Uma abordagem híbrida para planejamento exploratório de trajetórias e controle de navegação de robôs móveis autônomos

Valéria de Carvalho Santos

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Valéria de Carvalho Santos

Uma abordagem híbrida para planejamento exploratório de trajetórias e controle de navegação de robôs móveis autônomos

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Fernando Santos Osório

Coorientador: Prof. Dr. Cláudio Fabiano Motta Toledo

USP – São Carlos
Dezembro de 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

d231a de Carvalho Santos, Valéria
Uma abordagem híbrida para planejamento
exploratório de trajetórias e controle de navegação de
robôs móveis autônomos / Valéria de Carvalho Santos;
orientador Fernando Santos Osório; coorientador
Cláudio Fabiano Motta Toledo. -- São Carlos, 2017.
111 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2017.

1. Planejamento Exploratório de Trajetórias. 2.
Navegação. 3. Algoritmos Genéticos. 4. Algoritmos de
Otimização por Colônia de Formigas. I. Santos Osório,
Fernando, orient. II. Motta Toledo, Cláudio
Fabiano, coorient. III. Título.

Valéria de Carvalho Santos

**A hybrid approach for exploratory path planning and
navigation control for autonomous mobile robots**

Doctoral dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Doctorate Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Fernando Santos Osório

Co-advisor: Prof. Dr. Cláudio Fabiano Motta Toledo

**USP – São Carlos
December 2017**

*Dedico este trabalho às pessoas que de alguma forma
me apoiaram e torceram pela minha vitória.*

AGRADECIMENTOS

A realização desse doutorado foi um dos maiores desafios da minha vida. Agradeço a Deus pela força, paciência, sabedoria e principalmente perseverança que me deu nesse período, que foi de grande aprendizado para mim. Eu sabia que fazer doutorado não seria fácil mas jamais imaginei que teria que enfrentar tudo que enfrentei. Também não sabia que eu seria forte o suficiente para lidar com tudo isso. Ser estudante de pós-graduação vai muito além de ser inteligente ou ter alguma habilidade científica.

Agradeço ao apoio da minha família que sempre me incentiva a ir em busca dos meus sonhos. Em especial aos meus pais Moacir e Arlene e aos meus irmãos Camilo e Clarisse.

Ao meu namorado Jadson por cuidar com tanta dedicação de mim, especialmente na etapa final da tese. Muito obrigada pela paciência, carinho e todo o apoio que tem me dado.

Sou muito grata à família que construí esses anos em São Carlos. Dizem que o amigo verdadeiro é aquele que se aproxima nos momentos difíceis. Se for assim, eu sou mesmo muito rica de amigos. Obrigada a cada um que cruzou meu caminho nesse tempo. Agradeço pelas conversas, abraços, sorrisos e tantos gestos de carinho que me ofereceram nesse período. São muitos amigos e cada um é muito especial para mim, por isso, vou ousar citar alguns nomes. Muito obrigada Bruno, Rodrigo Calvo, Chris, André Rossi, Luiz Coletta, Thiago Cupertino, Eduardo, Marcelão, Jonathan, Danilo Horta, Ricardo Cerri, Thiago Covões, Luís Paulo, Debora, Mantovani, Kemilly, Montanari, Faimison, Adam, Murillo, Everlandio, Fausto, Vinícius, Drica, Vini, Victor, Dario, Marina, Fernando Zuher, Erinaldo, Maria, Marci, Alberto, Jean, Flaviana, Joice, Letrícia, Silvana, Elis, Daniela, Rodolfo, Laura, Fernando, Nathalie, Rodolfo, Brauner, Stevão, Lina, Vânia, Giovana, Carlos, Alinne, Laíza.

Agradeço ao meu orientador Fernando Osório que acreditou na minha capacidade mais do que eu mesma. Obrigada pelo incentivo e por estar presente mesmo em períodos à distância. Ao final de cada reunião eu ficava mais tranquila e confiante para dar continuidade aos trabalhos. Agradeço também a todos os amigos do LRM pela convivência no laboratório.

Agradeço ao professor Jó Ueyama pelo apoio e pela indicação para o estágio sanduíche. À Solange Rezende que se tornou um anjo na minha vida e mostrou caminhos para que eu resgatar a auto-confiança. Agradeço ao Instituto Você e amigos que conheci lá. A energia que vocês me passaram foi muito importante para eu continuar seguindo em frente. Aos amigos de Aracaju que estão sempre torcendo pelas minhas conquistas.

Agradeço ao professor Fernando Otero, da University of Kent, por ter me recebido

durante o estágio sanduíche e pela grande contribuição no desenvolvimento do meu trabalho. Agradeço também ao professor Colin Johnson pela colaboração neste trabalho. A experiência que tive na Inglaterra enriqueceu muito a minha vida profissional e pessoal.

Agradeço aos amigos que conheci na Inglaterra, que foram essenciais para minha vivência lá, Carol e Fábio, Myriam, Armando, Danilo, Sarah, Bela, Fábio e Draylson. Agradeço também aos amigos de moradia e às queridas Carolina, Ilaria e Sophie. Ao meu grande amigo Daniel, que tive a oportunidade de reencontrar nesse período. Ter você por perto foi muito importante para mim.

Aos amigos da UFLA que acompanharam a fase final do doutorado e me deram apoio para concluir. Muito obrigada Paula, Victor, Nicole, Carol, Nelson, Andreza, Mayron, Evelise, André, Daniel, Iara, Renata, Maurício, Juliana, Janderson, Renato, Igor e Luíza.

Agradeço aos professores Adenilso, Graça Pimentel e Agma durante o período que coordenaram o programa. Muito obrigada pelo apoio de vocês. Agradeço também aos funcionários do ICMC/USP pela dedicação ao trabalho e à CAPES, CNPq e University of Kent pelo apoio financeiro.

RESUMO

SANTOS, V. C. **Uma abordagem híbrida para planejamento exploratório de trajetórias e controle de navegação de robôs móveis autônomos**. 2017. 111 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

A tarefa de planejamento de trajetórias de robôs móveis autônomos consiste em determinar objetivos intermediários para que um robô seja capaz de partir de sua localização inicial e alcançar seu objetivo final. Além do planejamento, é importante definir um método de controle da navegação (seguimento da trajetória) do robô para que ele seja capaz de realizar seu trajeto de forma segura. Este projeto propõe uma abordagem híbrida para planejamento exploratório e execução de trajetórias de robôs móveis autônomos em ambientes *indoor*. Para o planejamento de trajetória, foram investigados algoritmos de busca em espaço de estados, dando ênfase ao uso de algoritmos evolutivos e algoritmos de otimização por colônia de formigas para a descoberta e otimização da trajetória. O controle da navegação é realizado por meio de comportamentos locais reativos, baseado na exploração e uso de mapas topológicos, os quais permitem uma maior flexibilidade em termos de definição da localização da posição do robô móvel e sobre os detalhes do mapa do ambiente (mapas com informações aproximadas e não métricos). Assim, foi proposto e desenvolvido um método robusto capaz de planejar, mapear e explorar um caminho ótimo ou quase ótimo para que o robô possa navegar e alcançar seu objetivo de forma segura, com pouca informação prévia do ambiente ou mesmo sobre sua localização. Além disso, o robô pode reagir a ambientes com alterações dinâmicas em sua estrutura, considerando por exemplo, elementos dinâmicos como portas que possam ser abertas ou fechadas e passagens que são obstruídas. Por fim, foram realizados diversos testes e simulações a fim de validar o método proposto, com a avaliação da qualidade das soluções encontradas e comparação com outras abordagens tradicionais de planejamento de trajetórias (algoritmos A* e D*).

Palavras-chave: Planejamento Exploratório de Trajetórias, Navegação, Algoritmos Genéticos, Algoritmos de Otimização por Colônia de Formigas.

ABSTRACT

SANTOS, V. C. **A hybrid approach for exploratory path planning and navigation control for autonomous mobile robots**. 2017. 111 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

The task of planning path for autonomous mobile robots consists in determine intermediary goals in order to allow a robot be able to leave its initial location and reach its final goal. Besides the planning, it is important to define a method of navigation control (the trajectory following) of the robot for it be able to do its path safely. This project proposes a hybrid approach to path planning and execution of an autonomous mobile robot in indoor environments. For the path planning, search algorithms in state space have been investigated, with emphasis in evolutionary algorithms and ant colony optimization algorithms for the trajectory search and optimization. The navigation control is done by local reactive behaviors, based on topological maps, which allow more flexibility concerning localization definition of position of the mobile robot and about the details of the environment map (maps with approximate information and not metric). Thus, a robust method able to plan an optimum or almost optimum path for the robot to reach its goal safely has been proposed, with little previous information of the environment. Furthermore, the robot can react to dynamic elements in the environment structure, concerning, for example, dynamic elements such as doors that can be opened or closed and ways that are blocked. Finally, several tests and simulations has been carried out to validate the proposed method, with evaluation of the solutions quality and comparison with others traditional approaches for the path planning task (A* and D* algorithms).

Keywords: Exploratory Path Planning, Navigation, Genetic Algorithms, Ant Colony Optimization Algorithms.

LISTA DE ILUSTRAÇÕES

Figura 1 – O robô conhece as posições inicial e final aproximadas, mas não conhece o caminho para alcançar o objetivo.	28
Figura 2 – Exemplo de uma representação de mapa grade de ocupação. A figura (b) apresenta a região em destaque ampliada, para ilustrar as células ocupadas e não ocupadas.	29
Figura 3 – Exemplo de mapa e sua representação topológica em forma de grafo. Cada nó representa um local característico do ambiente e as arestas são a conectividade entre eles.	30
Figura 4 – Um exemplo de representação do indivíduo. Cada gene representa um código para uma ação.	47
Figura 5 – Um exemplo de mapa. S é a posição inicial do robô e G é o seu objetivo. . .	47
Figura 6 – O grafo correspondente da Figura 5.	48
Figura 7 – Um exemplo do robô em uma bifurcação onde todas as ações possíveis são representadas.	49
Figura 8 – Exemplo de cromossomo. Cada gene representa uma ação definida por um valor inteiro.	50
Figura 9 – Exemplo de mapa.	50
Figura 10 – O grafo correspondente da Figura 9.	51
Figura 11 – Exemplo de cruzamento uniforme.	51
Figura 12 – Exemplo dos três tipos de mutação.	52
Figura 13 – Exemplo das duas primeiras gerações de uma população com cinco robôs. .	54
Figura 14 – Exemplo de execução das sequências de ações de dois indivíduos que partem da mesma posição S para o objetivo G	54
Figura 15 – Exemplo de cálculo do valor de aptidão.	56
Figura 16 – Exemplo de um mapa e seu grafo correspondente.	57
Figura 17 – Mapa usado nos experimentos. S é a posição inicial do robô e G_1 , G_2 , G_3 e G_4 são os objetivos.	64
Figura 18 – Mapa usado nos experimentos. S_1 e G_1 são a posição inicial e final do robô, respectivamente.	67
Figura 19 – Mapa usado nos experimentos do algoritmo AG III.	71
Figura 20 – Porcentagem de soluções do AG que são melhores que os algoritmos A^* e C^* . .	74
Figura 21 – Mapa 2. S é o ponto de início e G é o objetivo.	75
Figura 22 – Mapa 3. S é o ponto de início e G é o objetivo.	76

Figura 23 – Mapa utilizado nos experimentos do algoritmo MMAS-SPP	77
Figura 24 – Aresta removida do início da melhor solução é destacada com a linha vermelha no mapa. A linha azul apresenta as novas arestas da solução.	85
Figura 25 – Aresta removida do meio da melhor solução é destacada em vermelho no mapa. A linha azul apresenta as novas arestas da solução.	86
Figura 26 – Aresta removida do final da melhor solução é destacada em vermelho. A linha azul apresenta as novas arestas da solução.	87
Figura 27 – O mapa com o novo objetivo determinado.	89
Figura 28 – Um exemplo de um robô no simulador de robô V-REP. Os pontos vermelhos indicam onde o sensor laser atinge os obstáculos.	90
Figura 29 – Visão geral da arquitetura do sistema Neuro-FSM.	95
Figura 30 – Exemplos de percepção e estado do robô (a) até (e) e o grafo representando um mapa topológico com um caminho pré-definido.	95
Figura 31 – Diagrama do cálculo da Sensibilidade de Ativação.	98
Figura 32 – Processo de ativação das ações.	99
Figura 33 – Filtros diferentes para o ND-NCD.	100
Figura 34 – Figura ilustrando o sensor laser e os ângulos de referência utilizados pelo reconhecedor de estados.	100

LISTA DE ALGORITMOS

Algoritmo 1 – D* Lite Algorithm	42
Algoritmo 2 – Algoritmo MMAS	58
Algoritmo 3 – Algoritmo MMAS-DPP	60

LISTA DE TABELAS

Tabela 1 – Trabalhos relacionados a algoritmos genéticos aplicados ao problema de planejamento de trajetórias de robôs móveis.	35
Tabela 2 – Trabalhos relacionados a algoritmos de otimização por colônia de formigas aplicados ao problema de planejamento de trajetórias de robôs móveis. . . .	38
Tabela 3 – Trabalhos relacionados a algoritmos genéticos aplicados ao problema de planejamento de trajetórias de robôs móveis.	43
Tabela 4 – Análise do tamanho da população.	64
Tabela 5 – Análise da taxa de mutação.	65
Tabela 6 – Análise da taxa de proporção elitista.	65
Tabela 7 – Análise do operador de reparo.	66
Tabela 8 – Resultados dos experimentos com diferentes objetivos.	66
Tabela 9 – Análise da técnica de cruzamento.	68
Tabela 10 – Análise do tamanho da população.	68
Tabela 11 – Análise da taxa de elitismo com tamanho de população 1000.	69
Tabela 12 – Análise da taxa de elitismo com tamanho da população 2000.	69
Tabela 13 – Resultados obtidos por todos os pontos de início e destino indicados na Figura 18.	69
Tabela 14 – Análise da técnica de cruzamento.	70
Tabela 15 – Análise do impacto do tamanho inicial sobre a sequência do indivíduo em crescimento.	72
Tabela 16 – Análise da taxa de mutação.	72
Tabela 17 – Análise do tamanho da população.	72
Tabela 18 – Análise do método de seleção.	73
Tabela 19 – Diferentes posições de início e destino.	73
Tabela 20 – Resultados obtidos pelos algoritmos A*, C* e AG para diferentes posições de início e destino, mostrado na Figura 19.	74
Tabela 21 – Resultados obtidos pelo AG nos mapas mostrados nas Figuras 21 e 22. . . .	75
Tabela 22 – Resultados obtidos pelos algoritmos A* e C* nos mapas apresentados nas Figuras 21 e 22.	76
Tabela 23 – Análise do número de iterações no critério de parada.	78
Tabela 24 – Análise do número de formigas.	79
Tabela 25 – Resultados dos algoritmos A*, AG e MMAS para as diferentes posições de início e objetivo mostradas na Figura 5.	79

Tabela 26 – Resultados dos algoritmos A*, AG e MMAS para os mapas 2 e 3 (Figuras 21 e 22).	80
Tabela 27 – Análise do número de formigas.	81
Tabela 28 – Taxa de evaporação.	82
Tabela 29 – Quantidade mínima de feromônio.	82
Tabela 30 – Resultados obtidos pelos algoritmos D* Lite e MMAS-DDP no ambiente estático.	83
Tabela 31 – Ambiente dinâmico - Suavização.	84
Tabela 32 – Ambiente dinâmico - Remoção de arestas.	84
Tabela 33 – Comparação entre os resultados obtidos pelos algoritmos D* Lite e MMAS-DPP.	88
Tabela 34 – Ambiente dinâmico - Mudança de objetivo.	88

LISTA DE ABREVIATURAS E SIGLAS

A*	Algoritmo de Busca Informada
ACO	<i>Ant Colony Optimization</i>
AG	Algoritmos Genéticos
C*	Algoritmo de Busca com Estimativa Realista de Custo
DN-NCD	Método de Detecção de Novidade com NCD
FSM	<i>Finite State Machine</i>
ICMC	Instituto de Ciências Matemáticas e de Computação
LRM	Laboratório de Robótica Móvel
MMAS	<i>Max-Min Ant System</i>
MMAS-DPP	<i>Max-Min Ant System for Dynamic Path Planning</i>
MMAS-SPP	<i>Max-Min Ant System for Static Path Planning</i>
NCD	<i>Normalized Compression Distance</i>
Neuro-FSM	<i>Neural Finite-State Automata</i>
NID	<i>Normalized Information Distance</i>
PSO	Particle Swarm Optimization
RNA	Rede Neural Artificial
USP	Universidade de São Paulo

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Objetivo	25
1.2	Principais Contribuições	26
1.3	Organização	26
2	O PROBLEMA DE PLANEJAMENTO DE TRAJETÓRIAS DE ROBÔS MÓVEIS AUTÔNOMOS	27
2.1	Planejamento de Trajetórias e Mapas	27
2.2	Trabalhos relacionados	31
2.2.1	<i>Algoritmos Genéticos</i>	31
2.2.2	<i>Algoritmos de Otimização por Colônia de Formigas</i>	36
2.2.3	<i>Outros algoritmos</i>	38
2.2.3.1	<i>Abordagem baseada em fronteira</i>	38
2.2.3.2	<i>Algoritmos Bug</i>	39
2.2.3.3	<i>Campos potenciais artificiais</i>	39
2.2.3.4	<i>Algoritmo A*</i>	40
2.2.3.5	<i>Algoritmo D* Lite</i>	40
2.3	Considerações Finais	42
3	ALGORITMOS PROPOSTOS PARA PLANEJAMENTO DE TRAJETÓRIAS	45
3.1	Algoritmos Genéticos	46
3.1.1	<i>Algoritmo Genético I</i>	46
3.1.2	<i>Algoritmo Genético II</i>	49
3.1.3	<i>Algoritmo Genético III</i>	53
3.2	Algoritmos de Otimização por Colônia de Formigas	57
3.2.1	<i>Algoritmo Max-Min Ant System for Static Path Planning</i>	57
3.2.2	<i>Algoritmo Max-Min Ant System for Dynamic Path Planning</i>	60
3.2.3	<i>Considerações Finais</i>	61
4	RESULTADOS	63
4.1	Algoritmos Genéticos	63
4.1.1	<i>Resultados do AG I</i>	63
4.1.2	<i>Resultados do AG II</i>	67

4.1.3	<i>Resultados do AG III</i>	70
4.2	Algoritmos de Otimização por Colônia de Formiga	76
4.2.1	<i>Algoritmo Max-Min Ant System for Static Path Planning</i>	77
4.2.2	<i>Algoritmo Max-Min Ant System for Dynamic Path Planning</i>	80
4.2.3	<i>Considerações Finais</i>	90
5	NAVEGAÇÃO TOPOLÓGICA	93
5.1	Navegação com Redes Neurais Artificiais	93
5.2	Navegação com Distância Normalizada de Compressão	96
5.3	Navegação com Reconhecimento de Estados baseado em Ângulos de Referência	98
5.4	Considerações Finais	101
6	CONCLUSÕES E TRABALHOS FUTUROS	103
6.1	Contribuições	104
6.2	Limitações e Dificuldades	104
6.3	Trabalhos Futuros	105
6.4	Lista de Publicações	105
6.5	Artigo Submetido	106
	REFERÊNCIAS	107

INTRODUÇÃO

Planejamento e execução de trajetórias são tarefas de grande importância para robôs móveis autônomos, pois permitem que a rota seja definida e que a navegação seja realizada de forma autônoma. Nesse processo, primeiro é necessário definir o caminho de um ponto inicial para um destino. Este processo pode ser realizado sem o conhecimento prévio das rotas disponíveis ou do mapa do ambiente (planejamento exploratório) ou através do uso de representações previamente disponibilizadas e conhecidas acerca do ambiente (planejamento baseado em mapas). A tarefa de navegação pode ser descrita por um conjunto de comportamentos, os quais indicam as principais ações a serem executadas para alcançar o objetivo final. Uma vez definida a trajetória, um ou mais robôs poderão usar essas ações e comportamentos para alcançar o destino.

Segundo [Nelson, Barlow e Doitsidis \(2009\)](#), projetar um modelo matemático para um planejamento ótimo ou quase ótimo e uma estratégia de controle de navegação de um robô são tarefas complexas. Métodos baseados em otimização heurística, como A*, D*, grafo de visibilidade ([CHOSSET *et al.*, 2005](#)), fornecem boas soluções para o planejamento de trajetórias de um robô. Todavia, algoritmos como esses requerem uma localização precisa do robô durante toda a execução do caminho e um mapa previamente fornecido e bem definido.

O foco da maioria dos trabalhos encontrados na literatura que utilizam otimização heurística para planejamento de trajetórias de robôs móveis está na busca da trajetória ótima e não se preocupam na executabilidade dessa trajetória em um ambiente robótico. O robô que vai navegar na trajetória, o sistema de localização utilizado, os sensores para identificar obstáculos no caminho e como o robô vai navegar no ambiente são questões que não são discutidas em muitos trabalhos, que acabam focando apenas na tarefa de planejamento.

Em geral, esses trabalhos adotam a representação de mapa métrico, que contém informação detalhada do ambiente, e a trajetória encontrada é composta por um conjunto de coordenadas que o robô deve seguir precisamente para alcançar o objetivo. A execução da trajetória nesse tipo

de representação exige o uso de algoritmos de localização precisa e de um mapa bem definido, que pode tornar a navegação impraticável em um ambiente real, conforme as restrições impostas em termos de hardware e software.

O problema de planejamento exploratório tratado nesta tese consiste em definir um caminho para que o robô deixe sua posição inicial e alcance seu objetivo sem conhecimento prévio do ambiente. Uma abordagem híbrida para planejamento exploratório de trajetórias e controle de navegação de robôs móveis autônomos é proposta. A abordagem é dita híbrida porque a etapa de planejamento está vinculada à navegação da trajetória encontrada.

