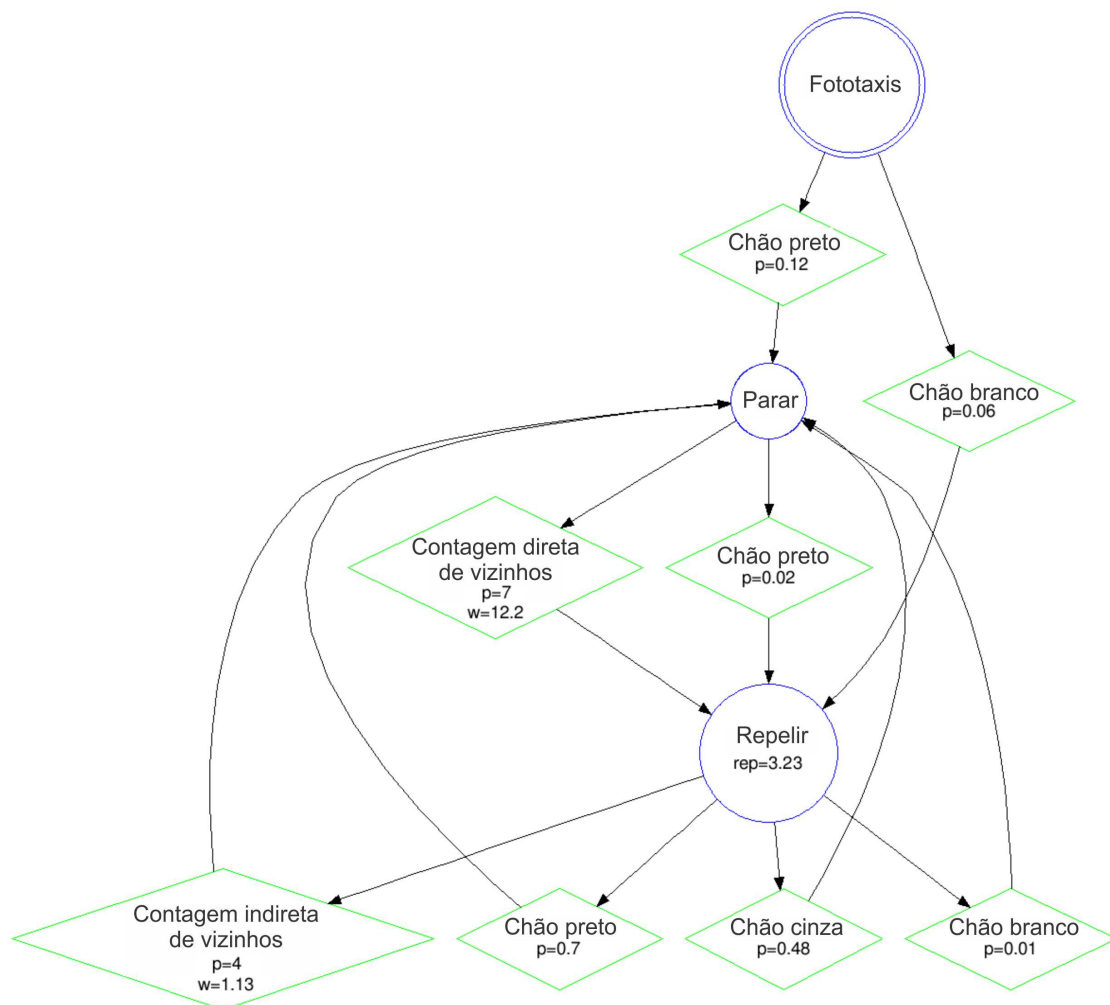
Na comparação dos resultados da simulação entre os métodos CHOCOLATE e EVOSPACE, CHOCOLATE possui pior desempenho que EVOSPACE em 6,3 *cm* limitado a 5,2 *cm*. Porém, nos robôs reais o desempenho de ambos é comparável.

Dos resultados observados pode ser concluído que o algoritmo evolutivo possui diferenças significantes no que diz respeito aos resultados simulados e com robôs físicos. Respeito ao método MATE, foi menos afetado na transferência de controladores aos robôs reais que no método CHOCOLATE.

Resultados qualitativos

Observando o comportamento emergente produzido tanto na simulação quanto nos robôs físicos, no método AUTOMODE-CHOCOLATE os robôs exploram a arena e quando encontram a região quadrada, eles ingressam e, normalmente, permanecem parados até o final do experimento. O método é incapaz de posicionar robôs uniformemente na região, deixando partes da região descobertas e regiões com excesso de agrupamento de robôs. Geralmente, a região central do local não é coberta uniformemente e os robôs são posicionados perto da borda da região. No final do experimento, mais de 16 robôs são observados em média na região. Na Figura 6.3 se observa um dos 10 controladores produzidos pelo método CHOCOLATE. Este controlador é considerado representativo, pois o comportamento é similar entre os 10 controladores projetados.

Figura 6.3: Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão **cobertura simples**.



Fonte: Elaborada pelo autor.

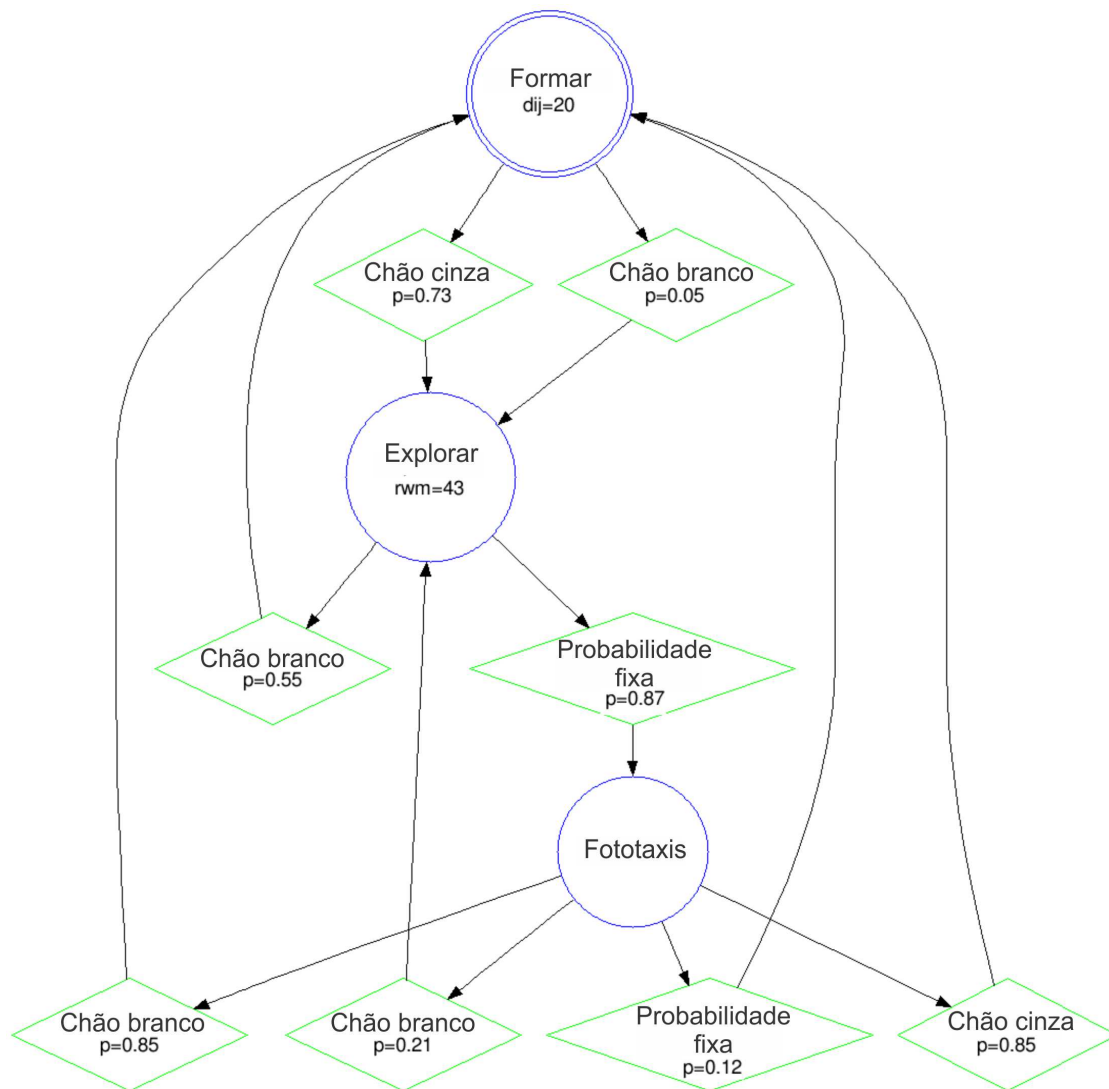
Nos controladores projetados, e em particular o da Figura 6.3, a exploração é normalmente realizada por intermédio do comportamento *fototaxis*. Como a luminosidade do ambiente é filtrada, e, não foi colocada a fonte luminosa externa neste experimento, produz que o comportamento *fototaxis* se comporte de forma similar ao comportamento *explorar*. Uma vez detectado o chão preto, a transição para o comportamento *parar* acontece com probabilidade 0,12. Logo, acontece a transição para o comportamento repulsão com probabilidade 0,02. Como o robô se encontra na região preta ele eventualmente volta se deter. Se o robô encontra a região cinza ele para na fronteira com probabilidade 0,48. Ainda o robô pode ser repellido por outros robôs se o robô possui aproximadamente 7 vizinhos.

Por outro lado, no método MATE, os robôs exploram a arena combinando os módulos de *explorar* e *formar* até encontrarem a área de interesse, e se distribuem uniformemente nessa área produto das interações do comportamento *formar*. Robôs que eventualmente deixam a região preta são atraídos para ela usando comportamentos individuais, como *atrair*, *formar* ou *explorar*. Em seguida, a quantidade de robôs distribuídos no quadrado preto é suficiente para cobrir todo o local, sendo mais difícil que novos robôs possam acessar à região. No final do experimento, entre 13 e 16 robôs uniformemente distribuídos são observados no local. Na Figura 6.4 se observa um dos 10 controladores representativos produzidos pelo método MATE. O projeto automático definiu que o módulo *formar* seja instanciado pelo menos uma vez em cada um dos 10 controladores. A distância ajustada para esta primeira instanciação do controlador foi de $d^* = 0.2 m$. Por sua vez, em 5 dos 10 controladores o controlador foi instanciado duas vezes. Na segunda vez o ajuste foi definido a distâncias menores, entre $0,09 \leq d^* \leq 0.13 m$.

No controlador específico da Figura 6.4, os robôs começam explorar o cenário combinando sequencialmente os comportamentos *formar*, *explorar* e *phototaxis*. Eventualmente, quando os robôs ingressem na região preta do cenário, eles mantêm a coesão. Se eles saírem da região, com probabilidade 0,73, os robôs voltam a explorar o cenário até encontrar novamente a região preta. Como mencionado, nos outros controladores projetados que utilizam o método MATE, o comportamento *atrair* é utilizado quando o robô sai da região preta, permitindo o robô reingressar na região.

No método EVOSPACE os robôs exploram a arena até encontrar a região de interesse, uma vez encontrada ingressam nela e começam rotacionar no seu próprio eixo. Dependendo da instância de controle, esta rotação pode ser mais ampla. Quanto mais ampla essa rotação existe mais perca de robôs da região. No final do experimento são posicionados entre 9 e 15 robôs como extremos com média de 10 robôs nos experimentos com robôs reais. A saída dos LEDs produz que os LEDs sejam ativados a maior parte do tempo, a combinação das entradas com as saídas é ponderada de tal forma que não se observa uma distribuição uniforme na região.

Figura 6.4: Controlador representativo produzido pelo método AUTOMODE-MATE para a missão **cobertura simples**.



Fonte: Elaborada pelo autor.