No planejamento exploratório, os robôs descobrem o ambiente à medida que buscam um caminho satisfatório para o destino, ou seja, não possuem conhecimento prévio do ambiente. As etapas de planejamento e navegação são baseadas na representação de mapa topológico adotado. Essa representação permite que o robô navegue no ambiente sem necessidade de algoritmos robustos de localização, pois apenas o reconhecimento de locais estratégicos do ambiente é necessário. Na etapa de planejamento foram investigados os Algoritmos Genéticos e Algoritmos de Otimização por Colônia de Formigas. Durante a busca pela melhor trajetória, os robôs navegam em uma representação de grafo, que considera os requisitos para a navegação no ambiente real. Depois, a navegação da melhor trajetória é executada em simuladores robóticos. Entretanto, a abordagem permite que a etapa de planejamento seja realizada em ambientes simulados ou reais.

Na etapa de navegação da trajetória foi estudado o sistema de navegação topológico *Neural Finite-State Automata* (Neuro-FSM) proposto por Sales, Osório e Wolf (2011), Sales (2012), Sales *et al.* (2012), Sales *et al.* (2014). Esse sistema foi desenvolvido para fornecer o controle e sistema de navegação para robôs móveis baseado em mapas topológicos previamente conhecidos e caminhos previamente definidos, que são especificados com base em mapas topológicos. Na abordagem Neuro-FSM, o mapa, o caminhos e as ações do robô foram definidos previamente, contudo, a abordagem desta proposta é baseada em mapas topológicos e comportamentos locais (fortemente reativos), fazendo com que o sistema não dependa de uma localização muito precisa ou de mapas métricos detalhados. O sistema neural (Rede Neural Artificial (RNA)) é responsável por reconhecer o estado atual e as mudanças de estado. O sistema *Finite State Machine* (FSM) é um sequenciador que usa o estado atual para associar uma ação específica ao contexto atual, e, quando o estado atual muda, uma nova ação é selecionada para ser executada. Por exemplo, se o robô está seguindo um caminho em um corredor, o estado atual é definido como “corredor” e a ação é mover-se no corredor. Quando o robô chega no fim do corredor, ou outra situação nova representando um contexto diferente (por exemplo, curvas, cruzamentos), o robô deve trocar a ação para outro comportamento que o permita lidar com a nova situação. A abordagem do Neuro-FSM resulta em um sistema robusto e de bom desempenho em plataformas de hardware mais simples. Por outro lado, essa abordagem não fornece uma estratégia para a exploração do ambiente e nem de planejamento de trajetórias.

Considerando o acima exposto, esta tese tem como motivação aplicações como por exemplo, os problemas do tipo busca e resgate. Neste tipo de aplicação, e no contexto deste trabalho, assume-se que é possível saber inicialmente a posição aproximada do robô e da vítima a ser resgatada, porém não há conhecimento sobre o mapa do ambiente previamente disponibilizado e nem a trajetória exata que leva até a vítima. Portanto, o desafio proposto é a realização de uma busca pelo caminho até o destino (vítima), através de uma abordagem de planejamento exploratório, sem a necessidade de criar um mapa detalhado ou de manter uma localização precisa, como seria o caso de outras abordagens mais complexas como o *SLAM Simultaneous Localization and Mapping* (ROMERO *et al.*, 2014). Além disso, também é importante que a solução encontrada seja eficiente, do ponto de vista do caminho (curto) e da realização da busca (rápida), que pode inclusive ser realizada com o uso de múltiplos robôs. Esse tipo de aplicação se enquadra em uma categoria de problemas de planejamento e navegação de robôs móveis até uma posição de destino, sem um conhecimento prévio do mapa.

1.1 Objetivo

Motivados pelas questões levantadas anteriormente e pelo trabalho proposto por Sales, Osório e Wolf (2011), Sales (2012), Sales *et al.* (2012), Sales *et al.* (2014), o objetivo principal deste projeto de doutorado é fornecer uma abordagem híbrida para planejamento e controle de navegação de robôs móveis em ambiente *indoor*, representado por um mapa topológico. Nessa abordagem, a busca pela trajetória é comprometida com a navegação dessa trajetória no contexto da robótica móvel e considerando todas as questões inerentes a ele. Assim, o robô é capaz de planejar um caminho ótimo ou quase ótimo entre sua localização inicial e seu objetivo e de realizar a navegação de forma segura.

Os objetivos específicos deste projeto são:

- Desenvolver uma abordagem evolutiva para encontrar e planejar a trajetória do robô sem necessidade de informação prévia detalhada do ambiente;
- Desenvolver um método para controlar a execução da trajetória do robô, baseado em comportamentos locais reativos;
- Validar o método proposto em ambientes simulados e reais;

Este projeto de doutorado foi desenvolvido no Laboratório de Robótica Móvel (LRM), localizado no Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP) e junto a *University of Kent* (Inglaterra, Reino Unido) na *School of Computing*, com a colaboração dos professores Dr. Fernando Otero e Dr. Colin Johnson.

1.2 Principais Contribuições

A originalidade dessa tese consiste em um planejamento de trajetória comprometido com sua execução em ambientes com robôs móveis e na definição da trajetória como um conjunto de ações (comportamentos reativos) que levam o robô da posição inicial ao objetivo. Essa definição permite que o planejamento e a navegação da trajetória sejam realizados sem a necessidade de uma maior precisão na representação do ambiente e na localização do robô.

Além disso, outras contribuições e resultados relevantes foram fornecidos:

- Desenvolvimento de algoritmos genéticos para o planejamento exploratório de trajetórias em mapas do tipo labirinto, no qual os robôs não tem conhecimento prévio do ambiente;
- Desenvolvimento de algoritmos de otimização por colônia de formigas no mesmo contexto, em ambientes estáticos e dinâmicos;
- Desenvolvimento de algoritmos de reconhecimento de padrões para identificar o estado atual do robô durante a execução da navegação.

1.3 Organização

Este documento está organizado da seguinte forma:

Capítulo 2: o problema de planejamento de trajetórias de robôs móveis autônomos tratado por essa pesquisa é apresentado, bem como os trabalhos relacionados;

Capítulo 3: os algoritmos propostos para planejamento de trajetórias são descritos;

Capítulo 4: os resultados dos algoritmos investigados são apresentados e discutidos;

Capítulo 5: são apresentados os métodos de navegação, que permitem que o robô percorra a trajetória encontrada e alcance o objetivo de forma segura;

Capítulo 6: são apresentadas as conclusões e trabalhos futuros.

O PROBLEMA DE PLANEJAMENTO DE TRAJETÓRIAS DE ROBÔS MÓVEIS AUTÔNOMOS

O problema de planejamento de trajetória em robótica consiste em determinar automaticamente o caminho que o robô deve percorrer, partindo de uma configuração inicial para uma final, sem colidir com obstáculos presentes no ambiente. Neste capítulo, esse problema é apresentado juntamente com trabalhos relacionados a esse tema. Na Seção 2.1 o problema é apresentado e na Seção 2.2 trabalhos recentes relacionados a esse problema que utilizam algoritmos genéticos e algoritmos de otimização por colônia de formigas são descritos.

2.1 Planejamento de Trajetórias e Mapas

O problema de planejamento de trajetórias em robótica consiste em determinar automaticamente o caminho que o robô deve percorrer, partindo de uma posição inicial para uma final, sem colidir com obstáculos presentes no ambiente (ADORNO; BORGES, 2014). A maioria dos algoritmos para planejamento de trajetórias assume que o robô tem conhecimento completo do ambiente no qual se encontra. No problema de planejamento investigado nesta tese, uma rota de um local inicial para um objetivo deve ser encontrada com o uso de vários robôs. Os robôs não possuem conhecimento prévio do ambiente, cada robô conhece apenas as coordenadas aproximadas da posição atual e final e os robôs devem alcançar o objetivo o mais rápido possível. Por isso, esse é um problema de planejamento exploratório, em que os robôs capturam informação do ambiente ao mesmo tempo que planejam a rota para alcançar o objetivo. A Figura 1 ilustra o cenário neste problema, em que cada robô conhece apenas seu ponto inicial S e o objetivo no ponto G .

A exploração nesse contexto é uma exploração direcionada, na qual o conhecimento local

Figura 1 – O robô conhece as posições inicial e final aproximadas, mas não conhece o caminho para alcançar o objetivo.

Fonte: Elaborada pelo autor.

do ambiente é adquirido pelo uso de sensores para planejar uma rota para alcançar o objetivo. Portanto, cada robô conhece as características do ambiente no seu entorno. Isso difere da tarefa de mapeamento completo, na qual o robô explora todo o ambiente para adquirir um modelo espacial do ambiente do robô (THRUN, 2002).

Na navegação baseada em mapas, o robô pode usar os dados coletados através dos sensores para se localizar e controlar o seu deslocamento no ambiente em que está inserido. Segundo Costa e Selvatici (2014), mapas são representações do ambiente, utilizadas para localização em várias atividades humanas, como navegação e geografia. Na robótica, os mapas podem ser utilizados para melhorar a estimativa de localização dos robôs, confrontando os dados dos sensores com os elementos representados no mapa em determinadas posições.

O tipo ideal de mapa depende da tarefa do robô e do ambiente em que ele está inserido. Também pode variar de acordo com as características do robô, tais como tipos de sensores, forma de se mover e capacidades computacionais para armazenamento e manipulação da informação.

Em geral, a nomenclatura dos tipos de mapas utilizados em robótica varia de acordo com os elementos usados para descrever o ambiente. Os tipos básicos de mapas podem variar em função do nível de abstração dos elementos utilizados na sua construção (COSTA; SELVATICI, 2014). Os dois principais tipos de mapas são os mapas métricos e os mapas topológicos, que possuem um nível mais baixo e um nível mais alto de abstração, respectivamente.

O principal objetivo dos mapas métricos é fornecer informações espaciais detalhadas do ambiente, de onde podem ser inferidos a forma e tamanho dos objetos e os limites das áreas livres para a navegação (COSTA; SELVATICI, 2014). Os principais exemplos de mapas métricos são mapas geométricos e grades de ocupação. O mapa geométrico representa características do ambiente em espaço contínuo (SIEGWART; NOURBAKHS, 2004). No mapa grade de ocupação, o mapa é dividido em células regulares que representam o estado da ocupação do

Figura 2 – Exemplo de uma representação de mapa grade de ocupação. A figura (b) apresenta a região em destaque ampliada, para ilustrar as células ocupadas e não ocupadas.

Fonte: Elaborada pelo autor.

local de interesse, indicando os objetos e limitando as áreas onde o robô pode navegar (COSTA; SELVATICI, 2014). Um exemplo de mapa grade de ocupação é ilustrado na Figura 2.

Mapas topológicos representam localidades conectadas por algum tipo de relacionamento, que pode ser adjacência ou conectividade entre elas. Nesse caso, as localidades são regiões no espaço que possuem a mesma característica, não sendo necessário definir uma posição específica. Assim, as informações contidas no mapa topológico são menos detalhadas e distribuem-se de forma dispersa, concentrando-se apenas em regiões de interesse, de acordo com a tarefa do robô. Por isso, a seleção da informação que será registrada deve ser criteriosa. O principal objetivo desse tipo de mapa é obter uma estrutura relacional, capaz de fornecer informações sobre os locais que o robô pode acessar.

A Figura 3 ilustra um exemplo de ambiente adotado nesta pesquisa. O ambiente é estruturado, tipo um labirinto, com corredores e bifurcações. A representação topológica utiliza um grafo, onde os pontos característicos do ambiente, como curvas e bifurcações, que representam pontos de decisão do robô, são os nós do grafo e as arestas indicam a navegabilidade entre esses nós.

Neste projeto, foi usado um sistema para gerar os mapas em formato de figura, no qual, dadas a largura e a altura do mapa, são criadas as paredes e alguns pontos são escolhidos para gerar as bifurcações. Após a geração do mapa, um sistema baseado em OpenCV (*Open Source Computer Vision Library*: <<http://www.opencv.org>>) é utilizado para obter o grafo correspondente.

Em certas aplicações, a representação baseada em mapas topológicos pode ser mais viável de ser executada em ambientes reais do que mapas em grade. Por exemplo, se o robô está se movendo em uma estrada e identifica uma bifurcação, ele pode tomar uma decisão como seguir em frente ou fazer uma curva. Em uma representação em grade, normalmente o caminho é planejado com base em coordenadas, sendo que o robô precisa conhecê-las e segui-las

Figura 3 – Exemplo de mapa e sua representação topológica em forma de grafo. Cada nó representa um local característico do ambiente e as arestas são a conectividade entre eles.

Fonte: Elaborada pelo autor.

precisamente para executar com sucesso sua tarefa. Em um mapa topológico, a localização do robô é mais flexível e pode se basear apenas na identificação de bifurcações e, assim, decidir que ação executar. Além disso, um algoritmo de localização pode consumir muito recurso computacional. Assim, uma exploração baseada em mapa topológico é adotada neste trabalho na qual o robô precisa apenas saber reconhecer algumas pontos característicos do ambiente para tomar uma decisão e ter uma localização aproximada do robô e do objetivo.

Neste trabalho de doutorado, um método exploratório para planejamento de trajetórias de robôs móveis autônomos foi proposto para o cenário descrito acima. Devido à adoção da representação de mapa topológico, a representação do caminho não será uma sequência de pontos específicos que o robô deve visitar, mas sim um plano global de ações necessárias para alcançar o objetivo. As ações são comportamentos reativos locais, como virar à esquerda e seguir em frente. As abordagens de exploração e planejamento de trajetórias que utilizam uma população de robôs podem explorar melhor o ambiente e encontrar um bom caminho para o ponto de destino de forma mais rápida do que abordagens que usam apenas um robô.

A navegação topológica adotada nesta pesquisa é descrita no Capítulo 5

2.2 Trabalhos relacionados

Nesta seção, são apresentados trabalhos recentes sobre planejamento de trajetórias que utilizam Algoritmos Genéticos (AG) (MITCHELL, 1995; ATMAR, 1992) e Algoritmos de Otimização por Colônia de Formigas (*Ant Colony Optimization* (ACO)) (DORIGO; STÜTZLE, 2004).

2.2.1 Algoritmos Genéticos

O conhecimento do domínio foi agregado a um AG para planejamento de trajetória de um robô móvel em ambientes estáticos e dinâmicos no trabalho de Hu e Yang (2004). O ambiente foi representado por grades enumeradas ordenadas. O cromossomo representa um caminho, composto por uma sequência de grades enumeradas, começando da posição inicial e finalizando na posição alvo. O conhecimento do domínio foi usado por operadores especializados, combinando uma técnica de busca local, para melhorar um caminho. A avaliação considerou caminhos viáveis (livres de colisão) e caminhos inviáveis. O valor da função de aptidão foi definido como a soma das distâncias de cada segmento de reta e, se um caminho inviável foi gerado, uma penalidade era adicionada. O algoritmo também foi usado em ambientes dinâmicos. Neste caso, quando um obstáculo era adicionado ao mapa, um novo caminho deveria ser encontrado pelo algoritmo genético. Entretanto, os autores não explicam se foram usados sensores para perceber e evitar obstáculos, nem métodos de localização e de controle de navegação. Os autores afirmam que o método foi avaliado em simulação e demonstrou efetividade e eficiência.

Sedighi *et al.* (2004) desenvolveu um AG para planejamento de trajetória de um robô móvel, evitando obstáculos locais em ambientes estáticos. O objetivo foi minimizar o tamanho do caminho e o número de voltas. O ambiente foi modelado como uma grade dividida em linhas e colunas. A estrutura do cromossomo foi composta por quatro variáveis: *path-flag*, *path-location*, *path-direction* and *path-switch*. O robô possui dois movimentos possíveis: por linha e por coluna. A variável *path-flag* indica os movimentos possíveis; a variável *path-location* refere-se à posição do robô; a variável *path-direction* indica a direção que o robô deve girar para se dirigir ao próximo nó; e a variável *path-switch* permite que o robô alterne entre um movimento de linha e coluna em um único caminho. O valor da função de aptidão considera o tamanho do caminho, o número de voltas e de colisões. De acordo com os autores, os experimentos mostraram que o algoritmo genético proposto trabalha melhor que outros algoritmos encontrados na literatura. Esse trabalho apresenta um algoritmo que encontra uma trajetória para o robô, porém não explica como o robô navega no ambiente, principalmente quando lida com imprecisão e incertezas do mundo real.

Yun, Ganapathy e Chong (2010) propuseram um algoritmo genético para planejamento de trajetória ótimo para robôs móveis. O ambiente foi representado por um modelo de grade enumerada, em que cada unidade de grade enumerada é um nó. Assim, o caminho foi definido

como uma sequência de nós, que representa o indivíduo. O operador cruzamento de ponto único aleatório foi usado para gerar novos indivíduos. No operador de mutação, um nó é selecionado aleatoriamente, e seu valor é modificado. Depois da aplicação de cada operador genético, um mecanismo de distinção foi usado para corrigir caminhos inviáveis. Um operador de refinamento foi proposto pelos autores para reduzir o tamanho do caminho, baseado no princípio triangular. Esse princípio estabelece que se três nós formam um triângulo retângulo, o nó que forma 90° com os demais é excluído. No operador de exclusão, se dois nós podem ser conectados sem cruzar com qualquer obstáculo, os nós intermediários entre eles podem ser excluídos. O valor da função de aptidão foi definido como o inverso da distância total do caminho. A efetividade do algoritmo genético proposto foi demonstrada em simulação e com um robô real. Na simulação, apenas o desempenho do AG foi avaliado. No experimento real, o controle da navegação se baseia na leitura dos sensores e em codificadores (*encoders*) que o robô possui para identificar suas coordenadas e ângulo.

Um algoritmo genético aplicado ao problema de planejamento de trajetória em ambientes dinâmicos foi proposto por [Tuncer e Yildirim \(2012\)](#). O ambiente foi representado por um modelo de grades enumeradas em ordem. Um cromossomo foi representado por uma sequência de inteiros, em que cada inteiro representa uma célula da grade enumerada no ambiente. A população inicial foi gerada aleatoriamente, porém reparada se a informação contida no cromossomo correspondia a uma interseção com um obstáculo. O cruzamento de ponto único foi aplicado em dois pais para gerar dois novos indivíduos. A função de aptidão foi definida como a soma das distâncias entre cada nó em um caminho. Se houvesse um obstáculo na direção do robô, uma penalidade devia ser adicionada ao valor da função objetivo. No operador de mutação, um nó é escolhido e todos os nós livres, que não intersectam um obstáculo, próximos a este nó são avaliados. O nó livre que gerar um caminho com menor valor de função de aptidão é escolhido. Segundo os autores, o algoritmo foi testado em ambientes dinâmicos. Eles afirmam que os experimentos mostraram que o método converge mais rápido que outros métodos na literatura.

Algoritmos genéticos também têm sido aplicados ao problema de planejamento de trajetória para múltiplos robôs, como o trabalho de [Mohanta, Parhi e Patel \(2011\)](#). Ele propôs uma abordagem que combina algoritmo genético com modelo de Redes de Petri ([PETRI, 1962](#)) para planejamento de trajetória de múltiplos robôs, que buscam por vários alvos na presença de obstáculos. A população inicial com tamanho pré-definido é criada a partir das distâncias obtidas pelos sensores. Cada cromossomo é composto por cinco elementos: distância ao obstáculo da frente; distância ao obstáculo à esquerda; distância ao obstáculo à direita, ângulo da posição atual em relação à posição alvo, sinal de conversão para sentido (horário, reto ou anti-horário), baseado na direção do ângulo da posição atual. Esses valores são convertidos para codificação binária. A função de aptidão foi definida para evitar obstáculos, baseada nas melhores distâncias do conjunto de distâncias instantâneas aos obstáculos, em relação à posição inicial, ao ângulo e direção alvos.

No operador de cruzamento, dois pais são combinados para aplicar o cruzamento de ponto único e gerar dois descendentes. No operador de mutação, um número aleatório é atribuído ao local e uma direção aleatória é baseada na direção referência. O comportamento de evitar obstáculos e colisões entre robôs para a navegação segura do robô foi incorporado pelo modelo de Redes Petri. Esse modelo é composto por sete tarefas: espera pelo sinal inicial; mover-se, evitando obstáculos; detectar conflitos; negociação; verificar conflitos e executar movimentos; esperar. No início, o robô não possui qualquer conhecimento do ambiente (primeira tarefa). Depois de receber um comando para iniciar, a navegação é realizada, controlada pelas tarefas definidas pelo modelo de Redes Petri. Os resultados dos experimentos em simulação e em ambiente real mostraram que o controlador proposto é capaz de resolver o problema de navegação de múltiplos robôs, sem que haja colisões entre os robôs ou com obstáculos.

Kala (2012) propôs uma solução para planejamento de rotas com múltiplos robôs que utiliza um algoritmo genético mestre e escravo em um mapa labirinto. O algoritmo escravo é responsável por encontrar o caminho ótimo para cada robô. O algoritmo mestre preocupa-se com caminhos ótimos globais. No algoritmo escravo, o cromossomo foi representado como uma sequência de valores inteiros. Cada valor representa um movimento do robô, que pode ser *esquerda*, *direita*, *cima*, *baixo*. Cruzamento uniforme e mutação uniforme foram os operadores usados no algoritmo. A função de aptidão foi calculada como o tamanho total do caminho. Se o robô não alcança o objetivo, uma penalidade foi adicionada, proporcional à distância entre o último ponto alcançado pelo robô e a posição alvo. Um fator que se refere a colisões entre robôs foi adicionado à função de aptidão, com o objetivo de alcançar a solução global.

No algoritmo mestre, o indivíduo foi definido como um conjunto de ponteiros, onde cada ponteiro se refere a algum indivíduo do algoritmo escravo. Similar ao algoritmo escravo, cruzamento uniforme foi utilizado. Na mutação, um ponteiro existente é excluído e um novo ponteiro aponta para algum outro indivíduo na população escrava. O valor da função de aptidão foi definido pelo tempo médio percorrido por cada robô. Uma penalidade foi adicionada caso o robô não alcance o objetivo. O método proposto foi executado em vários cenários e em cada um deles o algoritmo encontrou uma trajetória livre de colisão para todos os robôs da posição inicial para o objetivo. A simulação foi feita usando Java Applet baseada em GUI, mas os autores não explicaram se foram usados sensores e como robô lida com imprecisão.

Qu, Xing e Alexander (2013) apresentaram um algoritmo genético co-evolutivo para planejamento de trajetória global de múltiplos robôs. O termo co-evolutivo refere-se ao fato de que um AG foi proposto para resolver o problema de planejamento de trajetórias para um único robô móvel e, baseado nesse AG, um mecanismo evolutivo foi utilizado para resolver o mesmo problema para múltiplos robôs. O ambiente é representado em forma de grade, em que cada célula é enumerada em ordem crescente. O indivíduo genético é modelado como uma sequência de números, que representa uma sequência de células no ambiente. Na seleção dos indivíduos para reprodução, o método da roleta foi utilizado, seguido da seleção elitista para

manter os melhores indivíduos na população. No operador de mutação, um gene é substituído por um novo gene, não incluso no indivíduo. Um operador de modificação foi proposto para alterar um caminho de forma a evitar colisão com obstáculos. Se dois nós não-adjacentes podem ser conectados sem colidir com qualquer obstáculo, os nós intermediários entre eles são excluídos e a mesma quantidade de nós é aleatoriamente adicionada ao segmento de linha que conecta os dois nós não-adjacentes. Esse procedimento é realizado para manter o tamanho do gene. Nesse trabalho, a função de aptidão é uma combinação de duas funções: uma que avalia o caminho quanto à viabilidade e outra quanto à inviabilidade. O cruzamento de dois pontos foi utilizado. Na co-evolução, uma função de cruzamento foi definida para considerar a interferência do cruzamento entre os caminhos e a cooperação entre robôs.

Três experimentos foram realizados para demonstrar a viabilidade e efetividade dos algoritmos propostos. O primeiro simula o planejamento de trajetória para um robô móvel usando o AG proposto; o segundo compara o desempenho do AG proposto com o AG tradicional e com *Ant Colony Optimization - ACO* (DORIGO; STÜTZLE, 2004); e o terceiro simula o planejamento global de trajetória para múltiplos robôs, utilizando a co-evolução. No primeiro experimento, o algoritmo foi testado em três ambientes diferentes nos quais foi capaz de encontrar um caminho ótimo, livre de colisão. No segundo experimento, o AG proposto foi melhor em evitar o problema de ótimo local e baixa taxa de convergência, comparado com o AG convencional. Além disso, o método apresentou menor tempo de execução e encontrou soluções menores que o AG convencional. Em relação ao ACO, os testes foram feitos em três tipos de ambientes. Os resultados mostraram que o AG melhorado consome menor tempo de execução que o ACO e apresentou taxa de caminhos ótimos maior que o ACO. Em um dos ambientes, que possui muitas esquinas, o ACO encontrou uma solução menor que o AG proposto. Isso ocorre porque o AG proposto não considera apenas o menor caminho, mas também a segurança do robô em relação à distância aos obstáculos. Nos experimentos com múltiplos robôs, foram utilizados três robôs móveis e obstáculos estáticos, nos quais o algoritmo foi capaz de encontrar soluções ótimas ou quase ótimas, livres de colisão. Os autores não citam qual método foi utilizado para realizar a navegação de fato.

Em geral, os métodos tradicionais de planejamento de trajetória consideram apenas um dos objetivos desse problema, que, na maioria dos casos, é obter uma solução que apresente o menor caminho. Davoodi *et al.* (2013) consideraram dois objetivos para encontrar a solução ótima no problema de planejamento de trajetória: o tamanho e a folga do caminho em relação à distância do robô aos obstáculos. Para isso, eles propuseram dois modelos de otimização multi-objetivo, um para cada objetivo. Um algoritmo genético com o framework NSGA-II (DEB *et al.*, 2000) foi usado para resolver os problemas nesses modelos. O cromossomo é representado por uma lista ligada, em que cada nó armazena as coordenadas x e y de uma célula e um ponteiro para o próximo nó. Para a inicialização dos cromossomos, as primeiras k células livres são selecionadas aleatoriamente e um caminho é traçado entre elas. Para o primeiro objetivo, a função de aptidão é definida como o número de células visitadas entre a origem e o destino. Para

o segundo objetivo, é suficiente que o peso de uma célula seja maior ou igual a 1, em que o peso representa a distância entre a célula e o obstáculo mais próximo.

O operador seleção por torneio restrito foi utilizado para selecionar os indivíduos para reprodução. O cruzamento de ponto único é utilizado, no qual o ponto escolhido é uma célula comum entre os dois indivíduos pais. Se não há célula comum, os novos indivíduos são iguais aos pais. No operador de mutação, duas células do caminho são selecionadas aleatoriamente e o subcaminho entre elas é removido e substituído por um novo, gerado aleatoriamente. Além desses operadores padrões, os autores propuseram novos operadores: refinador curto, refinador de folga e refinador de auto-interseção. O primeiro, tenta melhorar o tamanho do caminho ligando duas células selecionadas com uma linha reta aproximada. Para aumentar a distância mínima entre a trajetória e os obstáculos, o operador que refina a folga foi proposto. Nesse operador, primeiro uma busca linear é realizada para encontrar um subcaminho com peso mínimo. Em seguida, uma célula vizinha aleatória com folga maior é localizada e conectada ao subcaminho. O refinador de auto-interseção é usado para remover ciclos.

O objetivo do primeiro modelo foi avaliado e comparado com os algoritmos Particle Swarm Optimization (PSO) (KENNEDY; EBERHART, 1995) e o A* (RUSSELL; NORVIG, 2009). O algoritmo A* encontra o menor caminho, mas consome mais tempo. O PSO foi mais rápido que o algoritmo proposto, mas as soluções encontradas pelo último foram consideravelmente melhores. Nos experimentos com o objetivo do segundo modelo, o algoritmo foi capaz de encontrar um conjunto de diversas soluções ótimas em vários cenários.

A Tabela 1 resume os trabalhos apresentados ressaltando as características de representação do ambiente (se usa grade de ocupação), se lida com ambientes dinâmicos, se descreve método de navegação para a execução da trajetória encontrada e se foi realizado experimento em robô real.

Tabela 1 – Trabalhos relacionados a algoritmos genéticos aplicados ao problema de planejamento de trajetórias de robôs móveis.

Referência	Representação do ambiente	Ambiente dinâmico	Método de navegação	Experimento Real
(HU; YANG, 2004)	Grade	Sim	Não	Não
(SEDIGHI <i>et al.</i> , 2004)	Grade	Não	Não	Não
(YUN; GANAPATHY; CHONG, 2010)	Grade	Não	Sim	Sim
(TUNCER; YILDIRIM, 2012)	Grade	Sim	Não	Não
(MOHANTA; PARHI; PATEL, 2011)	-	Sim	Sim	Sim
(KALA, 2012)	-	Não	Não	Não
(QU; XING; ALEXANDER, 2013)	Grade	Não	Não	Não
(DAVOODI <i>et al.</i> , 2013)	Grade	Não	Não	Não

Fonte: Dados da pesquisa.

2.2.2 Algoritmos de Otimização por Colônia de Formigas

Algoritmo de Otimização por Colônias de Formigas é uma técnica bem conhecida de busca baseada em modelo (ZLOCHIN *et al.*, 2004). ACO é uma meta-heurística inspirada pelo comportamento de busca por comida das formigas. Formigas constroem seus caminhos para o destino por uma escolha probabilística para se mover para um nó vizinho. A escolha probabilística é baseada no feromônio depositado por outras formigas e na função heurística. Então, as formigas caminham de volta de forma determinística, depositando feromônio no grafo. A quantidade de feromônio depende da qualidade das soluções. Assim, uma formiga artificial tem dois papéis importantes: gerar soluções e atualizar os parâmetros do modelo.

Uma abordagem para resolver o problema de planejamento de trajetórias de robôs móveis baseado na meta-heurística SACO-HM (*Simple Ant Colony Optimization Meta-Heuristic*) é apresentada no trabalho de Garcia *et al.* (2009). Os autores nomearam o algoritmo como SACO-dm, em que d é distância e m é memória. O processo de tomada de decisão é influenciado pela distância entre os nós origem e destino. A memória é usada para as formigas lembrarem os nós visitados, que são marcados temporariamente como obstáculos. No início, todas as formigas são colocadas no nó de origem. A cada iteração, todas as formigas constroem um caminho para o destino, usando uma fórmula probabilística que considera a quantidade de feromônio nos nós vizinhos ao nó atual. Um procedimento para remoção de ciclos é aplicado e o peso de cada rota é calculado. Depois, os passos de evaporação e atualização de feromônios são executados, sendo que no último as formigas voltam para o nó inicial depositando uma quantidade de feromônio, de acordo com a qualidade da solução encontrada. O processo continua até que uma das três condições de parada sejam satisfeitas: número máximo de iterações é atingido, uma solução aceitável foi encontrada ou todas as formigas seguem o mesmo caminho. A seleção do caminho ótimo usa um sistema de inferência nebuloso, que considera o tamanho do caminho e a dificuldade para navegação. O ambiente é discretizado em uma matriz de nós interconectados, sendo que cada nó pode ser marcado como livre ou ocupado por obstáculo. O algoritmo foi avaliado em um simulador proposto pelos autores em situações estáticas e dinâmicas. Quando um novo obstáculo é percebido, um novo caminho é gerado a partir da posição atual do robô. Entretanto, o artigo não explica quais sensores são usados para perceber a presença de um obstáculo.

Um algoritmo de navegação de robôs para ambientes desconhecidos, baseado no princípio do ACO foi proposto por Zhu *et al.* (2011). Dois grupos bidirecionais de formigas exploradoras cooperam um com o outro para encontrar um caminho ótimo local dentro do domínio de visão do robô. No algoritmo de navegação, a posição alvo global é mapeada para um sub-objetivo. Então, os robôs planejam um caminho estático local usando o algoritmo de cooperação de formigas multi-exploradoras. O ambiente é representado por um mapa métrico, como um conjunto de células em grade. De acordo com os autores, o algoritmo de navegação é muito rápido e capaz de gerar caminhos livres de colisão ótimos ou quase-ótimos em ambientes complexos,

desconhecidos e dinâmicos. Embora os autores expliquem que o domínio visual dos robôs é baseado nos seus sensores, essa região foi calculada com base no mapa em grade. Nenhum simulador ou sensor robótico foi apresentado.

O algoritmo de cooperação de formigas multi-exploradoras foi aplicado para interceptar um alvo móvel com o algoritmo de planejamento de trajetórias local rápido na proposta de [Zhu, Hu e Henschen \(2013\)](#). O algoritmo permite que o caminho global seja recomputado quando uma mudança é detectada na trajetória do alvo. O ponto de intercepção é o sub-objetivo, então o algoritmo de cooperação de formigas multi-exploradoras é usado para planejar o caminho de navegação local. Como na proposta de [Zhu et al. \(2011\)](#), o ambiente é representado por um conjunto de células em grade. Os experimentos foram realizados em um simulador proposto pelos autores, nos quais o robô foi capaz de interceptar o alvo em movimento com sucesso.

Um método que combina autômato celular e ACO para planejamento de trajetórias é apresentado por ([IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011](#)). O método cria caminhos livres de colisão para todos os robôs de uma equipe, ao mesmo tempo que mantém sua formação. O autômato celular é uma estrutura em grade que é atualizada pelo ACO para gerar caminhos livres de colisão. Cada célula do autômato pode ser livre, conter um robô, obstáculo ou feromônio. Toda célula evolui seu estado de acordo com seu estado atual e o estado das células vizinhas, aplicando uma regra de transição local. Três regras diferentes foram criadas: regras para evitar colisão, regras para simular as funções de feromônio e regras para controle de formação. Os autores criaram um ambiente de simulação para avaliar o algoritmo e também usaram o simulador Webots ([MICHEL, 2004](#)). No Webots, foram utilizados robôs *e-puck* ([MONDADA et al., 2009](#)), equipados com tipos diferentes de sensores. Os experimentos mostraram que o algoritmo proposto foi efetivo em criar caminhos livres de colisão. Os autores afirmam que o método apresenta baixa complexidade e baixo custo computacional que o torna adequado para implementação em sistemas reais.

Um algoritmo ACO de característica heterogênea para o problema de planejamento de trajetórias foi apresentado por [Yao et al. \(2015\)](#). Os autores definiram dois tipos de formiga: um dedicado à exploração de novos caminhos e outro dedicado à exploração de caminhos promissores. O número de cada tipo de formiga é gerenciado para controlar a taxa de convergência do algoritmo. O ambiente é representado por um modelo de mapa em grade, em que uma grade pode estar disponível ou com obstáculo. Os mapas usados nos experimentos são complexos em relação ao número de obstáculos e o algoritmo proposto foi capaz de encontrar soluções satisfatórias. Entretanto, os autores não mencionam como um robô pode navegar nesses mapas. O algoritmo também apresentou soluções melhores em pouco tempo quando comparado a algoritmos clássicos de ACO.

A Tabela 2 resume os trabalhos apresentados destacando a representação do ambiente, se lida com ambientes dinâmicos, referência a métodos de navegação e se foi realizado experimento em robô real.

Tabela 2 – Trabalhos relacionados a algoritmos de otimização por colônia de formigas aplicados ao problema de planejamento de trajetórias de robôs móveis.

Referência	Representação do ambiente	Ambiente dinâmico	Método de navegação	Experimento Real
(GARCIA <i>et al.</i> , 2009)	Nós interconectados	Não	Não	Não
(ZHU <i>et al.</i> , 2011)	Grade	Sim	Não	Não
(ZHU; HU; HENSCHEN, 2013)	Grade	Sim	Não	Não
(IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011)	Grade	Não	Não	Não
(YAO <i>et al.</i> , 2015)	Grade	Não	Não	Não

Fonte: Dados da pesquisa.

2.2.3 Outros algoritmos

2.2.3.1 Abordagem baseada em fronteira

A abordagem baseada em fronteira, proposta por Yamauchi (1997), foi definida para a tarefa de exploração, na qual os robôs não possuem um destino. Fronteiras são regiões limitadas no ambiente entre o espaço conhecido e o não explorado. Grades de evidência são usadas para representar o ambiente, no qual cada célula que compõe uma grade armazena a probabilidade de ser ocupada. A grade de evidência é construída usando a combinação de sensores sonar e laser e assim as regiões de fronteira são detectadas. Desta forma, cada célula é rotulada como livre, desconhecida ou ocupada, de acordo com uma probabilidade definida previamente. O robô explora as regiões desconhecidas ordenadas pelas regiões mais próximas.

Uma extensão da abordagem baseada em fronteira para múltiplos robôs foi apresentada por Yamauchi (1998). Cada robô tem uma grade de evidência global e quando ele está em uma nova fronteira, a nova região é percebida por ele, que constrói uma grade de evidência local. A grade global é atualizada com informação do novo ambiente e a grade local é transmitida para os outros robôs, que também atualizam sua grade global. Essa abordagem é cooperativa e descentralizada porque cada robô tem a informação obtida por outros robôs. Assim, baseado na informação compartilhada, cada robô pode decidir melhor sobre regiões a serem exploradas.

A abordagem baseada em fronteira pode ser uma boa estratégia para exploração mas pode falhar em um ponto de mínimo porque a decisão é baseada em informação local. Na solução para múltiplos robôs, há o custo de cada robô manter uma representação completa do mapa, que deve ser frequentemente atualizada. Concluindo, de forma resumida podemos dizer que a abordagem baseada em fronteira realiza um mapeamento exploratório do ambiente, com uma navegação onde o robô é atraído para as fronteiras desconhecidas, e tendo como características próprias a construção de mapas baseados em grades e o conhecimento da localização (precisa) do robô.

2.2.3.2 Algoritmos Bug

Lumesky e Stepanov ([LUMELSKY; STEPANOV, 1986](#)) propuseram os algoritmos Bug 1 e Bug 2 para o problema de planejamento de trajetórias de robôs que se movem em ambientes com obstáculos. Esses algoritmos assumem que não há informação disponível sobre o formato e a posição dos obstáculos, que possuem área finita e limites de tamanho finito. As coordenadas do alvo e dos obstáculos são fixas. O conhecimento local do ambiente é adquirido através de sensores. O caminho é formado por conjunto de pontos de encontros e partidas de obstáculos. O robô tem conhecimento apenas do ponto inicial de partida, do ponto de chegada e da sua posição atual em relação a esses pontos. Portanto, o problema como proposto e solucionado pelos algoritmos Bug se aproxima muito da proposta desta tese: conhecimento do ponto de partida, de destino, e ausência de mapa, realizando uma navegação exploratória.

No algoritmo Bug 1, o robô deixa a posição inicial e se move em direção ao objetivo. Quando encontra um obstáculo, armazena a posição do encontro e contorna o obstáculo usando o comportamento de seguir parede, mantendo o ponto de menor distância entre a localização atual e o objetivo. Quando finaliza o contorno, voltando ao ponto de encontro, o robô se move para o ponto de partida ponto de partida (ponto mais próximo do objetivo e que não esteja bloqueado na direção deste, onde o robô poderá finalizar o processo de seguimento da parede), contornando o obstáculo, e a partir desse ponto se dirige ao objetivo.

O algoritmo Bug 2 se baseia na determinação de uma linha reta virtual, que conecta o ponto inicial ao objetivo. O robô segue essa linha virtual até alcançar o objetivo ou encontrar um obstáculo. Quando encontra um obstáculo, o robô o contorna, seguindo a parede até encontrar a linha novamente. O processo se repete a cada novo obstáculo encontrado até que o robô alcance o objetivo.

Ambos os algoritmos retornam uma solução, se ela existir. O Bug 1 é um algoritmo de busca exaustivo porque olha todas as escolhas antes de decidir, porém tem desempenho mais estável. O Bug 2 é um algoritmo de busca guloso porque escolhe a primeira solução que parece ser a melhor e supera o Bug 1 em desempenho na maioria dos casos. De forma resumida podemos dizer que os algoritmos Bug1 e Bug2 são responsáveis pela navegação de um robô de uma posição inicial até um destino conhecido, desviando dos obstáculos no caminho. Esta abordagem, de acordo com a proposta original dos algoritmos, no entanto, não busca criar mapas do ambiente, constituindo-se de uma abordagem de navegação e não de mapeamento e planejamento exploratórios. Os algoritmos Bug1 e Bug2 não realizam uma exploração mais completa do ambiente a fim de obter uma trajetória mais otimizada até o destino.

2.2.3.3 Campos potenciais artificiais

No planejamento de trajetórias baseado em campos potenciais artificiais, proposto por [Khatib \(1986\)](#), a ideia básica é atribuir um potencial de atração, gerado pela configuração de

destino, e um potencial de repulsão, gerado pelos obstáculos. Dessa forma, o robô é induzido a se aproximar e alcançar a configuração alvo, enquanto desvia dos obstáculos no caminho. Assim, o princípio desse método usado para o planejamento e navegação de rotas consiste em calcular o movimento do robô a partir das forças de atração e repulsão que nele atuam. Idealmente, o movimento continua até que a força resultante seja nula (ADORNO; BORGES, 2014). O algoritmo sempre encontra algum momento em que a força resultante sobre o robô é nula; entretanto, ele pode ter encontrado um mínimo local, ou seja, não é garantido que o robô esteja na configuração final. Para evitar essa situação, a função potencial deve ser determinada criteriosamente. Em geral, essa escolha depende da topologia do espaço de configurações. A navegação baseada por campos potenciais usualmente é implementada sob a forma de comportamentos reativos e sem o uso de mapas conhecidos a priori (e também sem considerar a exploração, construção de mapas ou de planos de navegação). Concluindo, os campos potenciais são mais focados na navegação reativa, e não são considerados como um método de planejamento exploratório que busca achar caminhos otimizados, sendo apenas direcionados pelo objetivo.

2.2.3.4 Algoritmo A^*

O algoritmo de busca informada A^* é o mais bem conhecido algoritmo de busca em Inteligência Artificial (RUSSELL; NORVIG, 2009). O algoritmo A^* e suas variações tem sido aplicados para o problema de planejamento de trajetórias em Robótica (HAWA, 2013; CAGIGAS, 2005; AUTERE, 2002). Nesse algoritmo, cada nó a ser expandido é avaliado usando uma função de avaliação $f(n) = g(n) + h(n)$, em que $g(n)$ é o custo real de deixar a posição inicial e alcançar o nó atual n , e $h(n)$ é uma função heurística que estima o custo de deixar o nó atual n a alcançar o objetivo.

A^* mantém uma lista com os valores de f para cada nó visitado e cada nó disponível esperando para ser explorado. Em cada passo, decide-se explorar o nó com menor valor de f . O algoritmo pode explorar um nó vizinho relacionado ao nó atual visitado ou pode escolher visitar outro nó disponível pendente na lista, sendo necessário realizar um retrocesso.

O algoritmo A^* garante encontrar a solução ótima para heurísticas admissíveis. Uma heurística é admissível se nunca superestima o custo para alcançar o objetivo (RUSSELL; NORVIG, 2009). Porém, esse algoritmo utiliza representação precisa do ambiente, geralmente em grade, para que a solução encontrada seja aplicável no ambiente real. O algoritmo realiza vários retrocessos na busca pela solução ótima, que pode ser impraticável em algumas aplicações.

2.2.3.5 Algoritmo D^* Lite

D^* Lite é um algoritmo de busca informada para ambientes desconhecidos e dinâmicos. Ele é baseado no LPA* (KOENIG; LIKHACHEV; FURCY, 2004), uma versão incremental do algoritmo A^* (HART; NILSSON; RAPHAEL, 1968).

D* Lite mantém duas estimativas de custos para cada nó $s \in S$, em que S denota o conjunto finito de vértices do grafo: $g(s)$ e $rhs(s)$. O $g(s)$ é a estimativa da distância inicial de cada nó s . O rhs é o valor um passo a frente que satisfaz a seguinte relação:

$$rhs(s) = \begin{cases} 0 & \text{if } s = s_{start} \\ \min_{s' \in Pred(s)} (g(s') + c(s', s)) & \text{, caso contrário} \end{cases} \quad (2.1)$$

em que $Pred(s) \subset S$ denota o conjunto de predecessores de $s \in S$ no grafo; $0 < c(s, s') < \infty$ denota o custo de se mover de s para $s' \in Succ(s)$, $Succ(s) \subset S$ denota o conjunto de sucessores no grafo.