Pode-se observar que a distribuição na região é notoriamente mais eficiente no método MATE para cobrir a região que os outros dois métodos, tanto na simulação como também nos robôs físicos. Como esperado nesta missão, a quantidade de robôs no local não é tão relevante quanto sua distribuição na região de interesse. Por esta razão, pode-se observar que, embora o método MATE coloque em média menos robôs na região comparado com os outros métodos, o método Mate cobre a área uniformemente e consegue ajustar da forma que ela é coberta no tempo de execução.

6.3.2 Cobertura em rede

A missão tem o objetivo de que o enxame de robôs maximize a cobertura de uma região específica com a maior rede conectada possível de robôs do enxame. A arena é cinza e possui a região a cobrir da cor preta de aproximadamente $1 m^2$ de área. A função objetivo que deve ser maximizada é

$$C(\theta) = \sum_{i=0}^{T_m} \frac{A_{N_B}(t)}{A_{total}}, \quad (6.2)$$

em que N_B é definida como a rede com maior quantidade de robôs na região preta, $A_{N_B}(t)$ é a região preta coberta por N_B no instante de tempo t e A_{total} é a área total da região preta. N_B é composta por robôs com conectividade de $d_r \leq 0,30 m$ e o alcance da cobertura é de $d_c \leq 0,15 m$, permitindo redes com cobertura sem solapamento. A função objetivo é avaliada a cada $0,1 s$ e o experimento tem a duração de $T_m = 120 s$. A função objetivo mede a fração da região preta coberta. Assim, quanto maior o valor, melhor é feita a cobertura pelo enxame. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

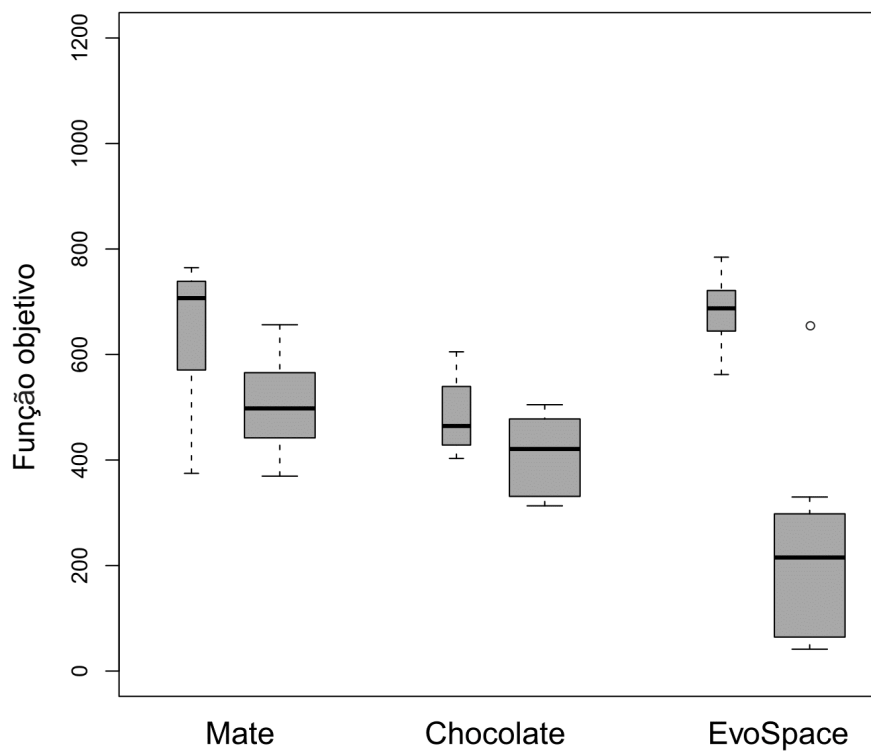
Algorithm 8 Cálculo da função objetivo para a missão **cobertura em rede**

- 1: $I_{n_{ins}} \leftarrow$ Instância de cenário com condições iniciais dos robôs definidas
 - 2: $\theta \leftarrow$ Controlador a avaliar
 - 3: $A_{N_B} \leftarrow 0$
 - 4: $A_{total} \leftarrow 1 m^2$
 - 5: $d_r = 0,3 m, d_c = 0,15 m$
 - 6: $n_{robos} \leftarrow 20$
 - 7: $T_m \leftarrow 1200$
 - 8: **for** $t = 1, t \leq T_m, t++$ **do**
 - 9: $r_p \leftarrow$ *ObterRobôsNaRegiãoPreta(robôs)*
 - 10: $N \leftarrow$ *DefinirRedesRegiãoPreta(r_p, d_r)*
 - 11: $N_B \leftarrow$ *DefinirRedeComMaiorQtRobôs(N)*
 - 12: $A_{N_B} \leftarrow A_{N_B} +$ *CalcularAreaCoberta $N_B(N_B, d_c)$*
 - 13: $C(\theta) \leftarrow A_{N_B} / (A_{total} T_m)$
 - 14: **return** $C(\theta)$
-

Na linha 1 são definidas as condições iniciais dos robôs e o cenário. Os robôs são posicionados uniformemente dentro do cenário em $y \geq 0,7 m$, como observado na Figura 6.1. Por sua vez, o cenário é representado como uma região dodecagonal que contém a região preta de $1 m^2$ de área. Para cada instante de tempo são determinados os robôs na região

preta (linha 9) e todas as redes formadas pelos robôs vizinhos nessa região preta (linha 10). Depois disso é escolhida a rede N_B com maior quantidade de robôs (linha 11) e logo é realizada o calculo da area que é coberta por essa rede N_B , acrescentando o valor ao valor acumulado anterior (linha 12). Finalmente, na linha 13 é finalizado o experimento com a devolução do valor da função objetivo na linha 14.

Figura 6.5: Diagrama de caixas apresentando o desempenho dos controladores para os métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão **cobertura em rede**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Resultados quantitativos

Observando o desempenho quantitativo da missão na Figura 6.5, pode ser concluído que de acordo com o teste de Wilcoxon 95 %, o desempenho do método MATE é superior ao método CHOCOLATE tanto na simulação quanto nos robôs físicos. Na simulação, a diferença de desempenhos é de 181,9 unidades, o que equivale a dizer que existe uma diferença de 18 s

na qual os controladores projetados com o método CHOCOLATE não estão cobrindo a região preta e os controladores do método MATE estão cobrindo ela totalmente. O intervalo de confiança limita este valor para 80,7 unidades (8 s). Observando as medianas de desempenho, o método MATE cobre aproximadamente 50% a mais da região do que cobre o método CHOCOLATE. Nos robôs físicos a diferença se reduz para 91,8 unidades limitado a 34,2 unidades como o pior caso. Pode ser observado uma queda no desempenho dos controladores do método MATE entre a simulação e os robôs físicos devido à dificuldade de manter a conectividade na fronteira da região nos robôs físicos, este efeito não se observa na simulação o que mostra que as interações entre robôs foram modeladas com uma conectividade mais rígida do que na realidade se apresentam nos robôs físicos.

Observando o desempenho dos controladores nos resultados simulados entre os métodos MATE e EVOSPACE, pode ser concluído que não existe diferença significativa entre ambos métodos. Porém, nos robôs físicos o desempenho do método MATE foi significativamente superior ao método EVOSPACE, com uma diferença de mediana de 279 unidades limitado no pior dos casos a 110,7 unidades.

Na simulação, os controladores do método CHOCOLATE possuem pior desempenho que os controladores do método EVOSPACE em com uma diferença de medianas de 215,9 unidades limitado no pior dos casos a 128,7 unidades. Nos resultados com robôs reais o desempenho dos controladores do método CHOCOLATE é superior aos controladores do método EVOSPACE em 210,4 unidades limitado pelo intervalo de confiança a 68,8 unidades.

A respeito do método EVOSPACE, pode ser observado que os desempenhos dos controladores foram afetados em grande medida na transferência do controlador da simulação para os robôs físicos. A respeito dos resultados dos métodos MATE e CHOCOLATE, foram menos afetados nessa transferência. A seguir será explicada essa diferença na análise do comportamento qualitativo.

Resultados qualitativos

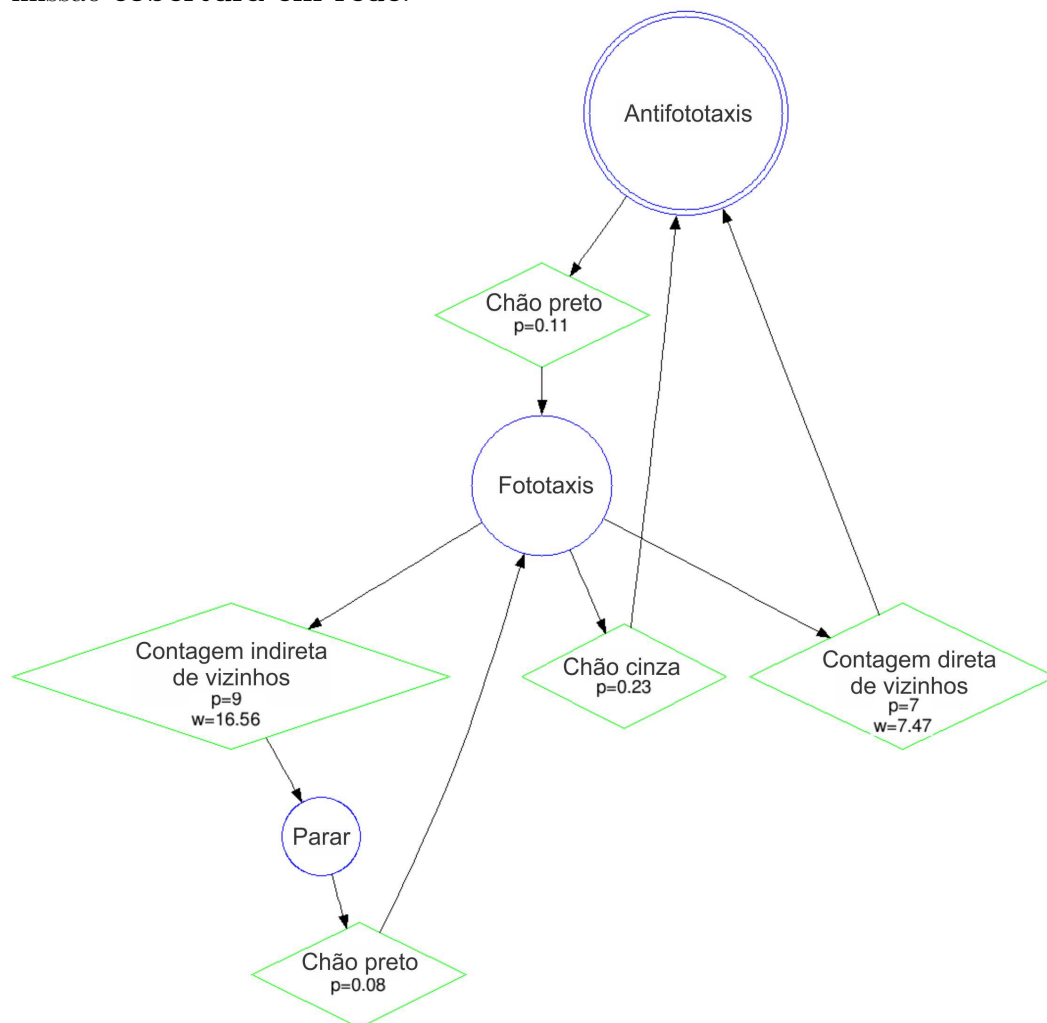
No método AUTOMODE-CHOCOLATE, os robôs começam a explorar a arena e, quando entram na região preta, os robôs continuam explorando, mas também ficam parados permanecendo nesse local, misturando ambos os comportamentos. Eventualmente, quando

os robôs estão dentro da região e encontram a borda entre as regiões cinza e preta, os robôs param definitivamente. Isso produz um comportamento emergente de barreira no perímetro da área preta. Essa barreira faz com que outros robôs dentro da área não possam escapar, mas ao mesmo tempo tem a desvantagem de que robôs fora da área não consigam entrar facilmente. Ao mesmo tempo, essa barreira geralmente forma uma rede conectada com robôs dentro da área, mas é incapaz de posicionar os robôs uniformemente na área, demonstrando que essa tarefa não é resolvida de forma eficiente. No final do experimento, essa barreira é composta por mais da metade dos robôs nas 10 instâncias executadas. Os resultados mostram um comportamento de enxame similar com os 10 projetos de controlador. Na Figura 6.6 se observa um dos 10 controladores representativos produzidos pelo método CHOCOLATE. Neste controlador específico, a exploração é realizada pelos comportamentos *fototaxis* e *anti-fototaxis*, tendo em conta a não existência da fonte luminosa externa neste experimento.