Um nó é considerado consistente se seu valor- g é igual ao seu valor- rhs , caso contrário, ele é inconsistente. A fila de prioridade mantém os nós inconsistentes, que precisam ser atualizados para se tornarem consistentes. A prioridade do nó s na fila de prioridade é:

$$k(s) = [k_1(s); k_2(s)] \quad (2.2)$$

em que $k_1(s) = \min(g(s), rhs(s)) + h(s, s_{goal})$ e $k_2 = \min(g(s), rhs(s))$.

O algoritmo 1 mostra os procedimentos *Main()* e *ComputeShortestPath* da versão básica do D* Lite. No início, o procedimento *Main()* invoca o método *Initialize()*, em que os valores- g e valores- rhs são inicializados com valor infinito. O valor- rhs do nó objetivo foi definido para zero, tornando-se inconsistente. Por isso, o nó objetivo é inserido na fila de prioridade.

Logo após, o algoritmo calcula o caminho mais curto do nó s_{start} para o objetivo e o robô segue esse caminho. Se o robô não atingiu o objetivo, ele se move para o próximo nó do caminho mais curto e atualiza s_{start} para o nó atual do robô. Caso alguma alteração ocorra no ambiente, o algoritmo atualiza os valores das arestas afetadas e invoca o método *UpdateVertex()* para atualizar os valores de rhs dos nós afetados pela alteração de custos nas arestas, para que eles satisfaçam novamente o equivalente da Equação 2.1. Ele também atualiza a fila de prioridade para conter novamente os vértices inconsistentes com prioridades que satisfazem o equivalente da Equação 2.2. Em seguida, o algoritmo atualiza as prioridades de todos os nós na fila de prioridade, recalcula um caminho mais curto e itera.

O algoritmo D* Lite tem características semelhantes ao A*, sendo que o primeiro é capaz de lidar com alterações no ambiente ocorridas nas arestas.

Algoritmo 1 – D* Lite Algorithm

```

procedimento COMPUTESHORTESTPATH()
  enquanto ( $U.TopKey() < CalculateKey(s_{start}) \vee rhs(s_{start}) \neq g(s_{start})$ ) faça
     $u = U.Pop()$ ;
    se ( $g(u) > rhs(u)$ ) então
       $g(u) = rhs(u)$ ;
      para todo  $s \in Pred(u)$  faça
         $UpdateVertex(s)$ ;
      fim para
    senão
       $g(u) = \infty$ ;
      para todo  $s \in Pred(u) \cup u$  faça
         $UpdateVertex(s)$ ;
      fim para
    fim se
  fim enquanto
fim procedimento
procedimento MAIN
   $Initialize()$ ;
   $ComputeShortestPath()$ ;
  enquanto ( $s_{start} \neq s_{goal}$ ) faça
    /*if( $g(s_{start}) = \infty$ ) then there is no known path*/
     $s_{start} = \operatorname{argmin}_{s' \in Succ(s_{start})} (c(s_{start}, s') + g(s'))$ ;
    Move to  $s_{start}$ ;
    Scan graph for changed edge costs;
    se (any edge costs changed) então
      para todo (directed edges  $(u, v)$  with changed edge costs) faça
         $UpdateEdgeCost(u, v)$ ;
         $UpdateVertex(u)$ ;
      fim para
      para todo  $s \in U$  faça
         $U.Update(s, CalculateKey(s))$ ;
      fim para
       $ComputeShortestPath()$ ;
    fim se
  fim enquanto
fim procedimento

```

2.3 Considerações Finais

Esta seção apresentou o problema de planejamento de trajetórias tratado por essa pesquisa, no qual o ambiente é um mapa do tipo labirinto, *indoor*, desconhecido pelos robôs (sem mapa previamente conhecido), onde o caminho até o objetivo final é descoberto pelo robô, representado de forma topológica. Como os robôs não possuem conhecimento prévio do ambiente, este é considerado um problema de planejamento exploratório de trajetórias, no qual os robôs descobrem o ambiente à medida que planejam sua rota para um objetivo. Além disso, cabe destacar que nesta pesquisa busca-se definir métodos que não dependam nem de uma localização precisa e nem de uma representação precisa do ambiente ou da trajetória.

A Tabela 3 apresentada abaixo, apresenta um resumo reunindo as diferentes abordagens estudadas e encontradas na literatura, comparando como cada uma delas aborda a questão de localização, representação e uso de mapas, planejamento e exploração de trajetórias e navegação, abordagens estas que permitem aos robôs alcançar uma determinada posição final no ambiente. Além disso também devem ser considerados os aspectos referentes a forma como a solução é buscada, considerando a exploração de múltiplas rotas, podendo ser realizada por um ou mais robôs.

Na Tabela 3, a coluna *Representação do ambiente* pode assumir os seguintes valores: Sm = Sem Mapa pré-definido, Cm = Com Mapa pré-definido ou Gm = Gera Mapa ou Representação da Trajetória (Inicia sem mapa); P = Mapa preciso ou I = Mapa impreciso; M = Mapa métrico, T = Mapa topológico ou O = outro. A coluna *Localização* Pode ter os valores: SL - Sem Localização, LI = Localização Imprecisa ou LP = Localização Precisa. A coluna *Exploração* pode ser: I = Individual (um só robô) ou M = Múltiplos robôs (busca por múltiplas soluções e trajetórias).

Tabela 3 – Trabalhos relacionados a algoritmos genéticos aplicados ao problema de planejamento de trajetórias de robôs móveis.

Referência	Tipo de Representação	Localização	Trajetória
(HU; YANG, 2004)	Cm;P;M	LP	M
(SEDIGHI <i>et al.</i> , 2004)	Grade	Não	Não
(YUN; GANAPATHY; CHONG, 2010)	Cm;P;M	LP	M
(TUNCER; YILDIRIM, 2012)	Cm;P;M	LP	M
(MOHANTA; PARHI; PATEL, 2011)	Gm;Im;N;O	SL	M
(KALA, 2012)	Sm;P;N;O	LP	M
(QU; XING; ALEXANDER, 2013)	Cm;P;M	LP	M
(DAVOODI <i>et al.</i> , 2013)	Cm;P;M	LP	M
(GARCIA <i>et al.</i> , 2009)	Cm;P;M	LP	M
(ZHU <i>et al.</i> , 2011)	Cm;P;M	LP	M
(ZHU; HU; HENSCHEN, 2013)	Cm;P;M	LP	M
(IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011)	Cm;P;M	LP	M
(YAO <i>et al.</i> , 2015)	Cm;P;M	LP	M
Abordagem baseada em Fronteiras	Gm;P;M	LP	I
Algoritmos Bug	Sm	LP	I
Algoritmo A*	Cm;P;M	LP	I
Algoritmo D* Lite	Cm;P;M	LP	I
Abordagem proposta na Tese	Gm;I;N;T	LI	M

Fonte: Dados da pesquisa.

De acordo com a Tabela 3 nota-se que a abordagem proposta nessa tese é diferenciada e original em relação as demais e atende aos requisitos de ser simples, eficiente (em relação à distância percorrida e obtendo uma trajetória satisfatória), onde pode ser aplicada em problemas como o de busca e resgate. A abordagem oferece vantagens sobre as demais como localização imprecisa, sem mapa prévio, gera uma representação da trajetória na forma de um mapa topológico, realiza uma exploração testando múltiplas soluções e encontra soluções ótimas ou quase-ótimas. Localização imprecisa quer dizer que nossa abordagem não depende de precisão da localização do robô. Por exemplo, algoritmos baseados em representação de grade são fortemente afetados de acordo com a precisão da grade.

ALGORITMOS PROPOSTOS PARA PLANEJAMENTO DE TRAJETÓRIAS

Algoritmos heurísticos que realizam busca em espaço de soluções podem ser classificados em algoritmos baseados em instância ou algoritmos baseados em modelo, de acordo com [Zlochin et al. \(2004\)](#). Os algoritmos baseados em instância geram novas soluções candidatas usando a solução atual ou a população de soluções atuais, como os algoritmos genéticos. Os algoritmos baseados em modelo geram soluções candidatas usando um modelo probabilístico parametrizado, que é atualizado de acordo com as soluções anteriores, permitindo que a busca se concentre em regiões com boas soluções.

Os Algoritmos Genéticos são metaheurísticas inspiradas no princípio de sobrevivência do indivíduo mais apto, que imitam processos de evolução biológica na solução de problemas ([MITCHELL, 1995](#)). Esse princípio faz com que indivíduos com variações mais adaptadas ao ambiente tenham maior probabilidade de sobrevivência, bem como maior probabilidade de reprodução ([ATMAR, 1992](#)).

Segundo [Rezende \(2003\)](#), uma tarefa de busca possui vários componentes, entre eles, o espaço de busca, nas quais as possibilidades de solução de um problema são consideradas, e a função de avaliação, que define como os elementos do espaço de busca serão avaliados. Diferente dos métodos de busca tradicionais, que iniciam o processamento com apenas uma solução candidata, os métodos evolutivos lidam com uma população de soluções candidatas em paralelo. Desse modo, elas exploram diferentes regiões do espaço de busca e, assim, têm maior chance de atingir áreas mais promissoras.

Para aplicar um AG em um problema de otimização, uma representação da solução busca codificar características do problema em questão. Essas soluções codificadas são os cromossomos cujos componentes são chamados genes. A maioria dos AG têm, pelo menos, os seguintes elementos em comum: população de cromossomos, seleção de acordo com a aptidão, cruzamento para produzir novos descendentes e mutação aleatória de novos descendentes ([MITCHELL,](#)

1995).

Algoritmo de Otimização por Colônias de Formigas (DORIGO; STÜTZLE, 2004)) é uma técnica de busca baseada em modelo bem estabelecida (ZLOCHIN *et al.*, 2004). ACO é uma meta-heurística inspirada pelo comportamento de busca por comida das formigas. Formigas constroem seus caminhos para o destino por uma escolha probabilística para se mover para um nó vizinho. A escolha probabilística é baseada no feromônio depositado por outras formigas e na função heurística. Então, as formigas caminham de volta de forma determinística, depositando feromônio no grafo. A quantidade de feromônio depende da qualidade das soluções. Assim, uma formiga artificial tem dois papéis importantes: gerar soluções e atualizar os parâmetros do modelo. É importante destacar que o ACO realiza uma exploração em paralelo do espaço de busca, considerando diferentes possíveis soluções que são exploradas simultaneamente.

Neste capítulo, são apresentados os algoritmos propostos nesta pesquisa para o problema de planejamento de trajetórias de robôs móveis autônomos, baseados em algoritmos genéticos e algoritmos de otimização por colônias de formigas. Na Seção 3.1 são apresentados os algoritmos propostos baseados em AG e na Seção 3.2 são apresentados os algoritmos propostos baseados em ACO.

3.1 Algoritmos Genéticos

Nesta seção são apresentados os algoritmos propostos baseados em Algoritmos Genéticos.

3.1.1 Algoritmo Genético I

O primeiro algoritmo genético investigado nesta pesquisa, conforme descrito em Santos *et al.* (2014), foi proposto para encontrar a melhor sequência de ações, ou comportamentos reativos, para que o robô alcance o seu destino de forma segura. O objetivo é automaticamente fornecer a sequência de comportamentos reativos para seguir um caminho usando a abordagem de navegação topológica. Na maioria dos trabalhos encontrados na literatura, o cromossomo é modelado como uma sequência de pontos exatos que o robô tem que visitar. Nessa pesquisa, foi proposta uma nova forma de representar o caminho, na qual o cromossomo é definido como uma sequência de números inteiros, sendo que cada número representa um comportamento reativo que pode ser executado pelo robô. A Figura 4 mostra um exemplo de cromossomo. As possíveis ações a serem executadas são enumeradas como *seguir em frente* (0), *virar à esquerda* (1), *virar à direita* (2), *dar meia volta* (3) e *seguir no corredor* (4).

A avaliação de um indivíduo nesse problema significa que o robô precisa navegar no ambiente, seguindo a sequência de comportamentos reativos, e alcançar o objetivo com sucesso. Realizar a navegação de cada indivíduo diretamente em um simulador robótico tornaria o processo evolutivo muito lento. Assim, a representação topológica do mapa em grafo é utilizada

Figura 4 – Um exemplo de representação do indivíduo. Cada gene representa um código para uma ação.

1	4	0	2	4	0	4	1	4	3	4	1	4	0	4	0	4	2	4	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Fonte: Elaborada pelo autor.

para cada indivíduo realizar seu percurso. Apesar do mapa ter sido considerado no processo evolutivo, a abordagem proposta nesta tese permite que seja feita uma implementação que funcione totalmente baseada nas leituras dos sensores do robô e sem o uso de um mapa prévio. A Figura 5 apresenta um exemplo de mapa e a Figura 6 apresenta a representação em grafo correspondente. Nessa representação o corredor é considerado um nó, por isso, a ação *seguir no corredor* foi especificada.

Por exemplo, assuma as posições inicial e final na Figura 5, o robô vindo do oeste e executando as ações definidas pela Figura 4. De acordo com esse cromossomo, o robô deve executar as ações: *virar à esquerda*, *seguir no corredor*, *seguir em frente*, *virar à direita*, *seguir no corredor*, *seguir em frente*, *seguir no corredor*, *virar à esquerda* e *seguir no corredor*. No grafo correspondente, ilustrado pela Figura 6, os seguintes nós serão visitados: 3, 4, 5, 8, 9, 10, 11, 12, 20 and 19. Pode-se observar que o cromossomo possui 20 ações possíveis de serem aplicadas, mas o objetivo foi alcançado com apenas nove ações. Neste caso, os genes restantes não são considerados.

Figura 5 – Um exemplo de mapa. S é a posição inicial do robô e G é o seu objetivo.

Fonte: Elaborada pelo autor.

Os genes de cada cromossomo da população inicial são preenchidos aleatoriamente com possíveis valores inteiros para as ações relacionadas. Cada cromossomo é avaliado pela execução da navegação do robô na representação de grafo. Nessa etapa, um operador de reparo é executado. Se uma ação indicada pelo gene não puder ser executada durante a navegação no

Figura 6 – O grafo correspondente da Figura 5.

Fonte: Elaborada pelo autor.

grafo, o operador de reparo substitui a ação por uma nova, de acordo com as possíveis ações para este nó no grafo. Por exemplo, se o robô está vindo do sul para o nó 5 na Figura 6, e a próxima ação do gene a ser aplicada é 1, a ação virar à esquerda deve ser executada. Entretanto, essa ação não é possível para esse nó e deve ser substituída. Há duas ações possíveis para esse nó, 0 ou 2, que é escolhida aleatoriamente pelo operador de reparo e fixada como novo valor de gene.

A função de avaliação é baseada no tamanho do caminho executado pelo robô e na distância da sua posição atual para o objetivo, que considera as coordenadas aproximadas de acordo com a representação de grafo. A distância percorrida é a soma das distâncias euclidianas entre cada par de nós V_i e V_j , do conjunto de nós incluso no caminho (Equação 3.1). Quando a ação *dar meia volta* (3) é escolhida, a distância entre os nós é somada duas vezes, uma vez que o robô entrou em um corredor bloqueado. Se o objetivo é alcançado, essa soma é dividida por uma constante α . Se o objetivo não é alcançado, a distância euclidiana entre os nós atual e objetivo é adicionada na soma, multiplicada por uma constante β . A equação 3.2 é a função de avaliação explicada, em que f é o valor de aptidão, α e β são valores constantes, V_a é o nó atual, V_g é o nó objetivo e $d(V_a, V_g)$ é a distância euclidiana entre os nós V_a e V_g .

$$soma = \sum d(V_i, V_j) \quad (3.1)$$

$$f = \begin{cases} soma/\alpha & \text{se o objetivo for alcançado} \\ soma + \beta * d(V_a, V_g) & \text{se o objetivo não for alcançado} \end{cases} \quad (3.2)$$

A população de cada geração t é formada por $p\%$ dos melhores indivíduos da população na geração $t - 1$. Os outros $(1 - p)\%$ são determinados por cruzamento e mutação. O torneio

entre dois indivíduos seleciona dois pais da população na geração $t - 1$. Assim, o cruzamento uniforme é aplicado sobre esses pais, gerando um novo indivíduo. A mutação pode ser aplicada ao novo indivíduo de acordo com a taxa de mutação. Se a taxa de mutação for satisfeita para um indivíduo, um gene escolhido aleatoriamente é alterado, que significa escolher aleatoriamente outra ação para ser executada. Finalmente, o valor de aptidão do novo indivíduo é determinado. O novo indivíduo é inserido na população, exceto se seu valor de aptidão for igual ao de um de seus pais. Neste caso, o processo de reprodução é repetido. O processo evolutivo é executado até que um dos dois critérios de parada seja satisfeito: se o melhor indivíduo encontrado até o momento não é melhorado por um número de gerações ou se o número máximo de gerações é alcançado.

Depois que o processo evolutivo acontece na representação de grafo, a melhor solução encontrada é executada usando o simulador robótico Player/Stage (COLLETT; MACDONALD; GERKEY, 2005) para demonstrar a factibilidade da representação da solução em um ambiente robótico. Nesse estágio, a RNA proposta por Sales, Osório e Wolf (2011) é usada para detectar mudanças no estado atual do robô, ativando as ações codificadas na melhor solução encontrada pelo AG. Os resultados dos trabalhos desenvolvidos baseados nesta abordagem são apresentados no Capítulo 4 e na seguinte publicação: (SANTOS *et al.*, 2014).

3.1.2 Algoritmo Genético II

Nessa etapa da pesquisa, descrita por Santos V. C.; Toledo e Osório (2014), nova estrutura de ambiente é proposta, na qual é possível executar quatro novas ações: virar à nordeste, virar à sudeste, virar à sudoeste e virar à noroeste. A Figura 7 ilustra um robô e todas as ações possíveis de serem executadas na nova estrutura de ambiente.

Figura 7 – Um exemplo do robô em uma bifurcação onde todas as ações possíveis são representadas.

Fonte: Elaborada pelo autor.

A representação do cromossomo é a mesma, definida como uma sequência de números inteiros, em que cada número representa uma ação. As ações possíveis e seus respectivos números são: *virar à oeste* (0), *virar à noroeste* (1), *seguir para o norte* (2), *virar à nordeste* (3), *virar à leste* (4), *virar à sudeste* (5), *seguir para o sul* (6) e *virar à sudoeste* (7).

Por exemplo, considere as posições inicial e final indicadas na Figura 9, o robô vindo do leste e o cromossomo da Figura 8. De acordo com esse cromossomo, o robô deve executar as seguintes ações: *seguir para o norte*, *virar à noroeste*, *virar à sudoeste*, *virar à leste* e *virar à noroeste*. No grafo correspondente ilustrado na Figura 10, a seguinte sequência de nós será visitada: 4, 3, 7, 8, 12 e 13.

Figura 8 – Exemplo de cromossomo. Cada gene representa uma ação definida por um valor inteiro.

2	1	7	4	1
---	---	---	---	---

Fonte: Elaborada pelo autor.

Figura 9 – Exemplo de mapa.

Fonte: Elaborada pelo autor.

Assim, o segundo algoritmo genético (AG II) foi proposto, em que a população inicial é gerada aleatoriamente e agora cada cromossomo tem um tamanho diferente, que varia entre os tamanhos mínimo e máximo estabelecidos. Cada gene é definido com um valor inteiro aleatório, de acordo com as ações possíveis. A avaliação é feita pela navegação na representação de grafo. Durante a navegação, o operador de reparo é aplicado. Se a ação indicada pelo gene não puder ser executada, ela é substituída por outra possível, escolhida aleatoriamente.

Figura 10 – O grafo correspondente da Figura 9.

Fonte: Elaborada pelo autor.

A população é substituída por novos indivíduos a cada geração. A seleção por torneio é usada para selecionar os pais para reprodução. Os filhos são gerados por cruzamento e mutação, sendo que o cruzamento uniforme, de um ponto e de dois pontos foram avaliados. Se os pais tem tamanhos diferentes, o novo indivíduo é preenchido até o tamanho do menor. A partir desse ponto, ele é preenchido até o tamanho do maior, sendo que cada gene pode ser adicionado com 50% de probabilidade.

Por exemplo, um novo indivíduo gerado por cruzamento uniforme é ilustrado na Figura 11. Até o quinto gene, escolhe-se se ele virá do pai 1 ou do pai 2. Os genes escolhidos são indicados pela cor cinza. A partir daí, os genes do pai 2 podem ser adicionados ou não, de acordo com a probabilidade. Nesse exemplo, o sexto e o oitavo genes do pai 2 foram adicionados ao filho.

Figura 11 – Exemplo de cruzamento uniforme.

pai 1:	2	1	7	4	1			
pai 2:	4	0	2	3	4	1	1	5
filho:	2	0	2	4	4	1	5	

Fonte: Elaborada pelo autor.

Três tipos de mutação podem ser aplicados: mudança, inserção ou remoção. Na mutação por mudança, uma quantidade de genes é escolhida para ser alterada, onde são substituídos por

uma nova ação, selecionada aleatoriamente. Na mutação por inserção, uma quantidade de genes é adicionada ao novo indivíduo, limitada ao tamanho máximo. Na mutação por remoção, um gene é escolhido para ser removido, se o tamanho do indivíduo for maior que o tamanho mínimo estabelecido. Depois de aplicar o operador de mutação, o valor de aptidão do novo indivíduo é determinado.

Na Figura 12, é apresentado um exemplo dos três tipos de mutação. Na mutação por mudança, o terceiro e o quinto genes dos filhos foram mudados dos valores 2 e 4 para 1 e 3, respectivamente. Na mutação por inserção, dois genes foram adicionados aos filhos com valores 6 e 0. Na mutação por remoção, o segundo gene dos filhos (valor 0) foi removido.