No método MATE, os robôs exploram a arena e fazem a transição para o comportamento *formar* quando os robôs detectam a região preta. Dependendo do controlador de software obtido no projeto, essa transição é feita mais lentamente para permitir que mais robôs entrem na área. As distâncias que os robôs devem manter entre si foram ajustadas no processo de otimização no valor máximo para maximizar a rede nessa região. Assim que mais robôs entrarem no local, eles começam a manter a coesão e cobrir a área uniformemente. Observa-se que quando um novo robô entra na área, ele produz uma breve quebra na estrutura, até que o robô faz a transição para o comportamento *formar*. Dependendo da instância, se os robôs saírem da área, eles continuarão mantendo a coesão fora do local ou explorarão para encontrar novamente a área. Observa-se que os robôs têm mais dificuldades para ingressar na área quando a quantidade de robôs aumenta. No equilíbrio, a quantidade de robôs se mantém estável dentro da região. Pode-se observar que a distribuição na região de interesse é significativamente melhor do que a distribuição obtida com os controladores produzidos pelo método AUTOMODE-CHOCOLATE, permitindo uma cobertura uniforme da região.

Na Figura 6.7 se observa um dos 10 controladores representativos produzidos pelo método MATE. Neste controlador específico, a exploração é realizada pelo comportamento *fototaxis*. A transição para o comportamento *formar* acontece quando o robô detecta o chão preto com probabilidade 0,09. Se o robô sai da região preta, começa explorar novamente pela detecção

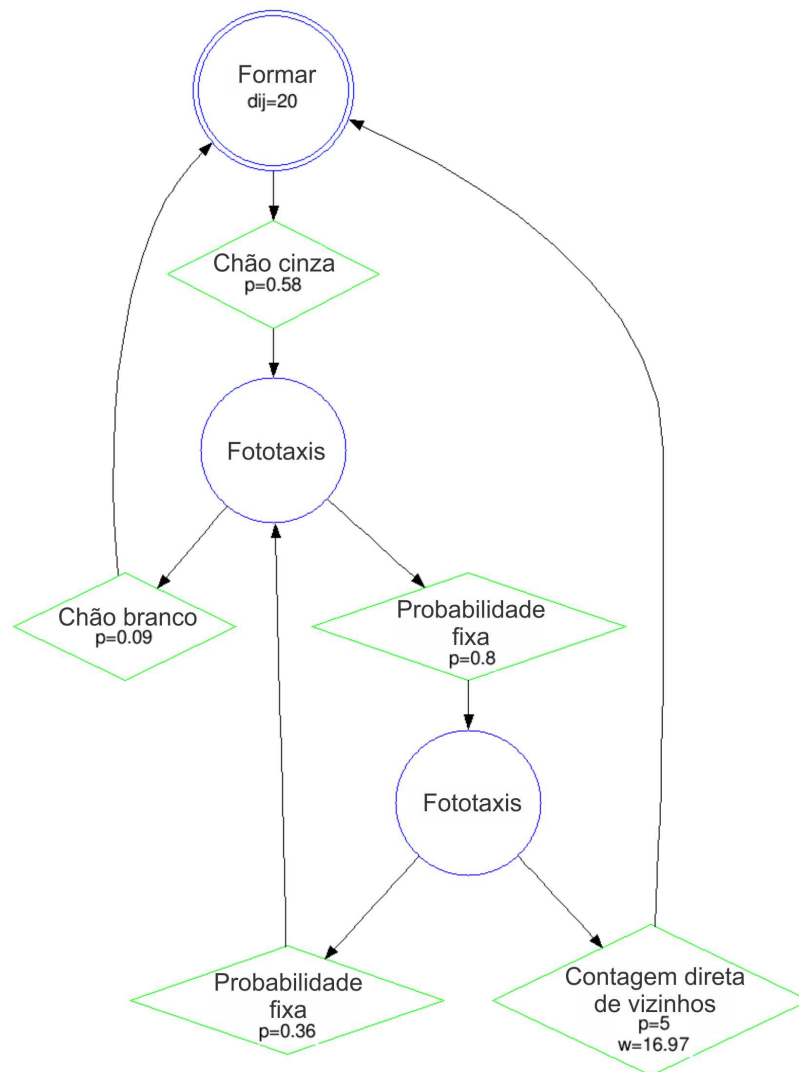
Figura 6.6: Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão **cobertura em rede**.



Fonte: Elaborada pelo autor.

do chão cinza. O projeto automático definiu que o módulo *formar* seja instanciado pelo menos uma vez em cada um dos 10 controladores. A distância ajustada para esta primeira instanciação do controlador foi de $d^* = 0.2m$. Por sua vez, em 3 dos 10 controladores o controlador foi instanciado duas vezes. Na segunda instanciação o ajuste foi definido a distâncias menores, entre $0,08 \leq d^* \leq 0.15m$. O ajuste na distância máxima permite que a cobertura total da região seja realizada com menos quantidade de robôs que a totalidade do enxame, permitindo redundância na cobertura com os robôs restantes, caso for necessário.

Figura 6.7: Controlador representativo produzido pelo método AUTOMODE-MATE para a missão **cobertura em rede**.



Fonte: Elaborada pelo autor.

O método EVOSPACE possui comportamentos diferentes entre os 10 controladores: em alguns deles os robôs ficam realizando agregação no intuito de manter a coesão, mas é feito fora da região de interesse e poucos robôs conseguem cobrir a área. Em outros controladores os robôs exploram fora da região e quando ingressam na região preta começam rotacionar no seu próprio eixo ou circularmente não mantendo uma boa conectividade. Em outros dos controladores os robôs tentam atingir a região por intermédio de um comportamento emergente de seguidor de parede, que no simulador esse comportamento é bem sucedido, mas nos robôs físicos eles ficam colidindo com a parede. Por sua vez, num dos controladores

os LEDs dos robôs permaneceram desligados em todo o experimento, não explorando as entradas do vetor de coesão entre robôs. Entre todos os controladores, consegue se observar que as características desejadas estão distribuídas e nenhum controlador contém todas as características positivas para desenvolver uma coesão como a desejada no experimento.

6.3.3 Cobertura condicional

A missão tem o objetivo de que o enxame de robôs maximize a cobertura de uma região aleatoriamente escolhida da arena dentre duas regiões possíveis. A região a ser coberta é condicionada pela região na qual os robôs são originalmente posicionados no começo do experimento. Assim, se no começo do experimento os robôs são posicionados na região branca (preta), os robôs deveriam cobrir a região preta (branca), respectivamente. A função objetivo que deve ser maximizada é

$$C(\theta) = \sum_{i=0}^{T_m} \frac{A_m(t)}{A_{total}}, \quad (6.3)$$

em que $A_m(t)$ é a área coberta por m robôs na região a cobrir e A_{total} é a área total a cobrir. O alcance da cobertura de cada robô é de $d_c \leq 0,15 m$. A função objetivo é avaliada a cada $0,1 s$ e o experimento tem a duração de $T_m = 120 s$. O pseudocódigo para o cálculo da função objetivo $C(\theta)$ é apresentado a seguir:

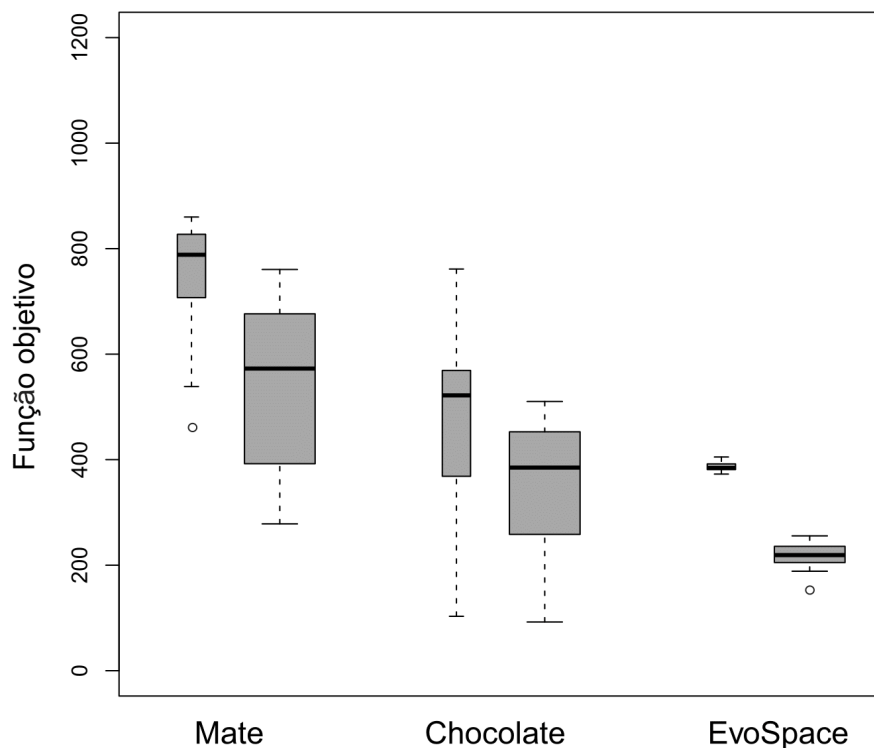
Algorithm 9 Cálculo da função objetivo para a missão **cobertura condicional**

- 1: $I_{n_{ins}} \leftarrow$ Instância de cenário com condições iniciais dos robôs definidas
 - 2: $\theta \leftarrow$ Controlador a avaliar
 - 3: $A_m \leftarrow 0$
 - 4: $A_{total} \leftarrow 1 m^2$
 - 5: $d_c \leftarrow 0,15 m$
 - 6: $n_{robos} \leftarrow 20$
 - 7: $T_m \leftarrow 1200$
 - 8: **if** *robots inicializados em região preta* **then**
 - 9: região \leftarrow *branca*
 - 10: **else**
 - 11: região \leftarrow *preta*
 - 12: **for** $t = 1, t \leq T_m, t++$ **do**
 - 13: $r_p \leftarrow$ *ObterRobôsNaRegiao(região, robôs)*
 - 14: $A_m \leftarrow$ *CalcularAreaCoberta(r_p, d_c)*
 - 15: $C(\theta) \leftarrow A_m / (A_{total} T_m)$
 - 16: **return** $C(\theta)$
-

Na linha 1 é definida as condições iniciais dos robôs e a definição do cenário. O cenário é representado como um cenário dodecagonal cinza com duas regiões opostas branca e preta de $1 m^2$ de área cada uma, localizadas em $y \leq -0,6 m$ e $y \geq 0,6 m$ limitadas pelas paredes da arena, respectivamente. Nas linhas 8-11 é definida que região vai ser coberta, em função da condição inicial dos robôs definida na linha 1. A cada $0,1 s$ são obtidos os robôs r_p na area a ser coberta (linha 13). Logo é realizado o cálculo da area coberta, acrescentando o valor ao valor acumulado anterior (linha 14). Finalmente, na linha 15 é finalizado o experimento com a devolução do valor da função objetivo na linha 16.

Resultados quantitativos

Figura 6.8: Diagrama de caixas apresentando o desempenho para os controladores dos métodos AUTOMODE-CHOCOLATE, AUTOMODE-MATE e EVOSPACE na missão **Cobertura em rede**, quanto maior a função objetivo melhor desempenho do método.



Fonte: Elaborada pelo autor.

Na Figura 6.8 são apresentados os resultados quantitativos do experimento, tanto no

simulador como nos robôs físicos. Como nas anteriores missões, cada método sintetiza 10 controladores. A avaliação do desempenho é realizada utilizando 5 (5) controladores com a condição inicial dos robôs ser posicionados na região branca (preta) e avaliando como cobrem a região preta (branca), respectivamente. De acordo com o teste de Wilcoxon 95 %, o desempenho do método MATE é superior ao método CHOCOLATE tanto nos robôs reais como na simulação. Na simulação a diferença de medianas é de 251,3 unidades limitada no pior dos casos a 96,8 unidades. Em termos de tempo, pode ser equivalente a uma diferença de 25 s em que o método MATE teria equivalentemente uma cobertura total da área e o método CHOCOLATE teria cobertura de área nula. Nos robôs físicos, a diferença cai para 207,1 unidades limitado em no mínimo 65,9 unidades. Em termos de área o método MATE cobre um 45 % a mais que o método CHOCOLATE na simulação e um 50 % a mais nos robôs físicos. Existe uma queda de desempenho entre a simulação e os robôs físicos e da mesma forma que na missão anterior, esta queda de desempenho é causada por falta de conectividade perto da borda que separa a região de cobertura do resto da arena.

Da observação dos resultados no diagrama de caixas, pode ser concluído que os controladores do método MATE possuem melhor desempenho que os controladores do método EVOSPACE tanto nos robôs reais, como nos robôs simulados. Na simulação foi reportada uma diferença de medianas de 371,9 unidades limitado no pior dos casos em 253,2 unidades. Nos robôs reais a diferença é de 335,5 unidades limitado em 229,1 unidades.

Observando o desempenho entre os métodos CHOCOLATE e EVOSPACE, pode ser concluído que na simulação não existem diferenças significativas. Porém, nos robôs físicos o desempenho do método CHOCOLATE é superior em desempenho ao método EVOSPACE em 151,8 unidades, limitado em 58,5 unidades.

Resultados qualitativos

No método MATE, quando no experimento é fazer cobertura de área na região branca, os robôs físicos parecem não conseguir o objetivo e dividem esforços por cobrir as duas áreas simultaneamente, ou inclusive mais a região preta. Somente em duas instâncias os robôs reais se conseguiram o objetivo de cobrir a região branca com a maioria dos robôs das cinco instâncias. Nos experimentos simulados se observa o mesmo comportamento,

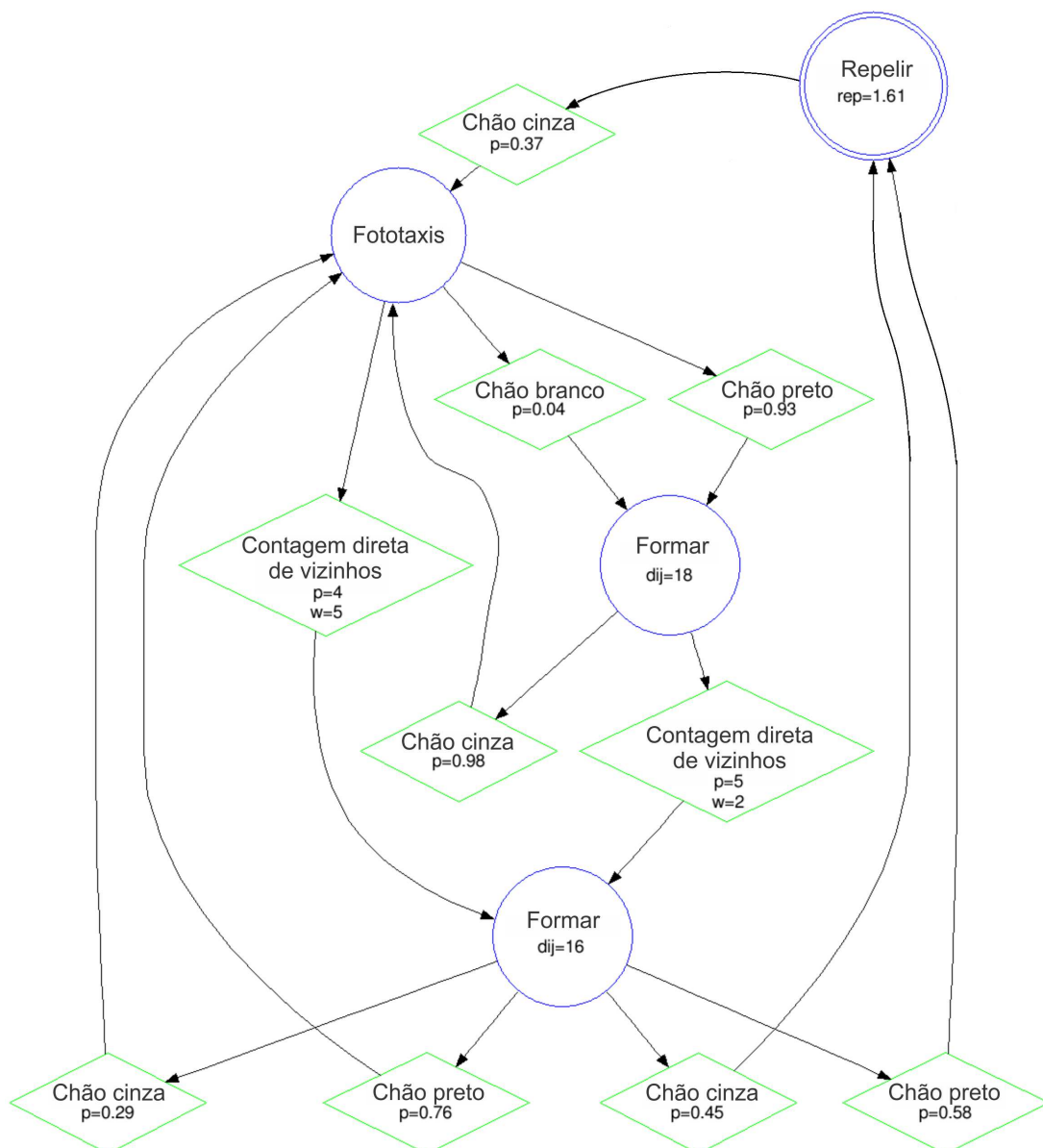
porém os robôs conseguem se manter distribuídos por mais tempo. Nestas cinco instâncias o comportamento *formar* é utilizado, fora da área de cobertura de forma intermitente e uma vez dentro, de forma fixa. Nas outras cinco instâncias que os robôs devem cobrir a região preta, em quatro delas os robôs conseguem se distribuir uniformemente, somente alguns robôs realizam cobertura da região branca. Em uma das instâncias o projeto definiu que o comportamento *formar* não era útil, os robôs começaram na região branca e muitos deles permaneceram por muito tempo nela, uma vez fora da região e em contato com a região oposta preta, os robôs possuem um comportamento de se deter e se movimentar, tentando permanecer na região preta. Neste experimento, os robôs tentaram cobrir as duas regiões, indo da região branca para a preta. Então, foram observados desempenhos disparez dependendo da posição inicial dos robôs, alguns dos controladores se especializaram em cobrir um tipo de região particular com bons desempenhos e outros se especializaram para ter desempenhos regulares, mas cobrindo as duas áreas. Na Figura 6.9 se observa um dos 10 controladores do método MATE, que consegue cobrir as duas regiões. O projeto automático definiu que o módulo *formar* seja instanciado pelo menos uma vez em 7 controladores com $d^* = 0,2 m$, duas vezes em 2 controladores ($0,09 m \leq d^* \leq 0,19 m$) e nenhuma vez num controlador. Estes resultados mostram a variabilidade dos controladores produzidos e da dificuldade da produção de controladores para esta tarefa coletiva particular.

No método CHOCOLATE, da mesma forma que no método MATE, consegue resolver uma única condição das duas possíveis dependendo das posição inicial dos robôs. Em alguns cenários que era para cobrir unicamente a região branca, os robôs cobriram a região preta. Em outras situações os robôs tentavam cobrir as duas simultaneamente, com mecanismos de detenção total, parando e explorando ou de forma mais dinâmica, por exemplo, intercambiando entre regiões. Na região cinza, os robôs atravessam ela rapidamente para chegar região oposta. Em alguns dos experimentos quando inicializado, os robôs começam sair da região paulatinamente. Normalmente, no final do experimento, ambas as regiões possuem aproximadamente a mesma quantidade de robôs. Os resultados desde o ponto de vista do comportamento não são satisfatórios, pois os robôs não cobrem as áreas de interesse completamente e por sua vez, existe excesso de robôs em regiões não interessantes para o experimento.

Na Figura 6.10 se observa um dos 10 controladores representativos produzidos pelo método

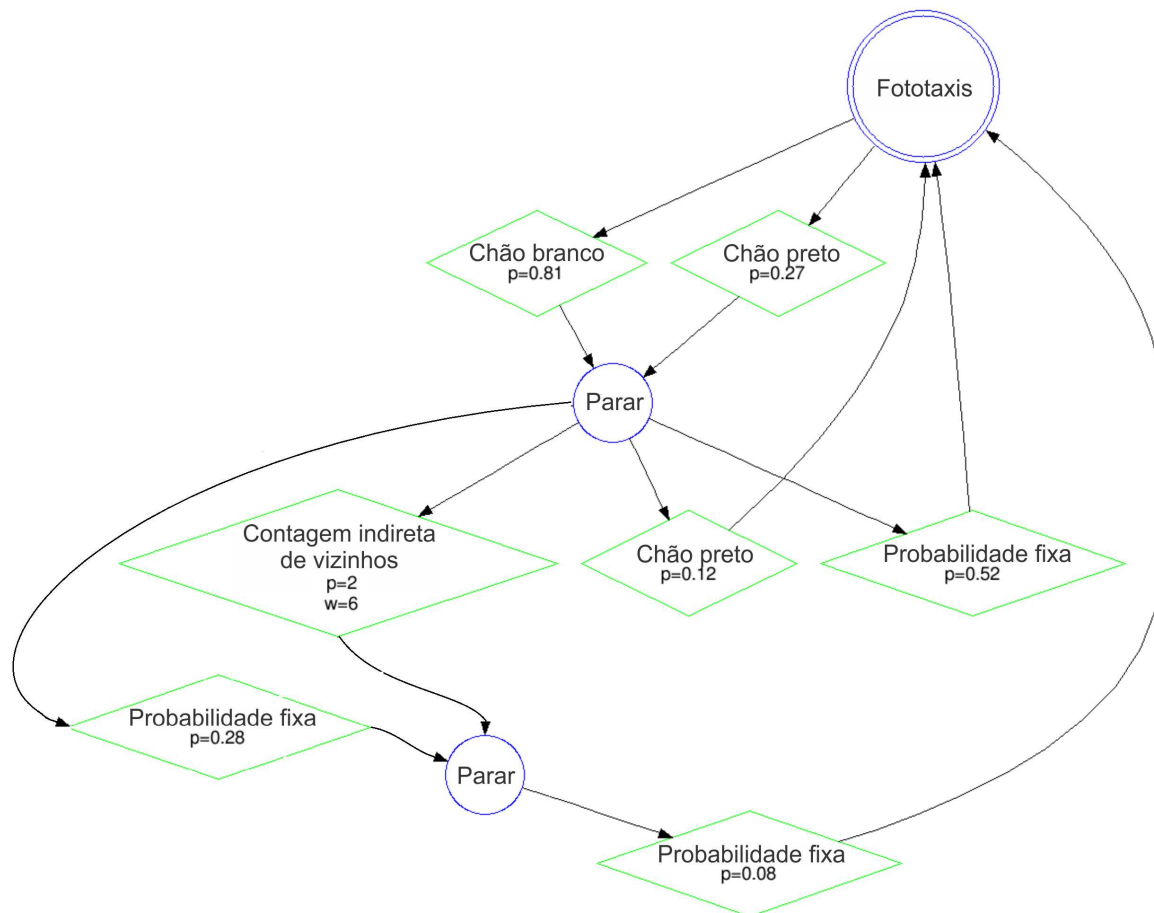
CHOCOLATE. Neste controlador específico, a exploração é realizada pelo comportamento *fototaxis*. Como observado na figura, quando é detectado o chão branco ou preto o robô realiza a transição para o comportamento *parar*. Observa-se que em ambas as regiões, este comportamento se alterna com o comportamento *fototaxis*, produzindo um movimento intermitente nos robôs.