Figura 12 – Exemplo dos três tipos de mutação.

filho:	2	0	2	4	4	1	5		
mudança:	2	0	1	4	3	1	5		
inserção:	2	0	2	4	4	1	5	6	0
remoção:	2	2	4	4	1	5			

Fonte: Elaborada pelo autor.

A função de aptidão é baseada no tamanho do caminho percorrido pelo robô e sua distância final para o objetivo. Assim, as coordenadas dos nós na representação de grafo são utilizadas, que são valores aproximados no mapa. O tamanho do caminho é a soma das distâncias euclidianas entre cada par de nós, v_i e v_{i+1} , do conjunto de nós V , de tamanho n , incluso no caminho (Equação 3.3). Quando uma ação de retorno é escolhida (*virar à sudeste (5)*, *seguir para o sul (6)* ou *virar à sudoeste (7)*), uma penalização γ é adicionada à soma porque não é desejável que robô se mova para trás.

$$soma = \sum_{i=1}^{n-1} \gamma * d(v_i, v_{i+1}) \quad (3.3)$$

Se o objetivo for alcançado, essa soma é dividida por um valor constante α . Caso contrário, a distância euclidiana entre o nó atual e o nó objetivo é adicionada à soma, multiplicada por um valor constante β . A função de aptidão é a mesma definida para o AG I na Equação 3.2.

Foram determinados dois critérios de parada para o AG: se o número máximo de gerações for alcançado ou se o melhor indivíduo encontrado não foi melhorado por 40% do número máximo de gerações

A melhor solução obtida pela evolução na representação de grafo é executada no simulador robótico Player/Stage (COLLETT; MACDONALD; GERKEY, 2005). O resultados são apresentados no Capítulo 4 e na publicação (SANTOS V. C.; TOLEDO; OSÓRIO, 2014).

3.1.3 Algoritmo Genético III

No último algoritmo genético proposto nesta pesquisa, apresentado por Santos, Toledo e Osorio (2015), a ideia de planejamento exploratório de trajetórias foi aprimorada, sendo que o objetivo é que os robôs alcancem a posição final percorrendo a menor distância possível. A trajetória dos robôs é construída incrementalmente, a cada geração do AG. Essa abordagem tem a vantagem de evoluir simultaneamente vários agentes para explorar o ambiente, onde a informação fornecida pelas melhores soluções encontradas é compartilhada através do processo evolutivo.

O AG proposto evolui sequências de ações que levarão um grupo de robôs da origem para o destino. A população tem um tamanho fixo de p indivíduos e cada indivíduo tem uma sequência inicial de ações com tamanho l . As ações representam comportamentos reativos locais que os robôs podem executar no ambiente desconhecido.

Assim como no AG II proposto, uma sequência de valores inteiros são usados para codificar as ações em um indivíduo, que são relativas à posição atual e orientação do robô. Os indivíduos da população inicial tem suas sequências de ações preenchidas aleatoriamente e cada sequência é executada pelo robô conectado a esse indivíduo na representação de grafo. Portanto, depois que o robô executa a sequência de ações, o indivíduo relacionado terá seu valor de aptidão atualizado, com base na trajetória percorrida no ambiente.

Novos indivíduos são criados através dos operadores de cruzamento e mutação. O cruzamento é executado sobre a sequência de ações fornecida pelos dois pais. Esses pais são selecionados por torneio, onde t indivíduos são escolhidos aleatoriamente e aquele com o melhor valor de aptidão se torna um pai. Quatro abordagens de cruzamento são avaliadas: uniforme, um ponto, dois pontos e aleatório. O cruzamento aleatório significa selecionar aleatoriamente uma entre as outras três abordagens de cruzamento cada vez que um indivíduo é gerado. Os tipos de mutação que podem ser aplicadas são de mudança, inserção ou remoção.

A cada geração t , p novos indivíduos com tamanho l podem ser criados substituindo a população anterior. Cada nova sequência de ações é executada por um robô relacionado, sendo que o valor de aptidão de cada novo indivíduo é calculado. A Figura 13 mostra um exemplo de evolução da geração 1 para a geração 2 em uma população com cinco indivíduos.

A execução dessas ações é ilustrada na Figura 14 para os dois primeiros indivíduos da Figura 13. Ambos os robôs começam na posição inicial s , mas a primeira ação do robô 1 é (2)-seguir para o norte, e do robô 2 é (4)-virar à leste. Assim, o mapa mostra o robô 1 se movendo para o norte e o robô 2 se movendo para leste. Quando ambos os robôs identificam um estado diferente de corredor, uma nova ação é disparada. A segunda ação do robô 1 é (4)-virar à leste; a segunda ação do robô 2 é virar à oeste. Esses dois passos são destacados em linhas pontilhadas na Figura 14. O restante do caminho executado pelos robôs são mostrados em linhas sólidas.

Figura 13 – Exemplo das duas primeiras gerações de uma população com cinco robôs.

Fonte: Elaborada pelo autor.

Figura 14 – Exemplo de execução das sequências de ações de dois indivíduos que partem da mesma posição *S* para o objetivo *G*.

Fonte: Elaborada pelo autor.

Algumas vezes o robô tenta executar uma ação mas não há caminho para executá-la. Por exemplo, a ação é *seguir para o norte* e há apenas duas opções no ambiente: *virar à leste* ou *virar à oeste*. Nesse caso, o operador de reparo é aplicado, em que outra ação é escolhida aleatoriamente até que uma ação viável seja selecionada.

Há dois pontos que distinguem o AG proposto de um AG tradicional. Primeiro, o valor de aptidão é acumulado através das gerações. Segundo, um indivíduo não é substituído quando seu robô relacionado alcança o objetivo, mas ele ainda pode fazer parte do operador de cruzamento. Esses pontos são detalhados a seguir.

O valor de aptidão de cada indivíduo depende do tamanho do seu caminho (distância percorrida) e da distância final para o objetivo. A Equação 3.4 determina a soma das distâncias euclidianas entre cada par de nós v_i e v_{i+1} do conjunto de nós V , com tamanho n , no caminho. Esse valor de distância é alcançado executando a sequência de ações de um indivíduo na geração t . A distância total percorrida pelo robô da geração 1 para a geração t é calculada pela Equação 3.5. A Equação 3.6 tem a função de aptidão, onde o valor da penalidade é adicionado para $totalDist_t$ quando um robô não alcança o objetivo. Essa penalidade é a distância euclidiana $d(v_f, v_g)$ entre o nó final v_f e o nó objetivo v_g , multiplicado por um valor constante α .

$$dist_t = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) \quad (3.4)$$

$$totalDist_t = dist_t + \sum_{k=1}^{t-1} dist_k \quad (3.5)$$

$$f = \begin{cases} totalDist_t & v_f = v_g \\ totalDist_t + \alpha * d(v_f, v_g) & v_f \neq v_g \end{cases} \quad (3.6)$$

Se um robô alcançou o objetivo em uma geração k , a sequência de ações codificada no indivíduo relacionado não será substituída nas próximas gerações $t \geq k$. Isso é um critério elitista que preserva a codificação de um indivíduo para as próximas gerações, cujo robô já alcançou o objetivo. A partir da Equação 3.6, esse indivíduo sempre terá o valor de aptidão baixo para as próximas gerações porque ele já alcançou o objetivo. Quanto menor o valor de aptidão, melhor é o indivíduo. Além de que essas ações terão mais chances de serem inseridas em novas sequências de ações, o que pode ajudar outros robôs a alcançarem o objetivo nas próximas gerações.

Se um robô não alcançou o objetivo em uma geração, seu valor de aptidão é penalizado e os operadores de cruzamento e mutação criarão uma nova sequência de ações para seus indivíduos na próxima geração. Note que o valor de aptidão não acumula penalidade. Quando uma nova sequência de ações é executada por um robô e o valor de aptidão do indivíduo é recalculado, a distância total é a distância percorrida na geração atual mais as distâncias percorridas na gerações anterior (Equação 3.5) sem o termo de penalização. Se o robô não alcança o objetivo, o termo de penalidade é adicionado ao valor de aptidão (Equação 3.6) na geração atual, independentemente do sucesso do robô em alcançar o objetivo nas gerações anteriores. A Figura 15 ilustra essas situações da geração t para a geração $t + 1$ para uma população com cinco robôs.

O primeiro indivíduo alcançou o objetivo durante a geração t e ele não é atualizado na geração $t + 1$. O segundo indivíduo não alcançou o objetivo na geração t e seu valor de aptidão é penalizado, mas ele é atualizado e alcançou o objetivo na geração $t + 1$, visto que seu valor de aptidão não é atualizado agora.

Figura 15 – Exemplo de cálculo do valor de aptidão.

Valor de Aptidão	População na Geração t	Valor de Aptidão	População na Geração t+1
3228.44	<div><div>2</div><div>4</div><div>6</div><div>4</div><div>5</div></div>	3228.44	<div><div>2</div><div>4</div><div>6</div><div>4</div><div>5</div></div>
194300.35	<div><div>4</div><div>0</div><div>3</div><div>3</div></div>	4277.12	<div><div>2</div><div>0</div><div>4</div><div>0</div><div>2</div></div>
142250.25	<div><div>2</div><div>6</div><div>6</div><div>4</div><div>1</div></div>	153290.78	<div><div>2</div><div>1</div><div>5</div><div>3</div></div>
122810.05	<div><div>4</div><div>0</div><div>2</div><div>5</div><div>3</div><div>2</div></div>	147690.60	<div><div>0</div><div>4</div><div>2</div><div>4</div><div>5</div></div>
172830.20	<div><div>1</div><div>2</div><div>0</div><div>2</div><div>1</div></div>	184406.23	<div><div>2</div><div>0</div><div>4</div><div>4</div><div>1</div></div>

Fonte: Elaborada pelo autor.

O AG proposto é comparado ao algoritmo Algoritmo de Busca Informada (A^*), o mais bem conhecido algoritmo de busca em Inteligência Artificial (RUSSELL; NORVIG, 2009), em relação à distância percorrida pelo robô para deixar seu ponto inicial e alcançar o objetivo. O algoritmo A^* e suas variações tem sido aplicados para o problema de planejamento de trajetórias em Robótica (HAWA, 2013; CAGIGAS, 2005; AUTERE, 2002). O algoritmo A^* avalia cada nó a ser expandido usando uma função de avaliação $f(n) = g(n) + h(n)$, em que $g(n)$ é o custo real de deixar a posição inicial e alcançar o nó atual n , e $h(n)$ é uma função heurística que estima o custo de deixar o nó atual n a alcançar o objetivo. A função heurística adotada nesta pesquisa é a distância euclidiana entre o nó atual e o destino.

A^* mantém uma lista com os valores de f para cada nó visitado e cada nó disponível esperando para ser explorado. Em cada passo, decide-se explorar o nó com menor valor de f . O algoritmo pode explorar um nó vizinho relacionado ao nó atual visitado ou pode escolher visitar outro nó disponível pendente na lista. Em aplicações robóticas reais, quando o novo nó explorado não é um vizinho direto em relação ao nó atual, o robô deve executar um movimento de retrocesso, ou seja, voltar para uma posição explorada anteriormente.

O algoritmo A^* clássico não considera o custo do retrocesso, que é uma distância a ser coberta pelo robô no problema considerado neste trabalho. Por essa razão, uma pequena modificação no algoritmo A^* é proposta, denominado algoritmo C^* (Algoritmo de Busca com Estimativa Realista de Custo (C^*)), que considera também o custo $i(n)$ de deixar o nó atual e alcançar o próximo nó a ser explorado. Assim, a função f passa a ser definida como $g(n) + h(n) + i(n)$. Essa mudança permite que o algoritmo desista de retrocessos com longas distâncias, decidindo explorar os nós próximos ao nó atual. Os resultados desse algoritmo são apresentados no Capítulo 4 e na publicação (SANTOS; TOLEDO; OSORIO, 2015).

3.2 Algoritmos de Otimização por Colônia de Formigas

Nesta seção são apresentados os algoritmos propostos baseados em Algoritmos de Otimização por Colônia de Formigas.

3.2.1 Algoritmo Max-Min Ant System for Static Path Planning

O algoritmo *Max-Min Ant System* (MMAS) (DORIGO; STÜTZLE, 2004) foi investigado para o problema de planejamento exploratório de trajetórias em ambientes estáticos, denominado *Max-Min Ant System for Static Path Planning* (MMAS-SPP), como apresentado por Santos *et al.* (2016), em que os robôs devem explorar o ambiente ao mesmo tempo que planejam a trajetória do ponto inicial ao objetivo. MMAS é um algoritmo de otimização por colônia de formigas que explora as melhores soluções encontradas. Para comparar os resultados com o algoritmo AG III e com o algoritmo A*, além de analisar a qualidade das soluções obtidas, uma análise da distância percorrida pelos robôs para encontrar o objetivo na primeira iteração do algoritmo MMAS-SPP foi realizada.

Assim como os algoritmos genéticos propostos, o algoritmo MMAS-PP foi executado em um grafo, como ilustrado na Figura 16. Os pontos característicos do mapa são os nós no grafo e as arestas representam a navegabilidade entre esses pontos. Cada nó possui uma coordenada aproximada da sua posição no mapa, onde as distâncias exatas não são tão relevantes quanto são distâncias relativas entre os nós. Por exemplo, a distância entre os nós 1 e 7 é maior que a distância entre os nós 1 e 2. Quando uma formiga está em um nó, ela tem acesso aos nós vizinhos à sua posição, simulando o sensor em um ambiente real.

Figura 16 – Exemplo de um mapa e seu grafo correspondente.

Fonte: Elaborada pelo autor.

Os passos do algoritmo MMAS-SPP podem ser seguidos no Algoritmo 2. A linha 1 indica o passo de inicialização, no qual todas as formigas são posicionadas no mesmo ponto inicial. A quantidade de feromônio é inicializada com valores altos em todas as arestas. Depois da primeira iteração, essa quantidade será alterada para o limite superior de feromônio.

Algoritmo 2 – Algoritmo MMAS

```

1: initialization();
2:  $path_{gb} \leftarrow \emptyset$ ;
3: enquanto (o critério de parada não é alcançado) faça
4:    $path_{ib} \leftarrow \emptyset$ ;
5:   para  $i \leftarrow 1$  até numberOfAnts faça
6:      $path_i \leftarrow constructPath()$ ;
7:     localSearch( $path_i$ );
8:     se quality( $path_i$ ) > quality( $path_{ib}$ ) então
9:        $path_{ib} \leftarrow path_i$ ;
10:    fim se
11:  fim para
12:  pheromoneEvaporation();
13:  pheromoneUpdate();
14:  se quality( $path_{ib}$ ) > quality( $path_{gb}$ ) então
15:     $path_{gb} \leftarrow path_{ib}$ ;
16:  fim se
17: fim enquanto
18: return  $path_{gb}$ ;

```

Na fase de construção, linhas 4 – 11, cada formiga constrói seu caminho $path_i$ da posição de início para o objetivo. Em cada passo da fase de construção, uma formiga k no nó atual i calcula a probabilidade de se mover para um nó vizinho j , exceto o predecessor do nó i . Por exemplo, no mapa da Figura 16, se o nó atual é 3 e a formiga vem do nó 2, os nós viáveis para visitar são 5 e 6. O nó predecessor apenas pode ser uma opção em um corredor sem saída, onde a única opção é dar meia volta.

A probabilidade p_{ij}^k de uma formiga k se mover do nó i para o nó j é dada pela Equação 3.7 (DORIGO; STÜTZLE, 2004), onde τ_{ij} é a quantidade de feromônio na aresta que liga os nós i e j ; η_{ij} é o valor heurístico para mover de um nó i para j ; α e β define a influência da quantidade de feromônio e da informação heurística, respectivamente; N_i^k é o conjunto de vizinhos do nó i .

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{ if } j \in N_i^k \quad (3.7)$$

A função heurística adotada nesta pesquisa é o inverso da distância euclidiana entre o nó atual e o objetivo ($\eta_{ij} = 1/d_{ij}$), sendo que esse cálculo é baseado nas coordenadas aproximadas do mapa topológico. Durante a fase de construção, a distância percorrida por cada formiga é calculada na primeira iteração para comparar os resultados com o algoritmo AG III.

Essa abordagem pode ainda gerar laços nos caminhos porque não há controle sobre os nós visitados anteriormente, além do nó anterior ao atual. Para lidar com isso, um procedimento de busca local (linha 7) para remover laços é aplicado depois que as formigas alcançam o objetivo. Nesse procedimento, os laços são identificados por nós repetidos no mesmo caminho. A melhor solução encontrada depois da fase de construção é mantida na variável $path_{ib}$ (linhas 8 – 10).

Depois disso, o passo de evaporação (linha 12) é executada, de acordo com a Equação 3.8, onde ρ é taxa de evaporação. A taxa de evaporação tem um papel importante na velocidade de convergência do algoritmo. Quanto maior é esse valor, mais rápida a convergência é alcançada, porém, soluções sub-ótimas podem ser encontradas.

$$\tau_{ij} = (1 - \rho)\tau_{ij} \quad (3.8)$$

Na volta para a posição inicial, apenas a formiga que encontrou a melhor solução na iteração atual pode depositar feromônio na fase de atualização (linha 13). As outras seguem o mesmo caminho, sem depositar feromônio. A atualização do feromônio é definida pela Equação 3.9, onde $\Delta\tau_{ij}^{best} = 1/C^{best}$ e C^{best} é o tamanho da melhor solução de uma iteração. Se a melhor solução for a melhor solução encontrada de toda a execução, ela é guardada na variável $path_{gb}$ (linhas 14 – 16). No final, todas as formigas seguem o mesmo caminho de volta e a distância percorrida é acumulada para cada formiga.

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best} \quad (3.9)$$

No algoritmo MMAS, os valores do feromônio em cada aresta é limitado por uma faixa para evitar estagnação. O valor máximo de feromônio é definido como:

$$\tau_{max} = 1/\rho C^{bs} \quad (3.10)$$

onde C^{bs} é o tamanho da melhor solução encontrada em toda a execução do algoritmo. O valor mínimo é definido como:

$$\tau_{min} = \tau_{max}/a \quad (3.11)$$

onde a é um valor constante (DORIGO; STÜTZLE, 2004)(STÜTZLE; HOOS, 2000).

O processo é repetido até que um dos critérios de parada sejam satisfeitos: a melhor solução não é melhorada por um número fixo de iterações ou o número máximo de iterações é alcançado. O algoritmo retorna a melhor solução encontrada em $path_{gb}$ (linha 18).

Os experimentos e resultados são discutidos no Capítulo 4 e no artigo (SANTOS *et al.*, 2016).

3.2.2 Algoritmo Max-Min Ant System for Dynamic Path Planning

A aplicação do algoritmo MMAS para ambientes dinâmicos foi realizada nesta etapa da pesquisa, em que o ambiente dinâmico se refere a alterações em sua estrutura, considerando por exemplo, elementos como portas que possam ser abertas ou fechadas e passagens que são obstruídas.

Os passos do algoritmo *Max-Min Ant System for Dynamic Path Planning* (MMAS-DPP) são mostrados no Algoritmo 3. No *Main procedure*, o passo de inicialização é chamado na linha 2, em que todas as formigas são posicionadas no ponto inicial. Como proposto no algoritmo MMAS original, os rastros de feromônio são inicializados com uma grande quantidade de feromônio em todas as arestas. Depois da primeira iteração, essa quantidade será definida como o limite superior de feromônio. O *runMMAS procedure* é chamado na linha 3, que vai retornar a melhor solução $path_{gb}$. O procedimento *runMMAS* é equivalente ao algoritmo 2 (MMAS-SPP) que encontra a solução no ambiente estático.

Algoritmo 3 – Algoritmo MMAS-DPP

```

1: procedimento MAIN
2:   initialization();
3:    $path_{gb} = runMMAS()$ ;
4:   se changeIsDetected() então
5:     applySmoothing();
6:      $path_{gb} = runMMAS()$ ;
7:   fim se
8: fim procedimento
9: procedimento RUNMMAS
10:   $path_{gb} \leftarrow \emptyset$ ;
11:  enquanto (critério de parada não foi alcançado) faça
12:     $path_{ib} \leftarrow \emptyset$ ;
13:    para  $k \leftarrow 1$  até numberOfAnts faça
14:       $path_k \leftarrow constructPath()$ ;
15:      localSearch( $path_k$ );
16:      se quality( $path_k$ ) > quality( $path_{ib}$ ) então
17:         $path_{ib} \leftarrow path_k$ ;
18:      fim se
19:    fim para
20:    pheromoneEvaporation();
21:    pheromoneUpdate();
22:    se quality( $path_{ib}$ ) > quality( $path_{gb}$ ) então
23:       $path_{gb} \leftarrow path_{ib}$ ;
24:    fim se
25:  fim enquanto
26:  return  $path_{gb}$ ;
27: fim procedimento

```

Depois que a solução é encontrada no ambiente estático, a estrutura do ambiente é

alterada e o algoritmo deve buscar uma nova solução. Dois tipos de mudanças no ambiente foram aplicadas: remoção de aresta (equivalente à obstrução de passagem) e mudança de posição do objetivo. Antes de chamar o procedimento *runMMAS* para buscar a nova solução, o procedimento *applySmoothing* é executado. Esse procedimento é sugerido pelo algoritmo MMAS original (DORIGO; STÜTZLE, 2004) para suavizar a quantidade de feromônio quando o algoritmo converge. Esse mecanismo aumenta a quantidade de feromônio proporcionalmente à sua diferença entre o limite máximo de feromônio, de acordo com a Equação 3.12.

$$\tau_{ij}^*(t) = \tau_{ij}(t) + \delta(\tau_{max}(t) - \tau_{ij}(t)) \quad (3.12)$$

$\tau_{ij}^*(t)$ e $\tau_{ij}(t)$ são a quantidade de feromônio antes e depois da suavização, respectivamente. O objetivo de suavizar é facilitar a exploração pelo aumento da probabilidade de selecionar componentes da solução com baixa quantidade de feromônio. Além disso, a informação compartilhada durante a execução do algoritmo não é completamente perdida, apenas enfraquecida. Para $\delta = 1$, esse mecanismo corresponde à reinicialização da quantidade de feromônio, e para $\delta = 0$ ele é desabilitado.