Figura 6.9: Controlador representativo produzido pelo método AUTOMODE-MATE para a missão **cobertura condicional**.



Fonte: Elaborada pelo autor.

Figura 6.10: Controlador representativo produzido pelo método AUTOMODE-CHOCOLATE para a missão **cobertura condicional**.



Fonte: Elaborada pelo autor.

No projeto dos controladores do método EVOSPACE, o algoritmo evolutivo encontrou que as soluções de seguidor de perímetro da arena são as melhores soluções para cobrir as duas regiões simultaneamente: a cada intervalo de tempo os robôs se encontram divididos entre as áreas de interesse e vão se deslocando, seguindo o perímetro e a outros robôs. Os robôs vão realizando um agregado que se desloca conjuntamente por toda a arena. No entanto, nos robôs físicos esta solução não se observa na maioria dos experimentos. Os resultados tanto no simulador como nos robôs físicos são regulares a pobres. Os robôs não cobrem a região de interesse como um conjunto.

Pode ser observado que das três missões, a terceira foi a mais complexa desde o ponto de vista da solução, pois o cenário é variável no que diz respeito à região onde os robôs são posicionados inicialmente. Esta cobertura condicionada à região de posicionamento inicial

dos robôs dificulta a otimização para achar a solução que aborde os dois tipos de cenários.

6.4 Considerações finais

Neste capítulo se abordou a inclusão de restrições de distâncias em dois métodos de projeto automático de controladores para enxames de robôs. Esta inclusão das restrições permite abordar o problema de falta de coesão dos controladores projetados com os métodos AUTOMODE e EVOSTICK. Estas deficiências tanto no método AUTOMODE como no método EVOSTICK foram observadas no capítulo 4 e se abordaram no presente capítulo.

Foram propostos dos métodos de projeto automático para enxames de robôs: MATE e EVOSPACE. Cada um deles estende os métodos AUTOMODE e EVOSTICK, respectivamente.

MATE é uma extensão do método AUTOMODE-CHOCOLATE. O método MATE adiciona um novo comportamento que permite ajustar as distâncias entre vizinhos para se posicionar no ambiente. A distância de interação entre os robôs é ajustada pelo projeto automático para adequar as interações de acordo com a tarefa a realizar. O método MATE inclui a câmera omnidirecional do robô e-puck para a medição das distâncias dos robôs.

O método EVOSPACE, por sua vez, estende as capacidades do método EVOSTICK modificando as entradas e saídas da rede neuronal, incluindo restrições de distâncias explícitas que permitem, da mesma forma que o método MATE, incluir restrições de distâncias nos controladores produzidos.

Tanto o método MATE como o método EVOSPACE foram especializados para conter essas características acima citadas. A especialização foi realizada pensando no futuro conjunto de tarefas a ser abordados, mas não tendo em vista uma tarefa particular, para que os métodos consigam ser gerais no que diz respeito as tarefas que podem ser resolvidas com eles. Na especialização foi colocado a restrição de distâncias em cada método, para permitir a coesão dos robôs quando a tarefa requerer.

Primeiramente foi estabelecida a especialização definindo a *Plataforma robótica*, o *espaço de busca do controlador*, o *simulador computacional* e o *algoritmo de otimização*. As *condições experimentais* e a *função objetivo* foram definidas posteriormente para atacar as tarefas específicas deste capítulo. Assim foram definidas missões em que o enxame devia se organizar com coesão no cenário de trabalho. Foram definidas três missões de uma mesma classe de

tarefas para abordar cobertura de regiões.

Comparando os controladores dos métodos MATE e CHOCOLATE nas três missões apresentadas, o método MATE possui melhor desempenho que o método CHOCOLATE nas três missões tanto nos robôs físicos quanto na simulação. Comparando o método MATE com o método EVOSPACE, na primeira e segunda missão possuem desempenhos comparáveis. Na terceira missão, o método MATE possui melhor desempenho que o método EVOSPACE. A respeito aos robôs físicos, o método MATE é significativamente superior que o método EVOSPACE nas três missões. Comparando o método CHOCOLATE com o EVOSPACE nos robôs físicos, o método CHOCOLATE apresentou melhores desempenhos em duas das três tarefas. Porém, na primeira tarefa apresentaram resultados similares. Na simulação, o método EVOSPACE teve melhor desempenho que o método CHOCOLATE nas duas primeiras tarefas e na última tarefa desempenhos similares.

Observando o comportamento coletivo do método CHOCOLATE, nas três missões foram observados comportamentos similares. O método CHOCOLATE fracassou em abordar tarefas com restrições de distâncias com coesão. O método não consegue abordar problemas em que a distribuição uniforme dos robôs no cenário de trabalho é de relevância. Isto foi visto nos resultados dos experimentos do capítulo 4 e confirmado nos experimentos deste capítulo. No total, foram apresentadas para a abordagem 5 tarefas diferentes da mesma classe de tarefas, no intuito de colocar diversos tipos de pressão no algoritmo de otimização para resolver esse problema. Nenhuma das funções objetivo no método CHOCOLATE conseguiu o propósito de produzir um comportamento coletivo de coesão entre os robôs do enxame. Isto reafirma que as interações que podem ser produzidas pelo método estão limitadas.

Em relação ao método EVOSPACE, a restrição de distâncias parece não ser bem explorada para abordar problemas de coesão do enxame. Os comportamentos obtidos são muito variáveis e dependentes do controlador produzido, não tendo consistência. Como exceções foram observados comportamentos relevantes similares ao método MATE. Por sua vez, o método EVOSPACE possui as mesmas limitações que o método EVOSTICK no que diz respeito ao ajuste das interações: O método EVOSPACE explora as deficiências na modelagem do robô no simulador e produz comportamentos não realizáveis nos robôs físicos, tais como o seguidor de parede que não foi observado nos robôs físicos, mas observado nos robôs simulados. Por isto, é observada a grande diferença de desempenho entre a simulação e os robôs físicos.

No tocante ao comportamento coletivo observado no método MATE, as restrições de distâncias incluídas foram relevantes e incluídas pelo projeto automático para resolver as tarefas coletivas apresentadas neste capítulo. Os robôs conseguem explorar a arena e logram manter distâncias específicas quando se distribuem para cobrir as regiões de interesse. A inclusão do módulo *formar* permite os robôs se organizarem no cenário de trabalho, maximizando a área coberta com a menor quantidade de robôs permitindo redundância na cobertura com os robôs restantes. O projeto automático ajustou as interações para os robôs se organizarem no cenário de trabalho, mostrando a relevância da inclusão da restrição de distâncias no método.

Capítulo 7

Conclusão

Nesta tese foi abordado o problema de restrições de distâncias no contexto de projeto automático de controladores para enxames de robôs. Este estudo foi efetuado no contexto de tarefas de robótica de enxame que devem ser realizadas com comportamentos coletivos de coesão para manter distâncias específicas entre os robôs.

Foram realizados experimentos para observar se os métodos de projeto automáticos possuem limitações para resolver tarefas coletivas em que a coesão é relevante. Para a realização dos experimentos foram utilizados os métodos AUTOMODE e EVOSTICK, referenciadas na bibliografia da robótica de enxame. Os métodos AUTOMODE e EVOSTICK são duas abordagens de projeto automático de controladores para robôs de enxame. Essas abordagens foram comparadas com três novas especializações do método CHOCOLATE, chamadas de CHOCOCAM, CHOCOLLIC e CHOCAMLLIC. Estas três novas especializações foram propostas para analisar se estas modificações da especialização produzem o comportamento desejado de coesão.

Foram realizados experimentos em simulação em 6 tarefas diferentes para comparar as abordagens. As tarefas foram divididas em três grandes grupos: *Comportamento de agrupação de objetos*, *Comportamento de agregação* e *Comportamento de formação de padrões*. Dos resultados experimentais, foi observado que todos os métodos resolvem as tarefas de *Comportamento de agrupação de objetos* e *Comportamento de agregação* com comportamentos coletivos similares. Porém, foi observado que nenhum dos métodos propostos conseguem resolver corretamente as tarefas coletivas que requerem *Comportamento de formação de*

padrões: a coesão entre robôs não aparece pois não são mantidas distâncias específicas entre os robôs do enxame.

Os experimentos foram realizados utilizando o mesmo protocolo, efetuando variações entre os métodos tais como o tipo de arquitetura de controlador utilizada, o tipo de hardware utilizado pelo robô, o algoritmo de otimização e a função objetivo. Como conclusão do anterior pode ser dito que a limitação dos métodos se encontra nas restrições das relações de entrada e saída das arquiteturas. O anterior nos leva a concluir que as restrições de distâncias incluídas nos métodos de projeto não permitiram obter o comportamento coletivo capaz de obter a coesão nas interações entre os robôs.

Para resolver a limitação, foi estudada a viabilidade de um comportamento, chamado de *formar* que permite modelar as interações entre robôs do enxame. O módulo utiliza a lei de campos potenciais artificiais e, mais especificamente, a função de Lennard-Jones como lei de controle de cada robô do enxame. O módulo *formar* foi utilizado numa arquitetura modular baseada em comportamentos para realizar movimento coletivo no enxame de robôs. O módulo *formar* define um comportamento individual que apresenta a flexibilidade necessária para um movimento com alta conectividade, e por sua vez, a capacidade de manobrabilidade para evitar colisões. Também, permite que robôs não informados do objetivo consigam se deslocar seguindo os seus robôs vizinhos do enxame. Este módulo foi construído sobre os princípios da escalabilidade, flexibilidade e localidade das interações. Assim, quando o módulo *formar* é instanciado nos robôs do enxame permite atingir o comportamento coletivo de coesão entre os robôs. Foi concluído dos resultados experimentais que o comportamento *formar* possui bom desempenho isoladamente e conjuntamente com os outros comportamentos de restrições de distâncias. Por sua vez, foi mostrada que a coesão entre robôs é necessária para manter controladas as interações entre robôs, evitar colisões entre robôs e manter a conectividade do sistema.

Finalmente, da observação das deficiências dos métodos automático de projeto de controladores em enxames de robôs, no que diz respeito as tarefas que requerem coesão, foi proposta uma extensão no método AUTOMODE-CHOCOLATE e no método EVOSTICK que incrementa a capacidade do enxame de manter distâncias específicas no cenário. As extensões dos métodos são chamadas de MATE e EVOSPACE. Foram especializados ambos métodos, incluindo o módulo *formar* como comportamento primitivo que permite ajustar as

relações de interação entre os robôs para produzir comportamentos coletivos relevantes em tarefas que requerem coesão entre os robôs. Por sua vez, os controladores produzidos pelo método MATE modificam o controlador de baixo nível do robô para permitir um controle de velocidade linear e angular com a rotação no próprio eixo do robô.

Logo, os métodos MATE e EVOSPACE foram comparados com o método CHOCOLATE em três missões que possuem restrições de distâncias na sua definição. Observando o comportamento coletivo do método CHOCOLATE nos experimentos com restrições de distâncias realizados nesta tese, observa-se que a especialização do método CHOCOLATE possui limitações para abordar a classe de tarefas que requerem coesão entre os robôs. O método EVOSPACE não conseguiu resolver de forma satisfatória estes mesmos problemas de coesão do enxame. As restrições de distâncias impostas na entrada da rede não foram bem exploradas, não obtendo controladores consistentes. A especialização de MATE, por sua vez, conseguiu abordar de forma satisfatória tarefas que requerem coesão.

O método MATE obteve comportamentos coletivos consistentes nas tarefas apresentadas nesta tese. Os resultados experimentais apontaram que o método teve melhor desempenho que os resultados dos métodos CHOCOLATE e EVOSPACE. Finalmente, a especialização de MATE conseguiu abordar uma ampla quantidade de tarefas na robótica de enxame, por possuir uma especialização mais abrangente, no que diz respeito a quantidade de tarefas que o método pode abordar.