Os experimentos e resultados são discutidos no Capítulo 4.

3.2.3 Considerações Finais

Este capítulo apresentou os algoritmos propostos nesta tese de doutorado para o problema de planejamento exploratório de trajetórias de robôs móveis autônomos. Os algoritmos assumem que os robôs não possuem conhecimento prévio do ambiente. Enquanto a maioria dos trabalhos encontrados na literatura propõem uma solução composta por conjunto de pontos, nós propomos uma solução composta por um conjunto de ações que o robô deve executar para alcançar o objetivo, devido ao uso de mapas topológicos. Isso permite que os robôs não dependam da obtenção precisa de localização durante a execução da trajetória encontrada.

O objetivo do primeiro algoritmo genético proposto foi encontrar o menor caminho para sair da posição inicial e alcançar o objetivo. Os mapas avaliados nesse algoritmo foram do tipo labirinto com ângulos retos nos cruzamentos. No segundo algoritmo genético, os mapas avaliados foram mais complexos, com cruzamentos retos e de 45° de inclinação. No terceiro algoritmo genético, a ideia de planejamento exploratório foi aprofundada, no qual a construção da solução foi realizada incrementalmente. Nesse caso, o objetivo foi alcançar o objetivo percorrendo a menor distância possível. Esse algoritmo foi comparado com o algoritmo clássico de busca heurística A*. Em todos os algoritmos genéticos propostos, a função de avaliação se baseia na distância já percorrida e na distância para alcançar o objetivo.

Um algoritmo de otimização por colônia de formigas (MMAS-SPP) foi proposto, baseado no *Max-Min Ant System* que se caracteriza por explorar as regiões mais promissoras no espaço de busca. Nesse algoritmo, além da distância para o objetivo, a quantidade de feromônio no

ambiente é considerada na tomada de decisão. Isso facilita a convergência do algoritmo para as melhores soluções, visto que a quantidade de feromônio é proporcional à qualidade das soluções encontradas, que favorece a troca de informação entre os robôs. Uma extensão do algoritmo MMAS-SPP foi proposta para lidar com ambientes dinâmicos, o algoritmo MMAS-DPP. Os experimentos mostraram que o algoritmo é capaz de encontrar uma nova solução quando ocorre uma mudança no ambiente, que pode ser uma obstrução de caminho ou mudança de posição do objetivo.

Assim, os algoritmos genéticos e os algoritmos de otimização por colônia de formigas são métodos eficientes na tarefa de planejamento exploratório de trajetórias de robôs móveis para ambientes representados por mapas topológicos. As abordagens propostas foram todas idealizadas de modo a garantir não somente a obtenção de um plano de navegação, mas também permitir que esse plano possa ser cumprido de forma adequada por robôs em aplicações reais, considerando portanto a navegação do robô (comportamentos locais e robustos de navegação). No capítulo a seguir, os resultados e discussões do uso desses algoritmos são apresentados.

RESULTADOS

Neste capítulo são apresentados e discutidos os resultados obtidos nesta pesquisa para os algoritmos genéticos e os algoritmos de otimização por colônia de formigas propostos. Na Seção 4.1 são apresentados os resultados dos algoritmos genéticos e na Seção 4.2 são apresentados os resultados dos algoritmos de otimização por colônia de formiga.

4.1 Algoritmos Genéticos

Nesta seção são apresentados e discutidos os resultados dos algoritmos genéticos propostos: AG I, AG II e AG III. Estes resultados também foram apresentados e discutidos nas seguintes publicações (SANTOS *et al.*, 2014; SANTOS V. C.; TOLEDO; OSÓRIO, 2014; SANTOS; TOLEDO; OSORIO, 2015).

4.1.1 Resultados do AG I

Para avaliar o desempenho do primeiro AG, foram realizados experimentos no mapa apresentado na Figura 17, que é similar ao mapa usado por Kala (2012). Nesses experimentos, os nós inicial e final são indicados na figura por *S* e *G1*, respectivamente. Os valores $\alpha = 2$ and $\beta = 2$ foram definidos para a função de aptidão e o tamanho do indivíduo foi fixado em 60 genes. Os critérios de parada são dez gerações sem atualizar a melhor solução ou alcançar o número máximo de 50 gerações. Esses valores foram empiricamente determinados com base em testes prévios.

Outros parâmetros do AG proposto são apresentados a seguir. O primeiro parâmetro avaliado é o tamanho da população, como descrito na Tabela 4. As colunas representam o tamanho da população, o número de gerações, o valor de aptidão do melhor indivíduo e o tempo de execução em milissegundos, respectivamente. Esses resultados são a média aritmética seguida pelo desvio padrão de 100 execuções do AG. Nesses experimentos, a taxa de mutação foi fixada

Figura 17 – Mapa usado nos experimentos. S é a posição inicial do robô e G1, G2, G3 e G4 são os objetivos.

Fonte: Elaborada pelo autor.

em 0.1 e uma estratégia elitista, chamada $E_1/2$, foi adotada. Esse elitismo significa que 50% dos melhores indivíduos na população na geração t permanecerão na geração $t + 1$.

Os valores de tamanho da população avaliados são 25, 50, 100, 150, 200, 250, 300, 500, 1000. Pode-se notar na Tabela 4 que o número de gerações diminui e os melhores valores de aptidão melhoram com o aumento do tamanho da população, embora o tempo de execução aumente. O algoritmo encontrou a mesma melhor solução final nas execuções com 300, 500 e 1000 indivíduos. Entretanto, o tamanho da população de 300 indivíduos apresenta o melhor compromisso entre o valor da solução e o tempo computacional. Por isso, esse valor foi definido para avaliar a taxa de mutação.

Tabela 4 – Análise do tamanho da população.

Tamanho da população	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
25	15.91 ± 5.28	53.86 ± 7.25	3700 ± 4852.36
50	15.59 ± 5.44	50.20 ± 2.45	7800 ± 5427.20
100	15.42 ± 4.92	49.10 ± 1.57	15200 ± 6273.24
150	14.77 ± 5.13	48.63 ± 0.76	22400 ± 8302.37
200	13.75 ± 3.70	48.68 ± 0.83	27200 ± 7664.69
250	12.88 ± 3.07	48.50 ± 0.25	32100 ± 8444.11
300	12.53 ± 2.50	48.48	37700 ± 8391.31
500	11.54 ± 2.23	48.48	58100 ± 12366.05
1000	10.38 ± 0.80	48.48	112500 ± 10187.63

Fonte: Dados da pesquisa.

Os diferentes valores aplicados para avaliar a mutação foram 0.05, 0.1, 0.15 e 0.2, apresentados na Tabela 5. Pode-se observar que a taxa de mutação teve pouca influência nos resultados. O menor valor de 0.05% gerou os melhores resultados com convergência mais rápida.

Tabela 5 – Análise da taxa de mutação.

Taxa de mutação	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
0.05	12.44 ± 2.72	48.48	37100 ± 8444.11
0.1	12.53 ± 2.5	48.48	37700 ± 8391.31
0.15	12.88 ± 2.87	48.50 ± 0.25	38300 ± 9107.11
0.2	12.64 ± 2.79	48.52 ± 0.39	38200 ± 9574.80

Fonte: Dados da pesquisa.

Usando o tamanho de população 300 e a taxa de mutação de 0.05, a última análise dos parâmetros do AG é sobre a proporção de indivíduos mantidos para a próxima geração durante o processo evolutivo. As estratégias elitistas avaliadas foram denominadas como E_0 , E_1 , $E_{1/3}$ e $E_{1/2}$. A estratégia E_0 significa atualizar toda a população com novos indivíduos, enquanto E_1 mantém apenas o melhor indivíduo da população anterior. As estratégias $E_{1/2}$ e $E_{1/3}$ mantêm, respectivamente, metade e um terço dos melhores indivíduos da população anterior. A Tabela 6 mostra os resultados alcançados.

Tabela 6 – Análise da taxa de proporção elitista.

Taxa de proporção elitista	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
E_0	47.49 ± 7.44	55.67 ± 4.77	247100 ± 38828.68
E_1	10.98 ± 1.34	48.48	62300 ± 7895.15
$E_{1/3}$	11.75 ± 1.92	48.48	47500 ± 8918.82
$E_{1/2}$	12.44 ± 2.72	48.48	37100 ± 8444.11

Fonte: Dados da pesquisa.

Quando a melhor solução foi descartada de uma geração para a outra (E_0), a convergência foi mais lenta e o número máximo de gerações quase foi atingido. Quando o elitismo foi introduzido, a melhor solução foi encontrada em todos os testes sem diferença relevante entre as estratégias E_1 , $E_{1/3}$ e $E_{1/2}$. Nesse caso, a estratégia E_1 retorna a melhor solução mais rápida que as outras.

Para avaliar o impacto do operador de reparo proposto, o AG também foi executado sem esse operador. Nesse experimento, o tamanho da população foi definido com 300 indivíduos, taxa de mutação de 0.05 e estratégia elitista E_1 . Os resultados são apresentados na Tabela 7. O método retornou o mesmo valor em todas as execuções, consumindo mais tempo do que quando o operador é aplicado. Além disso, o método com operador de reparo retornou a melhor solução.

A Tabela 8 apresenta os resultados encontrados para todos os objetivos ilustrados na Figura 17. O método proposto gasta quase o mesmo tempo computacional e número de gerações

Tabela 7 – Análise do operador de reparo.

Operador de reparo	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
Não	30.58 ± 0.50	63.64	275200 ± 46176.65
Sim	10.98 ± 1.34	48.48	62300 ± 7895.15

Fonte: Dados da pesquisa.

para os diferentes objetivos e sempre encontra a melhor solução. Esses resultados mostram a eficácia e eficiência da abordagem proposta para resolver esse problema.

Tabela 8 – Resultados dos experimentos com diferentes objetivos.

Objetivo	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
1	10.98 ± 1.34	48.48	62300 ± 7895.15
2	10	31.72	53600 ± 4824.18
3	10.01 ± 0.1	44.35	52900 ± 4776.84
4	11.18 ± 1.35	57.65	62800 ± 8297.50

Fonte: Dados da pesquisa.

Uma simulação na navegação completa foi desenvolvida para o mapa representado na Figura 17, onde o início e o objetivo são indicados na figura por *S* e *GI*, respectivamente. Esse experimento foi realizado no simulador robótico Player/Stage, com o robô simulado Pioneer P3-AT com um sensor SICK Lidar 180°. A rede neural proposta por Sales, Osório e Wolf (2011) foi treinada para reconhecer sete estados possíveis: corredor, cruzamento, curva à esquerda, curva à direita, bifurcação à esquerda, bifurcação à direita e bifurcação em forma de T. A base de dados de treinamento foi coletada em um mapa diferente com características estruturais semelhantes para avaliar a robustez da rede neural e a capacidade de generalização. A RNA usada é do tipo perceptron multicamadas e foi treinada com o algoritmo de retropropagação resiliente. Cinco comportamentos reativos foram modelados de acordo com as ações definidas, enumerados de 0 a 4. Em corredores, o comportamento de seguir parede foi implementado; em cruzamentos, bifurcações e curvas o comportamento de seguir parede foi usado para alinhar o robô e então aplicou-se uma velocidade constante e ângulo de rotação de acordo com a ação.

A sequência de ações encontrada pelo melhor indivíduo foi: seguir no corredor, seguir no corredor, seguir em frente, seguir no corredor, virar à direita, seguir no corredor, virar à direita, seguir no corredor, seguir em frente, virar à esquerda, seguir no corredor, virar à esquerda, seguir no corredor, seguir em frente, seguir no corredor, virar à esquerda, seguir no corredor, virar à direita, seguir no corredor, seguir em frente, seguir no corredor, virar à direita, seguir no corredor, virar à esquerda, virar à esquerda e seguir no corredor. O robô executou com sucesso essa sequência de ações usando a abordagem de navegação topológica proposta por Sales, Osório e Wolf (2011). Um vídeo mostrando essa navegação está disponível em <<https://youtu.be/rWA8gdt5kj0>>.

4.1.2 Resultados do AG II

Os experimentos computacionais verificam o desempenho do AG proposto considerando alguns aspectos como técnicas de cruzamento, tamanho da população e taxa de elitismo. Os experimentos foram realizados no mapa ilustrado na Figura 18, em que os nós inicial e final são indicados como S_1 e G_1 , respectivamente.

Os valores $\alpha = 2$, $\beta = 10$ e $\gamma = 2$ foram adotados para a função de aptidão. O tamanho do indivíduo varia entre 5 e 20 genes. O AG para a execução quando um número máximo de 1000 gerações é alcançado ou se 400 gerações são executadas sem atualizar a melhor aptidão. Esses valores foram definidos em testes prévios.

Figura 18 – Mapa usado nos experimentos. S_1 e G_1 são a posição inicial e final do robô, respectivamente.

Fonte: Elaborada pelo autor.

Na primeira análise, o cruzamento uniforme, de um ponto e de dois pontos foram avaliados. Os resultados são descritos na Tabela 9. As colunas representam a técnica de cruzamento, o número de gerações, o valor de aptidão do melhor indivíduo e o tempo de execução gasto em milissegundos, respectivamente. Esses resultados são valores de média aritmética seguidos pelo desvio padrão de 100 execuções do AG.

O tamanho da população foi definido como 100 e o elitismo não foi usado. É possível observar que o cruzamento de um ponto obteve uma melhor solução, embora o número de

gerações e o tempo gasto tenha sido maior que os outros. Assim, o cruzamento de um ponto foi definido para a próxima avaliação.

Tabela 9 – Análise da técnica de cruzamento.

Cruzamento	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
Uniforme	595.43 $\pm$ 196.01	1439.32 $\pm$ 795.41	5.91 $\pm$ 3.40
Um ponto	656.13 $\pm$ 211.87	1102.45 $\pm$ 558.00	6.00 $\pm$ 3.42
Dois pontos	608.49 $\pm$ 196.68	1506.19 $\pm$ 759.34	5.52 $\pm$ 3.15

Fonte: Dados da pesquisa.

O segundo parâmetro avaliado foi o tamanho da população. Os valores de tamanho da população testados foram 50, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800 e 2000. Os resultados na Tabela 10 indicam uma melhor solução para populações com muitos indivíduos, mas o tempo computacional também aumenta. A partir de 1000 indivíduos, uma solução com qualidade melhor é obtida em menos que 60 milissegundos, em média. Entretanto, o melhor resultado médio foi encontrado com 2000 indivíduos em 115.42 milissegundos, em média. Por isso, esses dois valores são avaliados nos próximos experimentos.

Tabela 10 – Análise do tamanho da população.

Tamanho da população	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
50	642.63 $\pm$ 206.69	1252.22 $\pm$ 645.11	3.17 $\pm$ 1.80
100	656.13 $\pm$ 211.87	1102.45 $\pm$ 558.00	6.00 $\pm$ 3.42
200	587.86 $\pm$ 196.65	881.68 $\pm$ 358.85	10.77 $\pm$ 6.09
400	595.78 $\pm$ 192.71	776.00 $\pm$ 248.00	21.23 $\pm$ 12.43
600	594.39 $\pm$ 168.32	707.80 $\pm$ 134.88	32.89 $\pm$ 18.32
800	598.20 $\pm$ 181.14	665.52 $\pm$ 113.26	42.74 $\pm$ 24.53
1000	618.38 $\pm$ 195.07	634.48 $\pm$ 57.57	57.33 $\pm$ 33.85
1200	630.14 $\pm$ 197.38	625.33 $\pm$ 41.72	69.66 $\pm$ 40.51
1400	630.51 $\pm$ 199.83	624.56 $\pm$ 45.36	84.49 $\pm$ 47.26
1600	581.93 $\pm$ 199.40	624.52 $\pm$ 44.31	87.98 $\pm$ 48.20
1800	614.65 $\pm$ 196.96	620.49 $\pm$ 31.58	102.90 $\pm$ 58.02
2000	609.39 $\pm$ 178.63	615.26 $\pm$ 18.53	115.42 $\pm$ 66.08

Fonte: Dados da pesquisa.

Três estratégias elitistas foram analisadas: todos os indivíduos são substituídos de uma geração para outra (0), apenas a melhor solução é mantida para a próxima geração (1) ou os 10% melhores indivíduos da população são mantidos para a próxima geração (100 ou 200). Os resultados foram obtidos com o cruzamento de um ponto e o tamanho da população de 1000 indivíduos são apresentados na Tabela 11 e de 2000 na Tabela 12.

Conclui-se que a estratégia elitista que mantém 10% dos melhores indivíduos é a melhor abordagem para ambos os valores de tamanho de populações. O AG com 1000 indivíduos é capaz de encontrar o melhor resultado com essa estratégia em um tempo computacional menor. A melhor solução encontrada nesses experimentos é ilustrada na Figura 18.

Tabela 11 – Análise da taxa de elitismo com tamanho de população 1000.

Taxa de elitismo	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
0	618.38 ± 195.07	634.48 ± 57.57	57.33 ± 33.85
1	629.45 ± 187.29	619.17 ± 49.61	58.83 ± 32.56
100	414.17 ± 7.56	595.63	56.09 ± 32.78

Fonte: Dados da pesquisa.

Tabela 12 – Análise da taxa de elitismo com tamanho da população 2000.

Taxa de elitismo	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
0	609.39 ± 178.63	615.26 ± 18.53	115.42 ± 66.08
1	599.20 ± 177.93	603.27 ± 13.34	112.89 ± 65.77
200	410.24 ± 5.91	595.63	117.16 ± 67.42

Fonte: Dados da pesquisa.

Na Tabela 13, os resultados para diferentes pontos de início e destino com os parâmetros estabelecidos são apresentados. O AG é executado com o cruzamento de um ponto, 1000 indivíduos e taxa de elitismo de 10%. O método proposto consome quase o mesmo tempo computacional e número de gerações para as diferentes posições de início e objetivo. Além disso, ele sempre encontra a mesma solução final, exceto para o início S_3 e objetivo G_3 . Entretanto, nesse caso, o desvio padrão da melhor aptidão é menor que 1.

Tabela 13 – Resultados obtidos por todos os pontos de início e destino indicados na Figura 18.

Início	Destino	Gerações para convergir	Melhor aptidão	Tempo (milissegundos)
S_1	G_1	414.17 ± 7.56	595.63	56.09 ± 32.78
S_1	G_2	405.98 ± 4.49	504.25	55.58 ± 31.93
S_1	G_3	400.01 ± 0.1	381.79	53.59 ± 30.78
S_1	G_4	409.17 ± 7.16	552	55.66 ± 31.86
S_2	G_1	404.73 ± 3.88	520.57	52.07 ± 29.84
S_2	G_2	400.53 ± 1.04	429.19	49.24 ± 28.21
S_2	G_3	400.12 ± 0.43	423.77	49.32 ± 28.27
S_2	G_4	401.37 ± 1.82	477	50.79 ± 29.13
S_3	G_1	400.04 ± 0.24	439.35	50.56 ± 29.00
S_3	G_2	400	320.83	39.23 ± 22.41
S_3	G_3	505.07 ± 113.01	670.68 ± 0.94	67.84 ± 39.84
S_3	G_4	400.99 ± 1.47	490.12	52.35 ± 29.98

Fonte: Dados da pesquisa.

O caminho ilustrado na Figura 18 foi executado no simulador Player/Stage, com o robô simulado Pioneer 3-AT e o sensor 180° SICK Lidar. O robô começa em um corredor. Quando o reconhecedor de estado indica uma mudança de estado, o robô busca um ponto de reconhecimento, que é o ponto onde o robô está fora do corredor. Nesse momento, o estado em que o robô está é identificado, ou seja, identifica quais são as áreas livres: leste, nordeste, norte, noroeste ou oeste. Em seguida, a ação indicada pelo AG é ativada.

O robô executa o comportamento de seguir parede quando está se movendo no corredor. Quando uma ação é ativada, um comportamento reativo é executado, no qual o robô usa uma bússola para definir a orientação e as curvas são feitas com velocidades linear e angular definidas.

O reconhecedor de padrão proposto é utilizado em toda a execução do caminho encontrado pelo AG. A navegação para o ponto de início S_1 e objetivos G_1, G_2, G_3 e G_4 podem ser visualizados em <https://www.youtube.com/watch?v=FYhfcvYPyE>, https://www.youtube.com/watch?v=0_3TnlGLq_8, <https://www.youtube.com/watch?v=YcLic2vJam8&feature=youtu.be> e <https://www.youtube.com/watch?v=wl5viepZ-4E>, respectivamente.

4.1.3 Resultados do AG III

O AG proposto foi avaliado considerando a seleção de alguns de seus operadores e parâmetros. Depois, a melhor configuração do AG foi comparada com os algoritmos A* e C*, mencionados na Seção 3.1.3. Esses experimentos foram executados usando a representação em grafo.

Inicialmente, foi definido o tamanho da sequência de ações em 5, taxa de mutação 1, tamanho da população 5 e seleção por torneio com 2 indivíduos. O parâmetro de penalidade foi definido em $\alpha = 10^4$ na função de aptidão. A execução é finalizada quando todos os robôs alcançam o objetivo ou o número máximo de 1000 gerações é executado. Esses valores iniciais foram definidos com base em testes prévios. O total de 100 execuções foram realizadas usando o mapa da Figura 19, em que as posições de início e destino são indicadas por S_1 e G_1 , respectivamente.

A primeira avaliação se refere aos operadores de cruzamento, cujos resultados são mostrados na Tabela 14. A Coluna *Média da melhor aptidão* é a média aritmética dos melhores valores de aptidão, a Coluna *Gerações para convergir* é a média aritmética do número de gerações necessário para que todos os robôs alcançassem o objetivo e a Coluna *Média do melhor tempo* é a média aritmética do tempo gasto pelo primeiro robô que alcançou o objetivo (melhor solução).

Tabela 14 – Análise da técnica de cruzamento.