7.1 Perspectivas de trabalhos futuros

Algumas linhas de pesquisa que podem ser seguidas a partir dos estudos apresentados nesta tese, bem como as questões que não foram foco deste trabalho, estão listadas abaixo como sugestões para trabalhos futuros:

- Avaliar o ajuste adaptável de parâmetros no comportamento individual de cada robô.
- Estudar uma nova especialização dos métodos EVOSTICK e EVOSPACE para produzir comportamentos coletivos relevantes com coesão entre robôs.
- Incluir novas características nos métodos de projeto automático para permitir produzir outros comportamentos relevantes com restrições de distâncias.

- Estudar como deve ser realizada a generalização dos métodos automáticos para permitir produzir outros comportamentos coletivos, além dos comportamentos coletivos com restrições de distância.
- Adicionar capacidades de comunicação entre robôs na síntese dos controladores.
- Estudar o impacto de características não modeladas no projeto automático de controladores para robôs de enxame.

Referências bibliográficas

- 1 SICILIANO, Bruno; KHATIB, Oussama. *Springer Handbook of Robotics*. 2nd. ed. Seacaus, USA: Springer Publishing Company, Incorporated, 2016.
- 2 ARKIN, Ronald C. *Behavior-based robotics*. Cambridge (Mass.), London, England: MIT Press, 1998.
- 3 DUDEK, Gregory; JENKIN, Michael R. M.; MILIOS, Evangelos; WILKES, David. A taxonomy for multi-agent robotics. *Autonomous Robots*, v. 3, n. 4, p. 375–397, Dec 1996.
- 4 PINCIROLI, Carlo; BIRATTARI, Mauro; TUCI, Elio; DORIGO, Marco; DEL REY ZAPATERO, Marco; VINKO, Tamas; IZZO, Dario. Lattice Formation in Space for a Swarm of Pico Satellites. In: DORIGO, Marco; BIRATTARI, Mauro; BLUM, Christian; CLERC, Maurice; STÜTZLE, Thomas; WINFIELD, Alan F. T. (Ed.). *Ant Colony Optimization and Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2008. p. 347–354.
- 5 HETTIARACHCHI, Suranga; SPEARS, William M. Distributed adaptive swarm for obstacle avoidance. *International Journal of Intelligent Computing and Cybernetics*, v. 2, n. 4, p. 644–671, 2009.
- 6 HOWARD, Andrew; MATARIĆ, Maja J.; SUKHATME, Gaurav S. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In: ASAMA, Hajime; ARAI, Tamio; FUKUDA, Toshio; HASEGAWA, Tsutomu (Ed.). *Distributed Autonomous Robotic Systems 5*. Tokyo, Japan: Springer Japan, 2002. p. 299–308.
- 7 CAO, Y. Uny; FUKUNAGA, Alex S.; KAHNG, Andrew. Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, v. 4, n. 1, p. 7–27, Mar 1997.
- 8 IOCCHI, Luca; NARDI, Daniele; SALERNO, Massimiliano. Reactivity and Deliberation: A Survey on Multi-Robot Systems. In: *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*. Berlin, Germany: Springer Berlin Heidelberg, 2001. p. 9–32.
- 9 PARKER, Lynne E. Multiple Mobile Robot Systems. In: SICILIANO, Bruno; KHATIB, Oussama (Ed.). *Springer Handbook of Robotics*. Berlin, Germany: Springer Berlin Heidelberg, 2008. p. 921–941.
- 10 YAN, Zhi; JOUANDEAU, Nicolas; CHERIF, Arab Ali. A Survey and Analysis of Multi-Robot Coordination. *International Journal of Advanced Robotic Systems*, v. 10, n. 12, p. 399, 2013.

- 11 DARMANIN, R. N.; BUGEJA, M. K. A review on multi-robot systems categorised by application domain. In: *2017 25th Mediterranean Conference on Control and Automation (MED)*. Valletta, Malta: IEEE, 2017. p. 701–706.
- 12 ŞAHIN, Erol. Swarm Robotics: From Sources of Inspiration to Domains of Application. In: ŞAHIN, Erol; SPEARS, William M. (Ed.). *Swarm Robotics*. Berlin, Germany: Springer Berlin Heidelberg, 2005. p. 10–20.
- 13 BRAMBILLA, Manuele; FERRANTE, Eliseo; BIRATTARI, Mauro; DORIGO, Marco. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, v. 7, n. 1, p. 1–41, Mar 2013.
- 14 LOPES, Yuri K.; TRENKWALDER, Stefan M.; LEAL, André B.; DODD, Tony J.; GROß, Roderich. Supervisory control theory applied to swarm robotics. *Swarm Intelligence*, v. 10, n. 1, p. 65–97, Mar 2016.
- 15 RUBENSTEIN, Michael; CORNEJO, Alejandro; NAGPAL, Radhika. Programmable self-assembly in a thousand-robot swarm. *Science*, American Association for the Advancement of Science, v. 345, n. 6198, p. 795–799, 2014.
- 16 GARATTONI, L.; BIRATTARI, M. *Swarm Robotics*. 2016. 1-19 p.
- 17 LABELLA, Thomas H.; DORIGO, Marco; DENEUBOURG, Jean-Louis. Division of Labor in a Group of Robots Inspired by Ants' Foraging Behavior. *ACM Trans. Auton. Adapt. Syst.*, ACM, New York, USA, v. 1, n. 1, p. 4–25, 2006.
- 18 FRANCESCA, G.; BRAMBILLA, M.; BRUTSCHY, A.; TRIANNI, V.; BIRATTARI, M. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, v. 8, n. 2, p. 89–112, 2014.
- 19 FRANCESCA, G.; BRAMBILLA, M.; BRUTSCHY, A.; GARATTONI, L.; MILETITCH, R. AutoMoDe-Chocolate : automatic design of control software for robot swarms. *Swarm Intelligence*, v. 9, n. 2-3, p. 125–152, 2015.
- 20 HASSELMANN, Ken; ROBERT, Frédéric; BIRATTARI, Mauro. Automatic Design of Communication-Based Behaviors for Robot Swarms. In: DORIGO, Marco; BIRATTARI, Mauro; BLUM, Christian; CHRISTENSEN, Anders L.; REINA, Andreagiovanni; TRIANNI, Vito (Ed.). *Swarm Intelligence*. Cham, Switzerland: Springer International Publishing, 2018. p. 16–29.
- 21 GAZI, V.; PASSINO, K. M. Stability Analysis of Social Foraging Swarms. *Trans. Sys. Man Cyber. Part B*, IEEE Press, Piscataway, USA, v. 34, n. 1, p. 539–557, 2004.
- 22 FRANCESCA, Gianpiero; BRAMBILLA, Manuele; TRIANNI, Vito; DORIGO, Marco; BIRATTARI, Mauro. Analysing an Evolved Robotic Behaviour Using a Biological Model of Collegial Decision Making. In: *From Animals to Animats*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 381–390.

- 23 GAUCI, Melvin; CHEN, Jianing; LI, Wei; DODD, Tony J.; GROß, Roderich. Self-organized aggregation without computation. *The International Journal of Robotics Research*, v. 33, n. 8, p. 1145–1161, 2014.
- 24 MONDADA, Francesco; BONANI, Michael; GUIGNARD, André; MAGNENAT, Stéphane; STUDER, Christian; FLOREANO, Dario. Superlinear Physical Performances in a SWARM-BOT. In: CAPCARRÈRE, Mathieu S.; FREITAS, Alex A.; BENTLEY, Peter J.; JOHNSON, Colin G.; TIMMIS, Jon (Ed.). *Advances in Artificial Life*. Berlin, Germany: Springer Berlin Heidelberg, 2005. p. 282–291.
- 25 DORIGO, M.; FLOREANO, D.; GAMBARDELLA, L. M.; MONDADA, F.; NOLFI, S.; BAABOURA, T.; BIRATTARI, M.; BONANI, M.; BRAMBILLA, M.; BRUTSCHY, A.; BURNIER, D.; CAMPO, A.; CHRISTENSEN, A. L.; DECUGNIERE, A.; DI CARO, G.; DUCATELLE, F.; FERRANTE, E.; FORSTER, A.; GONZALES, J. M.; GUZZI, J.; LONGCHAMP, V.; MAGNENAT, S.; MATHEWS, N.; MONTES DE OCA, M.; O'GRADY, R.; PINCIROLI, C.; PINI, G.; RETORNAZ, P.; ROBERTS, J.; SPERATI, V.; STIRLING, T.; STRANIERI, A.; STÜTZLE, T.; TRIANNI, V.; TUCI, E.; TURGUT, A. E.; VAUSSARD, F. Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. *IEEE Robotics Automation Magazine*, v. 20, n. 4, p. 60–71, Dec 2013.
- 26 BECKERS, Ralph; HOLLAND, Owen E.; DENEUBOURG, Jean-Louis. From Local Actions to Global Tasks: Stigmergy and Collective Robotics. In: CRUSE, Holk; DEAN, Jeffrey; RITTER, Helge (Ed.). *Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic*. Dordrecht, The Netherlands: Springer Netherlands, 2000. p. 1008–1022.
- 27 PETERSEN, Kirstin; NAGPAL, Radhika; WERFEL, Justin. TERMES: An Autonomous Robotic System for Three-Dimensional Collective Construction. In: DURRANT-WHYTE, Hugh F.; ROY, Nicholas; ABBEEL, Pieter (Ed.). *Robotics: Science and Systems*. Cambridge, USA: MIT Press, 2011.
- 28 GARATTONI, Lorenzo; BIRATTARI, Mauro. Autonomous task sequencing in a robot swarm. *Science Robotics*, v. 3, p. eaat0430, 07 2018.
- 29 SPEARS, William M.; SPEARS, Diana F.; HAMANN, Jerry C.; HEIL, Rodney. Distributed, Physics-Based Control of Swarms of Vehicles. *Autonomous Robots*, v. 17, n. 2, p. 137–162, 2004.
- 30 JONES, J. E. On the determination of molecular fields. *Proceedings of the Royal Society of London. Series A*, v. 106, p. 463–477, 1924.
- 31 NELSON, Andrew L.; BARLOW, Gregory J.; DOITSIDIS, Lefteris. Fitness Functions in Evolutionary Robotics: A Survey and Analysis. *Robot. Auton. Syst.*, North-Holland Publishing Co., Amsterdam, The Netherlands, v. 57, n. 4, p. 345–370, 2009.
- 32 KÖNIG, L. *Complex Behavior in Evolutionary Robotics*. Karlsruhe, Germany: De Gruyter, 2015.
- 33 MATARIĆ, Maja J. *Interaction and Intelligent Behavior*. 1994.

- 34 DUARTE, Miguel; OLIVEIRA, Sancho Moura; CHRISTENSEN, Anders Lyhne. Evolution of Hybrid Robotic Controllers for Complex Tasks. *Journal of Intelligent & Robotic Systems*, v. 78, n. 3, p. 463–484, 2015.
- 35 BAHÇEÇI, Erkin; SOYSAL, Onur; ŞAHİN, Erol. *A Review: Pattern Formation and Adaptation in Multi-Robot Systems*. Pittsburgh, USA, 2003.
- 36 BROOKS, Rodney. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, IEEE Press, v. 2, n. 1, p. 14–23, 1986.
- 37 KHATIB, O. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *Int'l Journal of Robotics Research*, Sage Publications, Inc., Thousand Oaks, USA, v. 5, n. 1, p. 90–98, 1986.
- 38 SPEARS, W. M.; GORDON, D. F. Using artificial physics to control agents. In: *Proceedings 1999 International Conference on Information Intelligence and Systems (Cat. No. PR00446)*. Bethesda, USA: IEEE, 1999. p. 281–288.
- 39 RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event processes. In: BENSOUSSAN, A.; LIONS, J. L. (Ed.). *Analysis and Optimization of Systems*. Berlin, Germany: Springer Berlin Heidelberg, 1984. p. 475–498.
- 40 PARKER, L. E. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, v. 14, n. 2, p. 220–240, 1998.
- 41 BALCH, T.; ARKIN, R. C. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, v. 14, n. 6, p. 926–939, 1998.
- 42 REIF, John H.; WANG, Hongyan. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, v. 27, n. 3, p. 171–194, 1999.
- 43 SHUCKER, Brian; BENNETT, John K. Scalable Control of Distributed Robotic Macrosensors. In: ALAMI, Rachid; CHATILA, Raja; ASAMA, Hajime (Ed.). *Distributed Autonomous Robotic Systems 6*. Tokyo: Springer Japan, 2007. p. 379–388.
- 44 SPEARS, William M.; SPEARS, Diana F.; HEIL, Rodney; KERR, Wesley; HETTIARACHCHI, Suranga. An Overview of Physicomimetics. In: ŞAHİN, Erol; SPEARS, William M. (Ed.). *Swarm Robotics*. Berlin, Germany: Springer Berlin Heidelberg, 2005. p. 84–97.
- 45 HSU, Harry; LIU, Alan. Multiagent-Based Multi-team Formation Control for Mobile Robots. *Journal of Intelligent and Robotic Systems*, Kluwer Academic Publishers, v. 42, n. 4, p. 337–360, 2005.
- 46 TURGUT, Ali E.; ÇELIKKANAT, Hande; GÖKÇE, Fatih; ŞAHİN, Erol. Self-Organized Flocking with a Mobile Robot Swarm. In: PADGHAM, Lin; PARKES, David C.; MÜLLER, PARSONS, Simon (Eds.). *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems*. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008. p. 12–16.

- 47 SHUCKER, B.; MURPHEY, T. D.; BENNETT, J. K. Convergence-Preserving Switching for Topology-Dependent Decentralized Systems. *IEEE Transactions on Robotics*, v. 24, n. 6, p. 1405–1415, 2008.
- 48 KAZADI, S. Model independence in swarm robotics. *International Journal of Intelligent Computing and Cybernetics*, v. 2, n. 4, p. 672–694, 2009.
- 49 MENDIBURU, F. J.; MORAIS, M. R. A.; LIMA, A. M. N. Behavior coordination in multi-robot systems. In: *IEEE International Conference on Automatica (ICA-ACCA)*. Curico, Chile: IEEE, 2016. p. 1–7.
- 50 LOPES, Yuri K.; LEAL, André B.; DODD, Tony J.; GROß, Roderich. Application of Supervisory Control Theory to Swarms of e-puck and Kilobot Robots. In: DORIGO, Marco; BIRATTARI, Mauro; GARNIER, Simon; HAMANN, Heiko; MONTES DE OCA, Marco; SOLNON, Christine; STÜTZLE, Thomas (Ed.). *Swarm Intelligence*. Cham, Switzerland: Springer International Publishing, 2014. p. 62–73.
- 51 LOPES, Yuri Kaszubowski; TRENKWALDER, Stefan M.; LEAL, André B.; DODD, Tony J.; GROß, Roderich. Probabilistic Supervisory Control Theory (pSCT) Applied to Swarm Robotics. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. Richland, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2017. (AAMAS '17), p. 1395–1403.
- 52 MONDADA, Francesco; BONANI, Michael; RAEMY, Xavier; PUGH, James; CIANCI, Christopher; KLAPTOCZ, Adam; MAGNENAT, Stéphane; ZUFFEREY, Jean christophe; FLOREANO, Dario; MARTINOLI, Alcherio. The e-puck, a robot designed for education in engineering. *9th Conference on Autonomous Robot Systems and Competitions*, Castelo Branco, Portugal, p. 59–65, 2009.
- 53 RUBENSTEIN, M.; AHLER, C.; NAGPAL, R. Kilobot: A low cost scalable robot system for collective behaviors. In: *2012 IEEE International Conference on Robotics and Automation*. Saint Paul, USA: IEEE, 2012. p. 3293–3298.
- 54 TURGUT, Ali; GÖKÇE, Fatih; ÇELIKKANAT, Hande; BAYINDIR, Levent; ŞAHİN, Erol. *Kobot: A mobile robot designed specifically for swarm robotics research*. Ankara, Turquia, 2007.
- 55 STANLEY, Kenneth O.; MIIKKULAINEN, Risto. Evolving Neural Networks Through Augmenting Topologies. *Evol. Comput.*, MIT Press, Cambridge, USA, v. 10, n. 2, p. 99–127, 2002.
- 56 QUINN, M.; SMITH, L.; MAYLEY, G.; HUSBANDS, P. Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, v. 361, n. 1811, p. 2321–2343, 2003.
- 57 DORIGO, Marco; TRIANNI, Vito; ŞAHİN, Erol; GROß, Roderich; LABELLA, Thomas H.; BALDASSARRE, Gianluca; NOLFI, Stefano; DENEUBOURG, Jean-Louis; MONDADA, Francesco; FLOREANO, Dario; GAMBARDELLA, Luca M. Evolving Self-Organizing Behaviors for a Swarm-Bot. *Autonomous Robots*, v. 17, n. 2, p. 223–245, 2004.

- 58 GAUCI, Melvin; CHEN, Jianing; LI, Wei; DODD, Tony J.; GROß, Roderich. Clustering Objects with Robots That Do Not Compute. In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. Richland, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2014. (AAMAS '14), p. 421–428.
- 59 SILVA, F.; DUARTE, M.; OLIVEIRA, S.; CORREIA, L. C.; CHRISTENSEN, A. The Case for Engineering the Evolution of Robot Controllers. In: *International Conf. on the Simulation and Synthesis of Living Systems - ALIFE*. New York, USA, 2014. p. 703–710.
- 60 DUARTE, Miguel; OLIVEIRA, Sancho; CHRISTENSEN, Anders. Evolution of Hierarchical Controllers for Multirobot Systems. *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, p. 657–664, 2014.
- 61 DUARTE, Miguel; OLIVEIRA, Sancho; CHRISTENSEN, Anders. Hybrid Control for Large Swarms of Aquatic Drones. *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, p. 785–792, 2014.
- 62 DUARTE, Miguel; COSTA, Vasco; GOMES, Jorge C.; RODRIGUES, Tiago; SILVA, Fernando; OLIVEIRA, Sancho Moura; CHRISTENSEN, Anders Lyhne. Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLoS ONE*, v. 3, n. 11, 2016.
- 63 DUARTE, Miguel; GOMES, Jorge; COSTA, Vasco; OLIVEIRA, Sancho Moura; CHRISTENSEN, Anders Lyhne. Hybrid Control for a Real Swarm Robotics System in an Intruder Detection Task. In: SQUILLERO, Giovanni; BURELLI, Paolo (Ed.). *Applications of Evolutionary Computation*. Cham, Switzerland: Springer International Publishing, 2016. p. 213–230.
- 64 HETTIARACHCHI, Suranga; SPEARS, William. Moving Swarm Formations through Obstacle Fields. In: *Proceedings of the 2005 International Conference on Artificial Intelligence, ICAI'05*. Pittsburgh, USA, 2005. v. 1, p. 97–103.
- 65 HETTIARACHCHI, Suranga; SPEARS, W. DAEDALUS for Agents with Obstructed Perception. *IEEE Mountain Workshop on Adaptive and Learning Systems*, p. 195–200, 2006.
- 66 KÖNIG, Lukas; MOSTAGHIM, Sanaz; SCHMECK, Hartmut. Decentralized Evolution of Robotic Behavior using Finite State Machines. *Int. Journal of Intelligent Computing and Cybernetics*, v. 2, p. 695–723, 11 2009.
- 67 KUCKLING, Jonas; LIGOT, Antoine; BOZHINOSKI, Darko; BIRATTARI, Mauro. Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms. In: DORIGO, Marco; BIRATTARI, Mauro; BLUM, Christian; CHRISTENSEN, Anders L.; REINA, Andreagiovanni; TRIANNI, Vito (Ed.). *Swarm Intelligence*. Cham, Switzerland: Springer International Publishing, 2018. p. 30–43.

- 68 PINCIROLI, C.; BIRATTARI, M.; TUCI, E.; DORIGO, M.; DEL REY ZAPATERO, M.; VINKO, T.; IZZO, D. Self-Organizing and Scalable Shape Formation for a Swarm of Pico Satellites. In: *2008 NASA/ESA Conference on Adaptive Hardware and Systems*. Noordwijk, The Netherlands: IEEE, 2008. p. 57–61.
- 69 FRANCESCA, Gianpiero. *A modular approach to the automatic design of control software for robot swarms*. Tese (Doutorado) — Université Libre de Bruxelles, Belgium, 2017.
- 70 RABIN, Michael O. Probabilistic Automata. *Information and Control*, v. 6, n. 3, p. 230–245, 1963.
- 71 PINCIROLI, Carlo; TRIANNI, Vito; O'GRADY, Rehan; PINI, Giovanni; BRUTSCHY, Arne; BRAMBILLA, Manuele; MATHEWS, Nithin; FERRANTE, Eliseo; DI CARO, Gianni; DUCATELLE, Frederick; STIRLING, Timothy; GUTIÉRREZ, Álvaro; GAMBARDELLA, Luca Maria; DORIGO, Marco. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*. Los Alamitos, USA: IEEE Computer Society Press, 2011. p. 5027–5034.
- 72 PINCIROLI, Carlo; TRIANNI, Vito; O'GRADY, Rehan; PINI, Giovanni; BRUTSCHY, Arne; BRAMBILLA, Manuele; MATHEWS, Nithin; FERRANTE, Eliseo; DI CARO, Gianni; DUCATELLE, Frederick; BIRATTARI, Mauro; GAMBARDELLA, Luca Maria; DORIGO, Marco. ARGoS: a Modular, Parallel, Multi-engine Simulator for Multi-Robot Systems. *Swarm Intelligence*, Springer, Berlin, Germany, v. 6, n. 4, p. 271–295, 2012.
- 73 BIRATTARI, Mauro; STÜTZLE, Thomas; PAQUETE, Luis; VARRENTRAPP, Klaus. A Racing Algorithm for Configuring Metaheuristics. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, USA: Morgan Kaufmann Publishers Inc., 2002. (GECCO '02), p. 11–18.
- 74 BIRATTARI, M.; YUAN, Z.; BALAPRAKASH, P.; STÜTZLE, T. F-Race and Iterated F-Race: An Overview. In: BARTZ-BEIELSTEIN, Thomas; CHIARANDINI, Marco; PAQUETE, Luís; PREUSS, Mike (Ed.). *Experimental Methods for the Analysis of Optimization Algorithms*. Berlin, Germany: Springer Berlin Heidelberg, 2010. p. 311–336.
- 75 LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. Pérez; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016.
- 76 MONDADA, Francesco; PETTINARO, Giovanni C.; GUIGNARD, Andre; KWEE, Ivo W.; FLOREANO, Dario; DENEUBOURG, Jean-Louis; NOLFI, Stefano; GAMBARDELLA, Luca Maria; DORIGO, Marco. Swarm-Bot: A New Distributed Robotic Concept. *Autonomous Robots*, v. 17, n. 2, p. 193–221, 2004.
- 77 DORIGO, M.; ŞAHIN, E. *Swarm Robotics*. 2004. 111–113 p.
- 78 TAN, Ying; ZHENG, Z.-Y. Research Advance in Swarm Robotics. *Defence Technology*, v. 9, n. 1, p. 18–39, 2013.