Técnica de cruzamento	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
Uniforme	30857.72 ± 16146.39	225.56 ± 76.08	0.15 ± 0.11
Um ponto	26832.68 ± 14499.01	230.16 ± 82.17	0.13 ± 0.08
Dois pontos	27285.76 ± 16757.34	227.82 ± 91.38	0.14 ± 0.10
Aleatório	29458.78 ± 19631.68	227.56 ± 97.78	0.15 ± 0.12

Fonte: Dados da pesquisa.

A partir dos valores obtidos, não houve um operador que tenha apresentado melhor resultado, sendo que todas as configurações tem desempenho similar. O cruzamento de um ponto foi escolhido para os próximos experimentos por apresentar o menor valor médio da melhor

Figura 19 – Mapa usado nos experimentos do algoritmo AG III.

Fonte: Elaborada pelo autor.

aptidão associado com o pouco tempo para encontrar a melhor aptidão. Apesar de não estar apresentado na Tabela 14, vale a pena mencionar que todos os robôs alcançaram o objetivo nas 100 execuções.

Foram testados tamanhos iniciais diferentes para a sequência de ações (l) na Tabela 15. A aptidão média com valores grandes não permitem definir o melhor valor de l . Decidiu-se pelo valor $l = 40$, que apresentou um melhor compromisso entre pequeno número médio de gerações e curto tempo computacional. Os robôs também alcançaram o objetivo para todos os valores de l testados.

Os resultados para diferentes valores da taxa de mutação são apresentados na Tabela 16, onde o método apresenta melhor desempenho quando o operador de mutação é sempre aplicado.

A Tabela 17 mostra que os valores de média e desvio padrão da aptidão melhora com o aumento da população, então o valor 20 foi escolhido para o tamanho da população. Os robôs alcançam o objetivo em todos os valores de tamanho.

Os resultados da seleção por torneio com 2, 4, 6, 8 e 10 indivíduos são mostrados na

Tabela 15 – Análise do impacto do tamanho inicial sobre a sequência do indivíduo em crescimento.

Tamanho do incremento	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
5	26832.68 ± 14499.01	230.16 ± 82.17	0.13 ± 0.08
10	31302.15 ± 21127.48	134.74 ± 50.51	0.08 ± 0.07
15	30353.03 ± 17708.67	89.00 ± 32.25	0.05 ± 0.03
20	28909.65 ± 20336.32	81.78 ± 31.34	0.05 ± 0.04
25	31425.24 ± 19523.74	54.21 ± 20.47	0.03 ± 0.02
30	34587.81 ± 22801.14	45.61 ± 19.95	0.03 ± 0.02
35	30981.72 ± 24711.52	42.04 ± 19.53	0.02 ± 0.02
40	30367.34 ± 21314.06	35.05 ± 13.40	0.02 ± 0.02
45	30848.48 ± 24009.31	35.22 ± 13.41	0.02 ± 0.02
50	31463.04 ± 25864.99	31.41 ± 14.92	0.02 ± 0.02

Fonte: Dados da pesquisa.

Tabela 16 – Análise da taxa de mutação.

Taxa de mutação	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
0.01	145158.78 ± 148287.72	183.62 ± 82.36	0.20 ± 0.47
0.05	155898.96 ± 149554.29	209.83 ± 88.29	0.21 ± 0.60
0.1	159479.25 ± 163966.55	206.16 ± 92.00	0.28 ± 0.64
0.5	44662.92 ± 35257.16	54.18 ± 22.31	0.03 ± 0.03
1	30367.34 ± 21314.06	35.05 ± 13.40	0.02 ± 0.02

Fonte: Dados da pesquisa.

Tabela 17 – Análise do tamanho da população.

Tamanho da população	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
5	30367.34 ± 21314.06	35.05 ± 13.40	0.02 ± 0.02
10	21289.88 ± 13317.01	39.04 ± 11.38	0.04 ± 0.03
15	16133.63 ± 9602.21	43.50 ± 13.67	0.05 ± 0.04
20	14408.04 ± 7694.13	46.86 ± 12.52	0.07 ± 0.04

Fonte: Dados da pesquisa.

Tabela 18. O torneio com dois indivíduos gasta menos gerações para convergir e é o único em que todos os robôs chegaram ao objetivo em todas as execuções, como pode ser visto na coluna *Robôs no objetivo*. Assim, o torneio de 2 é fixado para as próximas avaliações.

A seguir, o AG foi executado para diferentes posições de início e destino ($S_1, S_2, S_3; G_1, G_2, G_3$), mostrados na Figura 19. Nesses experimentos, o AG é sempre executado com $l = 40$, cruzamento de um ponto, taxa de mutação 1, tamanho de população 20 e torneio de 2. A Tabela 19 mostra a média dos resultados depois de 100 execuções do algoritmo para esses pontos de início e fim, indicados pela Coluna $S - G$. Os robôs conseguiram alcançar os objetivos em todas as execuções.

Tabela 18 – Análise do método de seleção.

Método de seleção	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo	Robôs no objetivo
Torneio de 2	14408.04 ± 7694.13	46.86 ± 12.52	0.07 ± 0.04	20
Torneio de 4	13662.63 ± 7461.15	387.59 ± 230.46	0.06 ± 0.07	19.99 ± 0.10
Torneio de 6	14006.10 ± 7004.62	1000	0.08 ± 0.10	18.89 ± 0.31
Torneio de 8	13598.74 ± 7775.23	1000	0.07 ± 0.07	17.91 ± 0.35
Torneio de 10	14149.98 ± 9139.12	1000	0.13 ± 0.39	16.82 ± 0.46

Fonte: Dados da pesquisa.

Tabela 19 – Diferentes posições de início e destino.

S-G	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
1-1	14408.04 ± 7694.13	46.86 ± 12.52	0.07 ± 0.04
1-2	13644.98 ± 8741.68	58.23 ± 17.51	0.05 ± 0.04
1-3	15843.37 ± 8803.28	67.95 ± 16.95	0.06 ± 0.05
2-1	8024.73 ± 6317.97	39.83 ± 13.55	0.03 ± 0.02
2-2	9622.97 ± 6706.14	51.98 ± 16.24	0.03 ± 0.02
2-3	12056.61 ± 7443.25	64.08 ± 18.23	0.04 ± 0.03
3-1	11576.61 ± 6227.41	47.07 ± 14.50	0.04 ± 0.03
3-2	10244.78 ± 7284.58	54.31 ± 13.96	0.04 ± 0.02
3-3	13821.54 ± 10146.31	67.11 ± 20.25	0.05 ± 0.04

Fonte: Dados da pesquisa.

Os melhores resultados obtidos pelo AG foram de S_2 para G_1 e G_2 . Nesses casos, observa-se que há mais linhas retas conectando S_2 para G_1 e G_2 que permitem que o robô evite as regiões mais complexas do meio do mapa. Por outro lado, os caminhos de S_1 e S_3 tem mais desvios que podem levar os robôs a explorarem o meio do mapa.

A Tabela 20 mostra os resultados obtidos pelos algoritmos A* e C* executados para diferentes posições de início e fim no mesmo mapa. A Coluna *Tamanho do caminho* mostra a distância gasta pelos algoritmos nos seus caminhos, incluindo as distâncias percorridas durante os retrocessos e a Coluna *Tempo* é o tempo em segundos gasto pelos algoritmos. As Colunas *Melhor tamanho do caminho*, *Melhor tempo*, *Pior tamanho do caminho* e *Pior tempo* geraram esses valores para a melhor e a pior solução do AG, considerando 100 execuções.

O algoritmo A* supera o C* nas situações S_2 para G_1 , G_2 e G_3 , e é também melhor de S_1 para G_2 . Isso faz sentido considerando que a exploração para alcançar esses objetivos tem menos armadilhas para o A*, portanto, menos necessidade de retrocessos. Assim o C* apresenta desempenho melhor em áreas onde mais retrocessos devem ser executados: de S_1 para G_1 , G_3 e de S_3 para G_1 , G_2 , G_3 . Entretanto, a melhor sequência de ações do AG retorna um caminho cujo custo é sempre melhor que os algoritmos A* e C*, enquanto as piores soluções do AG levam o robô a percorrer distâncias maiores que o A* e o C*.

Embora o pior tamanho do caminho do AG seja maior que os tamanhos obtidos pelos

Tabela 20 – Resultados obtidos pelos algoritmos A*, C* e AG para diferentes posições de início e destino, mostrado na Figura 19.

S-G	A*		C*		AG			
	Tamanho do caminho	Tempo	Tamanho do caminho	Tempo	Melhor tamanho do caminho	Melhor tempo	Pior tamanho do caminho	Pior tempo
1-1	45280.79	0.08	27782.95	0.07	2274.31	0.02	45071.13	0.16
1-2	32131.09	0.05	51841.34	0.11	1754.73	0.01	44286.19	0.12
1-3	51457.50	0.08	31292.88	0.07	2258.97	0.02	42198.94	0.22
2-1	13620.92	0.03	15831.04	0.04	1704.65	0.01	32323.74	0.10
2-2	32450.60	0.06	42404.60	0.10	1652.06	0.01	40425.91	0.14
2-3	30934.27	0.06	56030.89	0.14	2377.52	0.02	38511.96	0.22
3-1	34728.54	0.06	14869.33	0.05	1963.24	0.01	36969.64	0.10
3-2	48322.65	0.08	45930.42	0.11	2072.22	0.01	46529.57	0.14
3-3	64915.37	0.08	42002.14	0.10	2904.60	0.02	59931.26	0.19

Fonte: Dados da pesquisa.

algoritmos A* e C*, isso acontece poucas vezes, como mostrado na Figura 20. O eixo das abcissas indica a porcentagem de soluções retornadas pelo AG que são melhores que as soluções retornadas pelo A* e C*, em que o AG retorna melhores tamanhos do caminho que o A* e o C* em mais que 80% dos casos para a maioria dos resultados, depois de 100 execuções.

Figura 20 – Porcentagem de soluções do AG que são melhores que os algoritmos A* e C*.

Fonte: Elaborada pelo autor.

Esses métodos são comparados em outros dois mapas, apresentados nas Figuras 21 e 22. Esses mapas tem uma armadilha, onde os caminhos pelo meio do mapa parecem estar próximos ao objetivo, mas não há conexão com o objetivo por esses caminhos. Os algoritmos A* e C*

consideram a distância euclidiana na função heurística para escolher o próximo nó a ser avaliado, então, provavelmente ele explorará o meio do mapa antes de encontrar o caminho certo. O AG pode lidar melhor com essa situação porque usa uma escolha aleatória quando define uma nova ação durante o uso do operador de reparo e mutação.

Na Tabela 21 a média dos resultados encontrados pelo AG nos mapas das Figuras 21 e 22 são apresentados, onde todos os robôs sempre encontram o destino. Os resultados do A* e C* nesses mapas são mostrados na Tabela 22. Para o mapa na Figura 21, todas as execuções do AG atingiram menores tamanhos que os algoritmos A* e C* em curto tempo computacional. No mapa da Figura 22, o AG obteve menor tamanho que o A* em 99% dos casos e que o C* em 92% dos casos.

Figura 21 – Mapa 2. *S* é o ponto de início e *G* é o objetivo.

Fonte: Elaborada pelo autor.

Tabela 21 – Resultados obtidos pelo AG nos mapas mostrados nas Figuras 21 e 22.

Mapa	Média da melhor aptidão	Gerações para convergir	Média do melhor tempo
2	10040.29 ± 7310.81	143.58 ± 45.54	0.10 ± 0.62
3	20278.02 ± 15441.98	139.53 ± 37.35	0.12 ± 0.11

Fonte: Dados da pesquisa.

A navegação do caminho obtido pelo AG proposto para o ambiente da Figura 19, de S_1 para G_1 , foi executada usando o simulador robótico Player/Stage (COLLETT; MACDONALD;

Tabela 22 – Resultados obtidos pelos algoritmos A* e C* nos mapas apresentados nas Figuras 21 e 22.

Map	A*		C*		AG			
Mapa	Tamanho do caminho	Tempo	Tamanho do caminho	Tempo	Melhor tamanho do caminho	Melhor Tempo	Pior tamanho do caminho	Pior tempo
2	85697.93	0.11	66195.01	0.11	1681.84	0.01	40019.44	0.11
3	63137.15	0.16	47063.00	0.10	2751.85	0.02	68972.28	0.26

Fonte: Dados da pesquisa.

Figura 22 – Mapa 3. S é o ponto de início e G é o objetivo.

Fonte: Elaborada pelo autor.

GERKEY, 2005). O reconhecedor de estado descrito por Santos V. C.; Toledo e Osório (2014) foi usado para identificar quando o estado em que o robô está muda. Nesse caso, o estado padrão do robô é corredor, sendo que há uma mudança quando o robô encontra alguma bifurcação. Nesse momento, uma ação da sequência de ações encontrada é executada. A navegação do robô com a sequência de ações do AG pode ser vista em <<http://youtu.be/d-X1Q12krIA>> e <<http://youtu.be/as5lp5fU-3o>>.

4.2 Algoritmos de Otimização por Colônia de Formiga

Nesta seção são apresentados e discutidos os resultados dos algoritmos de otimização por colônia de formigas propostos.

4.2.1 Algoritmo Max-Min Ant System for Static Path Planning

Nesta seção, são apresentados a configuração experimental e os resultados da aplicação do algoritmo MMAS-SPP. Os resultados são comparados com os obtidos pelo algoritmo AG III. Assim, o mesmo mapa foi adotado para realizar os experimentos, apresentado na Figura 23, no qual o algoritmo também foi executado 100 vezes.

Figura 23 – Mapa utilizado nos experimentos do algoritmo MMAS-SPP

Fonte: Elaborada pelo autor.

Nas tabelas a seguir 23 e 24, *primeiro objetivo* se refere à melhor distância percorrida pela formiga que alcançou o objetivo pela primeira vez na primeira iteração do algoritmo, de todas as execuções; *Média primeiro objetivo* é a média de *primeiro objetivo* em todas execuções; *Média da melhor distância percorrida* é a média aritmética da distância percorrida pela formiga que encontrou a melhor solução de todas as execuções do algoritmo; *Média da distância por iteração* se refere à média da distância percorrida por cada formiga em uma iteração; *porcentagem* se refere a quantas vezes o algoritmo encontrou a melhor solução entre as 100 execuções; e *Média do melhor caminho* é a média aritmética das melhores soluções encontradas pelo algoritmo.

A Tabela 23 mostra a análise do primeiro critério de parada: a melhor solução não é atualizada por um certo número de iterações. Os outros parâmetros do algoritmo foram definidos como: número máximo de iterações = 1000; $\alpha = 1$; $\beta = 0.1$; número de formigas = 20, $a = 10$. Número máximo de iterações é o segundo critério de parada, α e β são parâmetros da Equação

3.7, a é usado para definir a quantidade mínima de feromônio.

O número de iterações não tem influência na primeira iteração. Quanto maior é o número de iterações, maior é a distância percorrida para encontrar a melhor solução, embora melhores soluções sejam encontradas. A distância média por iteração mostra que as melhores soluções são encontradas com o número maior de iterações. Baseado nesses resultados, o número de iterações 50 foi fixado porque apresenta um melhor compromisso entre a qualidade da solução e a média da distância percorrida em relação aos outros valores.

Tabela 23 – Análise do número de iterações no critério de parada.

Número de iterações	Primeiro objetivo	Média do primeiro objetivo	Média da melhor distância percorrida	Média da distância por iteração	Porcentagem	Média da melhor solução
5	1424.54	4762.11 $\pm$ 1773.90	144288.83 $\pm$ 80948.04	17625.68 $\pm$ 1998.52	10%	1205.31 $\pm$ 63.64
10	1618.13	4585.44 $\pm$ 1607.36	210664.12 $\pm$ 102608.24	14705.20 $\pm$ 1353.40	31%	1164.64 $\pm$ 72.22
20	1427.88	4667.01 $\pm$ 1848.38	249341.84 $\pm$ 128513.64	12257.86 $\pm$ 961.51	42%	1161.34 $\pm$ 70.74
30	1836.58	4588.88 $\pm$ 1416.40	302796.47 $\pm$ 171794.79	11279.91 $\pm$ 996.36	47%	1151.78 $\pm$ 62.28
50	1765.57	4741.35 $\pm$ 1906.60	372306.82 $\pm$ 206199.95	10048.43 $\pm$ 693.50	56%	1136.22 $\pm$ 57.86
100	1634.46	4545.71 $\pm$ 1863.68	621405.83 $\pm$ 522241.55	8991.95 $\pm$ 462.52	72%	1118.07 $\pm$ 45.17

Fonte: Dados da pesquisa.

No próximo experimento, a influência do número de formigas é investigado. A Tabela 24 mostra os resultados dos experimentos. Os outros parâmetros do algoritmo foram definidos como número máximo de iterações = 1000; $\alpha = 1$; $\beta = 0.1$; número de iterações = 50; $a = 10$. Uma grande quantidade de formigas não melhora significativamente a primeira vez que uma formiga alcança o objetivo (*primeiro objetivo*) mas melhora a média desse valor em *média do primeiro objetivo*. É interessante notar que usando mais formigas, a média da distância percorrida para encontrar a melhor solução diminui. Isso sugere que o número maior de formigas facilita encontrar o objetivo por causa do trabalho colaborativo. A partir de cinco formigas, o valor de *média da distância percorrida por iteração* é quase o mesmo, ou seja, as formigas gastam aproximadamente a mesma distância para deixar a posição inicial e alcançar o objetivo. Além disso, com cinco formigas esse valor da média é menor do que a média dos melhores valores alcançados pelo AG III, como pode ser observado na primeira linha da Tabela 25.

Quanto maior é o número de formigas, o algoritmo MMAS pode melhorar a qualidade das soluções. Usando 100 formigas, a solução ótima foi sempre encontrada. Apesar disso, o número de formigas foi fixado como 20 porque ele fornece o menor valor da distância média por iteração e é o mesmo valor usado como tamanho da população pelo AG III, que torna a comparação entre os algoritmos mais justa.

O algoritmo AG III foi avaliado em um cenário onde os robôs deveriam alcançar o objetivo o mais rápido possível, baseado na menor distância percorrida de cada execução do algoritmo. Os resultados do AG III também foram comparados com o algoritmo A* (HART; NILSSON; RAPHAEL, 1968), método de busca heurística bem estabelecido (RUSSELL; NORVIG, 2009). No algoritmo A*, cada nó é avaliado para ser expandido, de acordo com a função

Tabela 24 – Análise do número de formigas.

Número de formigas	Primeiro objetivo	Média do primeiro objetivo	Média da melhor distância percorrida	Média da distância por iteração	Porcentagem	Média da melhor solução
1	3288.25	31507.21 ± 23980.87	815297.95 ± 504828.20	15670.93 ± 2406.07	0	1258.38 ± 46.60
5	2412.07	9205.97 ± 4596.54	406735.76 ± 254652.78	10453.24 ± 1031.97	6%	1216.92 ± 51.68
10	1648.65	6635.75 ± 3524.54	332228.36 ± 205154.26	10133.92 ± 772.10	25%	1175.57 ± 64.33
20	1487.06	4702.57 ± 1801.84	389604.33 ± 239650.44	10065.26 ± 675.70	60%	1132.06 ± 59.33
30	1742.55	4354.05 ± 1805.32	318430.24 ± 180433.27	10097.23 ± 586.22	78%	1115.26 ± 46.14
40	1369.32	3968.88 ± 1310.82	313190.23 ± 180117.21	10114.84 ± 515.35	90%	1103.53 ± 34.13
50	1498.35	3377.00 ± 1134.47	294487.96 ± 156961.04	10141.91 ± 421.27	95%	1097.39 ± 18.20
100	1527.82	3049.75 ± 681.70	187401.35 ± 79433.51	10182.05 ± 221.74	100%	1093.27

Fonte: Dados da pesquisa.

$f(n) = g(n) + h(n)$. A função $g(n)$ é o custo real de deixar a posição inicial e chegar ao nó atual n ; $h(n)$ é uma função heurística que estima o custo de deixar o nó atual n e chegar ao objetivo.

A Tabela 25 mostra os resultados do algoritmo A*, AG III e MMAS para diferentes posições de início e destino no mapa ilustrado pela Figura 19. *S-G* são os pontos de início e destino; *distância solução* é a distância percorrida pelo A* para encontrar a melhor solução e chegar ao objetivo; *melhor distância solução* é melhor distância percorrida pelo AG III de 100 execuções; *pior distância solução* é pior distância percorrida pelo AG III de 100 execuções; *média melhor aptidão* é média aritmética da melhor distância percorrida de 100 execuções do AG III; *primeiro objetivo* e *último objetivo* mostram a melhor e a pior distância percorrida do algoritmo MMAS para encontrar o objetivo; *média primeiro objetivo* mostra a média aritmética da melhor distância percorrida na primeira iteração de 100 execuções do algoritmo MMAS; e *média distância por iteração* mostra a média da distância percorrida por cada formiga em cada iteração.

Tabela 25 – Resultados dos algoritmos A*, AG e MMAS para as diferentes posições de início e objetivo mostradas na Figura 5.

	A*	AG			MMAS			
S-G	Distância solução	Melhor distância solução	Pior distância solução	Média melhor aptidão	Primeiro objetivo	Último objetivo	Média primeiro objetivo	Média distância por iteração
1-1	45280.79	2274.31	45071.13	14408.04 ± 7694.13	1487.06	11140.33	4702.57 ± 1801.84	10065.26 ± 675.70
1-2	32131.09	1754.73	44286.19	13644.98 ± 8741.68	1286.81	9376.22	4762.40 ± 1901.44	8921.65 ± 314.44
1-3	51457.50	2258.97	42198.94	15843.37 ± 8803.28	1352.34	12442.80	5055.63 ± 2377.28	9687.88 ± 442.94
2-1	13620.92	1704.65	32323.74	8024.73 ± 6317.97	1379.28	9053.54	3392.73 ± 1458.43	8596.14 ± 666.91
2-2	32450.60	1652.06	40425.91	9622.97 ± 6706.14	1556.17	12724.38	4462.96 ± 1875.78	10260.56 ± 712.98
2-3	30934.27	2377.52	38511.96	12056.61 ± 7443.25	1613.36	13864.48	5071.99 ± 2336.05	10687.96 ± 744.64
3-1	34728.54	1963.24	36969.64	11576.61 ± 6227.41	1727.32	9516.46	4359.05 ± 1570.80	10583.75 ± 0.59
3-2	48322.65	2072.22	46529.57	10244.78 ± 7284.58	1694.27	16885.51	5040.17 ± 2535.35	10422.69 ± 883.81
3-3	64915.37	2904.60	59931.26	13821.54 ± 10146.31	1419.51	18481.61	5276.38 ± 2868.52	11244.90 ± 874.20

Fonte: Dados da pesquisa.