- 79 MATARIĆ, Maja J. *A Distributed Model for Mobile Robot Environment-Learning and Navigation*. Cambridge, USA, 1990.
- 80 PENG, X.; ZHANG, S.; HUANG, Y. Pattern formation in constrained environments: A swarm robot target trapping method. In: *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*. Macau, China: IEEE, 2016. p. 455–460.
- 81 GUZEL, M. S.; GEZER, E. C.; AJABSHIR, V. B.; BOSTANCI, E. An adaptive pattern formation approach for swarm robots. In: *2017 4th International Conference on Electrical and Electronic Engineering (ICEEE)*. Ankara, Turkey: IEEE, 2017. p. 194–198.
- 82 OH, H.; JIN, Y. Evolving hierarchical gene regulatory networks for morphogenetic pattern formation of swarm robots. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. Beijing, China: IEEE, 2014. p. 776–783.
- 83 YANG, B.; DING, Y.; HAO, K. Area coverage searching for swarm robots using dynamic Voronoi-based method. In: *2015 34th Chinese Control Conference (CCC)*. Hangzhou, China: IEEE, 2015. p. 6090–6094.
- 84 DE SOUZA, B. J. O.; ENDLER, M. Coordinating movement within swarms of UAVs through mobile networks. In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. St. Louis, USA: IEEE, 2015. p. 154–159.
- 85 ABSHOFF, S.; CORD-LANDWEHR, A.; FISCHER, M.; JUNG, D.; HEIDE, F. M. Gathering a Closed Chain of Robots on a Grid. In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Chicago, USA, 2016. p. 689–699.
- 86 LENG, Y.; ZHANG, Y.; ZHANG, W.; HE, X.; BIAN, D.; ZHOU, W. SociBuilder: A novel task-oriented swarm robotic system. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Zhuhai, China: IEEE, 2015. p. 48–53.
- 87 BALDASSARRE, G.; TRIANNI, V.; BONANI, M.; MONDADA, F.; DORIGO, M.; NOLFI, S. Self-Organized Coordinated Motion in Groups of Physically Connected Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 37, n. 1, p. 224–239, Feb 2007.
- 88 KEGELEIRS, M.; GARZÓN RAMOS, D.; BIRATTARI, M. *Random walk exploration for swarm mapping*. Brussels, Belgium, 2019.
- 89 FERRANTE, Eliseo; TURGUT, Ali Emre; HUEPE, Cristián; STRANIERI, Alessandro; PINCIROLI, Carlo; DORIGO, Marco. Self-organized Flocking with a Mobile Robot Swarm: A Novel Motion Control Method. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, Sage Publications, Inc., Thousand Oaks, USA, v. 20, n. 6, p. 460–477, dez. 2012.
- 90 NAVARRO, Iñaki; MATÍA, Fernando. Distributed orientation agreement in a group of robots. *Auton. Robots*, v. 33, n. 4, p. 445–465, 2012.

- 91 ELKILANY, B. G.; ABOUELSOUD, A. A.; FATHELBAB, A. M. R. Adaptive formation control of robot swarms using optimized potential field method. In: *2017 IEEE International Conference on Industrial Technology (ICIT)*. Toronto, Canada: IEEE, 2017. p. 721–725.
- 92 CHEN, J.; GAUCI, M.; GROß, R. A strategy for transporting tall objects with a swarm of miniature mobile robots. In: *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, 2013. p. 863–869.
- 93 RUIZ, D.; BACCA, B.; CAICEDO, E. Control Strategy Based on Swarms Algorithms to Cooperative Payload Transport Using a Non-Holonomic Mobile Robots Group. *IEEE Latin America Transactions*, v. 14, n. 2, p. 445–456, 2016.
- 94 TRIANNI, V.; DE SIMONE, D.; REINA, A.; BARONCHELLI, A. Emergence of Consensus in a Multi-Robot Network: From Abstract Models to Empirical Validation. *IEEE Robotics and Automation Letters*, v. 1, n. 1, p. 348–353, Jan 2016.
- 95 SCHEIDLER, A.; BRUTSCHY, A.; FERRANTE, E.; DORIGO, M. The k-Unanimity Rule for Self-Organized Decision-Making in Swarms of Robots. *IEEE Transactions on Cybernetics*, v. 46, n. 5, p. 1175–1188, May 2016.
- 96 TRABATTONI, Marco; VALENTINI, Gabriele; DORIGO, Marco. Hybrid Control of Swarms for Resource Selection. In: *Proceedings of the 11th International Conference, ANTS 2018*. Rome, Italy: Springer Cham, 2018. p. 57–70.
- 97 STROBEL, Volker; FERRER, Eduardo Castelló; DORIGO, Marco. Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. Stockholm, Sweden, 2018. (AAMAS '18), p. 541–549.
- 98 AGASSOUNON, William; MARTINOLI, Alcherio. Efficiency and Robustness of Threshold-based Distributed Allocation Algorithms in Multi-agent Systems. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*. New York, USA: ACM, 2002. (AAMAS '02), p. 1090–1097.
- 99 PANG, B.; ZHANG, C.; SONG, Y.; WANG, H. Self-organized task allocation in swarm robotics foraging based on dynamical response threshold approach. In: *2017 18th International Conference on Advanced Robotics (ICAR)*. Hong Kong, China: IEEE, 2017. p. 256–261.
- 100 IRFAN, M.; FAROOQ, A. Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities. In: *2016 International Conference on Intelligent Systems Engineering (ICISE)*. Islamabad, Pakistan: IEEE, 2016. p. 210–215.
- 101 MONDADA, Francesco; FRANZI, Edoardo; IENNE, Paolo. Mobile robot miniaturisation: A tool for investigation in control algorithms. In: YOSHIKAWA, Tsuneo; MIYAZAKI, Fumio (Ed.). *Experimental Robotics III*. Berlin, Germany: Springer Berlin Heidelberg, 1994. p. 501–513.

- 102 BONANI, M.; LONGCHAMP, V.; MAGNENAT, S.; RÉTORNAZ, P.; BURNIER, D.; ROULET, G.; VAUSSARD, F.; BLEULER, H.; MONDADA, F. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei, Taiwan: IEEE, 2010. p. 4187–4193.
- 103 GUTIERREZ, A.; CAMPO, A.; DORIGO, M.; DONATE, J.; MONASTERIO-HUELIN, F.; MAGDALENA, L. Open E-puck Range & Bearing miniaturized board for local communication in swarm robotics. In: *2009 IEEE International Conference on Robotics and Automation*. Berlin, Germany: Springer Berlin Heidelberg, 2009. p. 3111–3116.
- 104 GUTIERREZ, A.; CAMPO, A.; DORIGO, M.; AMOR, D.; MAGDALENA, L.; MONASTERIO-HUELIN, F. An Open Localization and Local Communication Embodied Sensor. In: *Sensors*. Basel, Switzerland: MDPI AG, 2008. v. 8, n. 11, p. 7545–7563.
- 105 OGATA, Katsuhiko. *Modern Control Engineering (3rd Ed.)*. Upper Saddle River, USA: Prentice-Hall, Inc., 1997.
- 106 ROSS, T. J. *Fuzzy logic with engineering applications*. West Sussex, UK: John Wiley & Sons, 2010.
- 107 ÅSTRÖM, Karl Johan; WITTENMARK, Bjorn. *Adaptive Control*. 2nd. ed. Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.
- 108 MAXIM, Paul M. Uniform Coverage. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 341–366.
- 109 BRAVI, Andrea; CORRADI, Paolo; SCHLACHTER, Florian; MENCIASSI, Arianna. Local Oriented Potential Fields: Self Deployment and Coordination of an Assembling Swarm of Robots. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 129–144.
- 110 TRIANNI, V. *Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots*. Berlin, Germany: Springer, 2008.
- 111 FINE, Terrence L. *Feedforward Neural Network Methodology*. New York, USA: Springer-Verlag New York, 1999.
- 112 FLOREANO, Dario; HUSBANDS, Phil; NOLFI, Stefano. Evolutionary Robotics. In: *Springer Handbook of Robotics*. Berlin, Germany: Springer Berlin Heidelberg, 2008. p. 1423–1451.
- 113 MURPHY, Kevin P. *Machine Learning: A Probabilistic Perspective*. Cambridge, USA: The MIT Press, 2012.
- 114 FRANCESCA, Gianpiero; BIRATTARI, Mauro; DORIGO, Marco. *Automatic Design of Controllers in Swarm Robotics, Rapport d'avancement de recherche Annee Academique 2012/2013*. 2013.

- 115 KOENIG, N.; HOWARD, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan: IEEE, 2004. v. 3, p. 2149–2154.
- 116 ROHMER, E.; SINGH, S. P. N.; FREESE, M. V-REP: A versatile and scalable robot simulation framework. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan: IEEE, 2013. p. 1321–1326.
- 117 VAUGHAN, Richard. Massively multi-robot simulation in stage. *Swarm Intelligence*, v. 2, n. 2, p. 189–208, Dec 2008.
- 118 GERKEY, Brian P.; VAUGHAN, Richard T.; HOWARD, Andrew. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: . In *Proceedings of the 11th International Conference on Advanced Robotics*. Coimbra, Portugal, 2003. p. 317–323.
- 119 QUIGLEY, Morgan; CONLEY, Ken; GERKEY, Brian P.; FAUST, Josh; FOOTE, Tully; LEIBS, Jeremy; WHEELER, Rob; NG, Andrew Y. ROS: an open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*. Kobe, Japan: IEEE, 2009.
- 120 PITONAKOVA, Lenka; GIULIANI, Manuel; PIPE, Anthony; WINFIELD, Alan. Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators. In: GIULIANI, Manuel; ASSAF, Tareq; GIANNACCINI, Maria Elena (Ed.). *Towards Autonomous Robotic Systems*. Cham, Switzerland: Springer International Publishing, 2018. p. 357–368.
- 121 FLOREANO, D.; URZELAI, J. Evolutionary Robots with on-line self-organization and behavioral fitness. *Neural Networks*, v. 13, n. 4-5, p. 431–443, 2000.
- 122 BIRATTARI, M. *Tuning Metaheuristics: A Machine Learning Perspective*. Berlin, Germany: Springer Heidelberg, 2009.
- 123 CONOVER, W. J. *Practical Nonparametric Statistics*. Third. New York: John Wiley & Sons, 1999.
- 124 ULB. *Cluster Majorana*. 2019. <<http://majorana.ulb.ac.be/wordpress/>>. Online; acessado 01-05-2019.
- 125 SPEARS, Diana F.; REBGUNS, Antons; ANDERSON-SPRECHER, Richard; KLETSOV, Aleksey. A theoretical framework for estimating swarm success probability using scouts. *Int. J. Swarm. Intell. Res.*, IGI Global, Hershey, USA, v. 1, n. 4, p. 17–45, 2010.
- 126 SPEARS, Diana F.; ANDERSON-SPRECHER, Richard; KLETSOV, Aleksey; REBGUNS, Antons. A Statistical Framework for Estimating the Success Rate of Liquid-Mimetic Swarms. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 475–503.
- 127 SPEARS, Diana; THAYER, David; ZARZHITSKY, Dimitri V. A Multi-robot Chemical Source Localization Strategy Based on Fluid Physics: Theoretical Principles. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 223–249.

- 128 SPEARS, William M. Pushing the Envelope. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 93–125.
- 129 MAXIM, Paul M. Chain Formations. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 367–412.
- 130 APKER, Thomas; POTTER, Mitchell A. Physicomimetic Motion Control of Physically Constrained Agents. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 413–437.
- 131 HETTIARACHCHI, Suranga. Adaptive Learning by Robot Swarms in Unfamiliar Environments. In: *Physicomimetics: Physics-Based Swarm Intelligence*. Berlin, Germany: Springer Berlin Heidelberg, 2012. p. 441–473.
- 132 KOREN, Y.; BORENSTEIN, J. Potential field methods and their inherent limitations for mobile robot navigation. In: *1991 IEEE International Conference on Robotics and Automation*. Sacramento, USA: IEEE, 1991. v. 2, p. 1398–1404.
- 133 MABROUK, M. H.; MCINNES, C. R. Swarm robot social potential fields with internal agent dynamics. In: *12th International Conference on Aerospace Sciences and Aviation Technology, ASAT-12*. Cairo, Egypt: The Military Technical College, 2007. p. 1–14.
- 134 MABROUK, M. H.; MCINNES, C. R. Solving the potential field local minimum problem using internal agent states. *Robotics and Autonomous Systems*, v. 56, n. 12, p. 1050–1060, 2008.
- 135 BORENSTEIN, J.; KOREN, Y. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 19, n. 5, p. 1179–1187, 1989.
- 136 COUZIN, I. D.; KRAUSE, J.; JAMES, R.; RUXTON, G. D.; FRANKS, N. R. Collective Memory and Spatial Sorting in Animal Groups. *Journal of Theoretical Biology*, Elsevier, v. 218, n. 1, p. 1–11, 2002.