Os algoritmos também foram avaliados em dois outros mapas, com armadilha, ilustrados nas Figuras 21 e 22. Esses mapas tem uma armadilha porque o caminho pelo meio do mapa parece menor, considerando a distância euclidiana, mas não há conexão com o objetivo. A Tabela 26 apresenta os resultados dos algoritmos A*, AG III e MMAS nesses mapas.

Como pode ser observado, o MMAS alcançou o objetivo na primeira iteração percorrendo uma distância melhor do que o AG III em ambos os mapas. Esse resultado é inesperado em uma armadilha porque o MMAS usa a distância euclidiana para o objetivo como heurística, assim como o A*, o MMAS também poderia cair na armadilha, principalmente na primeira iteração, quando a quantidade de feromônio é a mesma em todas as arestas. Mas mesmo com a mesma quantidade, a influência do feromônio na escolha probabilística ajudou o algoritmo a balancear a decisão do próximo nó, durante a fase de construção. O MMAS também obteve resultados melhores na média do primeiro objetivo do que a média da melhor distância percorrida pelo AG III (Coluna *Média melhor aptidão*) e a distância percorrida pelo algoritmo A*. Além disso, o MMAS pode encontrar a solução ótima em todas as execuções nesses mapas.

Tabela 26 – Resultados dos algoritmos A*, AG e MMAS para os mapas 2 e 3 (Figuras 21 e 22).

Mapa	A*	AG			MMAS			
	Distância	Melhor distância solução	Pior distância solução	Média melhor aptidão	Primeiro objetivo	Último objetivo	Média primeiro objetivo	Média distância iteração
2	85697.93	1681.84	40019.44	10040.29 ± 7310.81	1471.31	14437.46	3755.42 ± 2376.22	12487.02 ± 744.63
3	63137.15	2751.85	68972.28	20278.02 ± 15441.98	2036.38	26634.66	6425.56 ± 4339	13132.97 ± 593.21

Fonte: Dados da pesquisa.

A navegação do melhor caminho obtido no mapa da Figura 19, entre S_1 e G_1 , foi executado no simulador robótico Player/Stage (COLLETT; MACDONALD; GERKEY, 2005), usando o robô Pioneer P3-AT robot com um sensor SICK Lidar 180°. O caminho fornecido pela nossa abordagem pode ser representada por uma sequência de ações, como proposto anteriormente. Na navegação desse caminho, o robô começa navegando em um corredor. Quando ele detecta uma mudança de estado, ou seja, quando o robô sai do corredor, a próxima ação é ativada. O reconhecedor de estado descrito por Santos V. C.; Toledo e Osório (2014) foi aplicado para identificar quando o estado do robô muda, baseado nos sensores do robô. O robô não possui informação prévia do ambiente, ele apenas conhece a sequência de ações que ele deve executar. As sequências de ações são: *virar à leste, virar à oeste, virar à leste, seguir para o norte, seguir para o norte, seguir para o norte, virar à oeste, seguir para o norte, virar à leste, virar à oeste, seguir para o norte, seguir para o norte, virar à leste, virar à oeste, seguir para o norte, seguir para o norte, virar à leste, virar à oeste, virar à leste, seguir para o norte, virar à oeste, virar à leste, virar à oeste, virar à leste, virar à oeste, virar à leste, seguir para o norte and virar à oeste*. O vídeo do robô executando o melhor caminho no mapa da Figura 19, de S_1 para G_1 , pode ser visto em <<https://youtu.be/U1JICygoHvI>>.

4.2.2 Algoritmo Max-Min Ant System for Dynamic Path Planning

O algoritmo MMAS-DPP é aplicado em duas etapas. Primeiro, ele encontra a solução no ambiente estático. Então uma mudança no ambiente ocorre e o algoritmo deve encontrar a nova solução. Dois tipos de mudança foram avaliados: remover uma aresta da melhor solução

encontrada na primeira etapa e mudar a posição do objetivo. Primeiro são apresentados os experimentos para ajustar os parâmetros no ambiente estático. Então, o comportamento do algoritmo no ambiente dinâmico é avaliado.

O mapa adotado é o mesmo utilizado pelo algoritmo MMAS-SPP, ilustrado na Figura 23. S_1 e G_1 são os pontos inicial e final, respectivamente. O algoritmo foi executado 100 vezes nesse mapa. Os resultados são apresentados em termos de média aritmética e desvio padrão.

A Tabela 27 apresenta os resultados relativos ao número de formigas. A distância percorrida é a distância gasta por uma formiga durante toda a execução do algoritmo. Cada vez que uma formiga se move de um nó para outro, a distância entre esses nós é somada à distância percorrida da formiga correspondente. A Coluna *porcentagem* se refere a quantas vezes a melhor solução foi encontrada entre as 100 execuções do algoritmo. A Coluna *melhor caminho* mostra a média e o desvio padrão da melhor solução encontrada pelas execuções. *Melhor distância percorrida* é a média da distância percorrida pela formiga que encontrou a melhor solução em todas as execuções durante a execução completa do algoritmo; *distância por iteração* se refere à média da distância percorrida por cada formiga em uma iteração.

Os outros parâmetros do algoritmo foram definidos como se segue. Para o critério de parada, o número fixo de iterações sem atualização da melhor solução é 100 e o número máximo de iterações é 1000; $\alpha = 1$ e $\beta = 0.1$ são parâmetros da Equação 3.7, que define a probabilidade de uma formiga se mover para o próximo nó; $\rho = 0.1$ é a taxa de evaporação da Equação 3.8; $a = 10$ é o parâmetro que define a quantidade mínima de feromônio, conforme a Equação 3.11. Na Tabela 27, pode-se observar que a *melhor distância percorrida* diminui usando mais formigas. O número de formigas não afeta a distância percorrida por iteração. O algoritmo foi capaz de encontrar sempre a solução ótima quando usa 50 formigas, por isso, esse valor foi definido para os próximos experimentos.

Tabela 27 – Análise do número de formigas.

Nº de formigas	Porcentagem	Melhor caminho	Melhor distância percorrida	Distância por iteração
10	27%	1168.38 $\pm$ 58.52	556975.88 $\pm$ 414802.66	8597.80 $\pm$ 696.51
20	72%	1119.80 $\pm$ 48.28	605291.47 $\pm$ 472561.49	8656 $\pm$ 453.74
30	95%	1096.78 $\pm$ 16.83	495381.75 $\pm$ 344283.67	8696.57 $\pm$ 340.20
40	99%	1094.02 $\pm$ 7.41	401832.06 $\pm$ 279262.32	8674.64 $\pm$ 235.36
50	100%	1093.27	275602.66 $\pm$ 167281.54	8741.87 $\pm$ 184.81
100	100%	1093.27	176561.11 $\pm$ 83673.91	8676.45 $\pm$ 122.92

Fonte: Dados da pesquisa.

A Tabela 28 apresenta a análise da taxa de evaporação ρ . Apesar dos resultados mostrarem que esse parâmetro não afeta muito o desempenho do algoritmo, a convergência dele é lenta quando o seu valor é muito baixo e o algoritmo pode convergir rapidamente para uma solução local mínima quando esse valor é muito alto. Baseado nos resultados, o valor $\rho = 0.1$ foi adotado para as próximas avaliações.

Tabela 28 – Taxa de evaporação.

Taxa de evaporação	Porcentagem	Melhor caminho	Melhor distância percorrida	Distância por iteração
0.01	99%	1094.02 ± 7.41	1354969.20 ± 475963.15	15479.11 ± 937.69
0.05	99%	1093.33 ± 0.60	413470.75 ± 238790.32	9900.13 ± 322.03
0.1	100%	1093.27	275602.66 ± 167281.54	8741.87 ± 184.81
0.3	100%	1093.27	188046.59 ± 116356.69	8023.37 ± 122.41
0.5	100%	1093.27	179750.01 ± 138014.02	7794.65 ± 110.69
0.6	99%	1094.16 ± 8.80	207929.42 ± 182693.14	7750.33 ± 111.34

Fonte: Dados da pesquisa.

Como explicado no Capítulo 3, MMAS-DPP define uma variação para a quantidade de feromônio para evitar estagnação, sendo que a quantidade mínima é definida como $\tau_{min} = \tau_{max}/a$. Quanto maior é a variação da quantidade de feromônio, mais lenta é a convergência do algoritmo, favorecendo a exploração do espaço de busca. Por outro lado, quanto menor é a variação da quantidade de feromônio, maior é chance do algoritmo explorar as melhores soluções encontradas. Os resultados mostrados na Tabela 29 refletem isso, em que a primeira coluna é o parâmetro a .

A qualidade dos caminhos é melhor com valores menores do parâmetro a . Isso é esperado porque o espaço de busca é mais restrito e o algoritmo tende a explorar as melhores soluções. Entretanto, nesses casos, as distâncias percorridas são maiores e o algoritmo tenderá a manter essas soluções, mesmo quando ocorre uma mudança no ambiente. Por isso, o valor 50 foi escolhido, que é um valor mediano e torna o algoritmo mais flexível para lidar com ambientes dinâmicos.

Tabela 29 – Quantidade mínima de feromônio.

Parâmetro a	Melhor caminho	Média da melhor distância	Distância por iteração
10	1094.15 ± 8.80	288731.99 ± 173807.23	8727.10 ± 201.08
20	1094.75 ± 10.55	216364.05 ± 123843.57	5750.01 ± 168.68
30	1095.92 ± 15.09	235027.81 ± 141441.83	5729.05 ± 186.61
40	1109.64 ± 43.30	202750.63 ± 124771.63	4494.66 ± 166.49
50	1111.47 ± 42.84	194960.87 ± 100896.87	4307.19 ± 174.98
60	1112.00 ± 42.94	195965.80 ± 109504.77	4158.63 ± 185.81
70	1117.88 ± 51.59	166252.38 ± 105274.34	4098.74 ± 161.54
80	1112.15 ± 39.69	183546.99 ± 102434.58	4007.26 ± 186.89
90	1112.54 ± 43.04	183388.60 ± 85755.18	3999.54 ± 153.66
100	1115.38 ± 47.76	169666.89 ± 87023.35	3985.88 ± 174.22

Fonte: Dados da pesquisa.

Os resultados do algoritmo MMAS-DPP foram comparados com o algoritmo D* Lite, descrito no Capítulo 2. A Tabela 30 mostra a comparação entre D* Lite e MMAS-DPP antes da mudança do ambiente, em relação à distância percorrida. A melhor distância percorrida e a distância média percorrida por iteração do MMAS-DPP é menor do que a distância percorrida pelo algoritmo D* Lite, embora a média da melhor distância percorrida não seja. Deste modo, o

MMAS-DPP pode encontrar um caminho adequado a uma distância percorrida menor do que o algoritmo D* Lite.

Tabela 30 – Resultados obtidos pelos algoritmos D* Lite e MMAS-DDP no ambiente estático.

D* Lite	MMAS		
Distância percorrida	Melhor distância	Média da melhor distância	Distância por iteração
36135.20	6935.26	194960.87 ± 100896.87	4307.19 ± 174.98

Fonte: Dados da pesquisa.

Depois de definir os parâmetros no ambiente estático, a análise de como o algoritmo lida com ambientes dinâmicos foi realizada. O MMAS-DPP encontra a melhor solução no ambiente estático. O ambiente muda e o algoritmo MMAS-DPP encontra a melhor solução para a nova configuração de ambiente. Os resultados foram comparados com os obtidos pelo algoritmo D* Lite (KOENIG; LIKHACHEV, 2002).

Primeiro, foi feito o ajuste do parâmetro delta para o procedimento de suavização, apresentado na Equação 3.12. A Tabela 31 mostra os resultados, na qual a primeira linha da tabela apresenta os resultados obtidos antes da modificação do ambiente. As linhas seguintes mostram os resultados após a modificação do ambiente, variando o parâmetro delta. A coluna *Melhor caminho* é a média das melhores soluções encontradas pelo algoritmo; *Média da melhor distância* é a média da distância percorrida da formiga que encontrou a melhor solução de todas as execuções durante toda a execução do ACO; *Melhor iteração* é a iteração na qual a melhor solução foi encontrada; *Total de iterações* é o número total de iterações gasto pelo algoritmo; *Distância por iteração* refere-se à média da distância percorrida por cada formiga em uma iteração.

A Tabela 31 mostra a análise dos resultados da variação do valor de delta no algoritmo para avaliar a adaptabilidade às mudanças no ambiente. Nesse experimento, a modificação dinâmica consiste em remover uma aresta do meio da melhor solução encontrada no ambiente estático. Como o procedimento de suavização é aplicado após o algoritmo terminar sua execução no ambiente estático, o valor delta não tem influência nos resultados do ambiente estático. A próxima melhor solução obtida por diferentes valores de delta é aproximadamente a mesma. No entanto, pode-se observar que quanto menor for o valor delta, menor a distância percorrida e a iteração para encontrar a melhor solução. A distância percorrida por iteração também é menor para $\delta = 0.1$. Isso sugere que manter uma pequena quantidade de feromônio a partir da solução anterior é útil, no entanto, esse pequeno valor permitiu ao algoritmo explorar outras possibilidades e escapar da solução anterior. Portanto, 0.1 é fixado para o valor delta nas próximas avaliações.

Tabela 31 – Ambiente dinâmico - Suavização.

		Melhor Caminho	Média da melhor distância	Melhor iteração	Total de iterações	Distância por iteração
Estático		1111.47 ± 42.84	194960.87 ± 100896.87	22.22 ± 25.57	123.22 ± 25.57	4307.19 ± 174.98
Delta	0.1	1151.36 ± 15.68	94998.22 ± 60163.73	12.11 ± 13.32	113.11 ± 13.22	3931.63 ± 153.77
	0.3	1151.71 ± 12.33	183621.44 ± 86213.99	15.35 ± 14.74	116.35 ± 14.74	4777.99 ± 222.09
	0.5	1151.24 ± 11.98	281552.84 ± 105949.99	21.12 ± 16.46	122.21 ± 16.45	5439.03 ± 287.03
	1.0	1161.05 ± 24.12	475914.37 ± 133456.30	29.03 ± 13.40	130.03 ± 13.40	6625.58 ± 324.98

Fonte: Dados da pesquisa.

A Tabela 32 apresenta os resultados para o ambiente dinâmico, quando as arestas são removidas do início, meio e fim da solução anterior. A Figura 24 apresenta a melhor solução e a aresta removida do início do caminho. A nova melhor solução também é apresentada. Quando essa aresta é removida, observa-se na Tabela 32 que o algoritmo MMAS-DPP encontra a nova solução com facilidade. Em todas as execuções, a melhor solução foi alcançada com poucas iterações, como apresentado na coluna *Melhor iteração*.

Tabela 32 – Ambiente dinâmico - Remoção de arestas.

		Melhor caminho médio	Média da melhor distância	Melhor iteração	Total de iterações	Distância por iteração
Estático		1111.47 ± 42.84	194960.87 ± 100896.87	22.22 ± 25.57	123.22 ± 25.57	4307.19 ± 174.98
Aresta	Início	1208.93	12960.33 ± 21449.54	0.66 ± 3.31	101.66 ± 3.31	3843.20 ± 82.85
	Meio	1151.36 ± 15.68	94998.22 ± 60163.73	12.11 ± 13.32	113.11 ± 13.22	3931.63 ± 153.77
	Fim	1217.37 ± 17.41	44368.81 ± 30722.85	3.18 ± 3.35	104.18 ± 3.35	4033.26 ± 113.25

Fonte: Dados da pesquisa.

Figura 24 – Aresta removida do início da melhor solução é destacada com a linha vermelha no mapa. A linha azul apresenta as novas arestas da solução.

Fonte: Elaborada pelo autor.

O algoritmo percorreu uma distância maior para encontrar a nova solução quando a aresta foi removida do meio, como observa-se na segunda linha da Tabela 32. Mesmo assim, o algoritmo pode encontrar a nova solução ótima nas primeiras iterações, de acordo com as colunas *Melhor iteração* e *Total de iterações*. Esse resultado mostra a adaptabilidade do algoritmo MMAS-DPP de lidar com ambientes dinâmicos, devido à sua capacidade de explorar o espaço de busca. A Figura 25 apresenta a aresta removida e a nova solução ótima.

Figura 25 – Aresta removida do meio da melhor solução é destacada em vermelho no mapa. A linha azul apresenta as novas arestas da solução.

Fonte: Elaborada pelo autor.

Para a aresta removida do final do caminho, o algoritmo encontrou a nova solução ótima em poucas iterações. A Figura 26 apresenta a aresta removida e a nova solução ótima.

Figura 26 – Aresta removida do final da melhor solução é destacada em vermelho. A linha azul apresenta as novas arestas da solução.

Fonte: Elaborada pelo autor.

A Tabela 33 apresenta a comparação entre os algoritmos D* Lite e MMAS-DPP, em relação à distância percorrida para encontrar a melhor solução. A primeira linha apresenta os resultados antes da modificação no ambiente. A coluna *Distância* é a distância percorrida pelo algoritmo D* Lite para encontrar a solução ótima. A Coluna *Melhor distância* é a melhor distância percorrida pelo algoritmo MMAS-DPP para encontrar a melhor solução. A coluna *Média da melhor distância* é a média da melhor distância percorrida de todas as execuções do algoritmo MMAS-DPP; e a coluna *Distância por iteração* é a média da distância percorrida por cada formiga em uma iteração.

A melhor distância percorrida pelo algoritmo MMAS-DPP é muito melhor do que a percorrida pelo algoritmo D* Lite, apesar de que em comparação com a média desse valor, o resultado obtido pelo algoritmo D* Lite foi menor. A distância média percorrida por iteração significa que cada formiga percorreu uma distância menor do que o D* Lite para deixar a posição inicial e atingir o objetivo. Isso mostra que o trabalho colaborativo gera melhores resultados em comparação ao trabalho individual.

Em experimentos no ambiente dinâmico, foram removidas as arestas do início, meio e fim da melhor solução obtida previamente. A Tabela 33 apresenta os resultados desses casos, considerando os algoritmos D* Lite e MMAS-DPP. A melhor distância percorrida pelo MMAS-

DPP é muito menor do que a percorrida pelo D* Lite na busca pela próxima melhor solução. Nos três casos, os valores obtidos pelo MMAS-DPP são muito próximos.

Em relação à média da melhor distância percorrida, se considerarmos o desvio padrão, o pior resultado obtido pelo MMAS-DPP está próximo à distância obtida pelo D* Lite. No segundo caso, que é o mais complexo, a média da melhor distância percorrida obtida pelo MMAS-DPP é pior do que a obtida pelo algoritmo D* Lite. As distâncias percorridas pelas formigas por iteração são similares e bem inferiores à distância gasta pelo D* Lite. As melhores distâncias alcançadas pelo MMAS-DPP também são inferiores.

Tabela 33 – Comparação entre os resultados obtidos pelos algoritmos D* Lite e MMAS-DPP.

		D* Lite	MMAS-DPP		
		Distância	Melhor distância	Média da melhor distância	Distância por iteração
Estático		36135.20	7702.31	163580.26 ± 79431.09	4322.43 ± 159.77
Dinâmico	Início	34217.4	1317.85	12960.33 ± 21449.54	3843.20 ± 82.85
	Meio	28519.2	1181.29	94998.22 ± 60163.73	3931.63 ± 153.77
	Fim	70289.8	1210.67	44368.81 ± 30722.85	4033.26 ± 113.25

Fonte: Dados da pesquisa.

O algoritmo MMAS-DPP também foi avaliado em um ambiente dinâmico, considerando que a posição do objetivo foi alterada. Os resultados do ambiente antes e depois da mudança são apresentados na Tabela 34. A Figura 27 apresenta os pontos de destino, em que G_1 é o objetivo inicial, G_2, G_3, G_4, G_5 e G_6 são as novas posições de destino. Observa-se na Coluna *Melhor caminho* que a nova melhor solução é encontrada em todas as execuções pelo algoritmo MMAS-DPP todas as vezes em que o objetivo final é alterado. A melhor distância percorrida é menor para encontrar a nova solução. A Coluna *Melhor iteração* mostra que a nova solução é encontrada nas primeiras iterações do algoritmo. Inclusive, quando o objetivo muda para G_3 e G_4 , a solução é encontrada na primeira iteração. A distância percorrida por iteração é muito próxima ao valor obtido no ambiente estático, de modo que para G_3 e G_4 ela é menor. Os autores do algoritmo D* Lite (KOENIG; LIKHACHEV, 2002) não mencionam a aplicação do algoritmo para esse tipo de ambiente dinâmico.

Tabela 34 – Ambiente dinâmico - Mudança de objetivo.

Goal	Melhor caminho	Melhor distância percorrida	Melhor iteração	Total de iterações	Distância por iteração
G1	1111.47 ± 42.84	194960.87 ± 100896.87	22.22 ± 25.57	123.22 ± 25.57	4307.19 ± 174.98
G2	1108.36	71768.60 ± 69149.38	2.03 ± 4.29	103.03 ± 4.29	4500.83 ± 119.80
G3	686.30	26483.03 ± 27585.10	0	101	2862.75 ± 78.99
G4	434.20	32515.89 ± 30792.79	0	101	2183.90 ± 82.31
G5	1099.68	74883.89 ± 64981.25	2.49 ± 5.64	103.49 ± 5.64	4249.93 ± 109.85
G6	1067.91	75089.63 ± 57757.71	2.12 ± 3.74	103.12 ± 3.74	4332.75 ± 111.82

Fonte: Dados da pesquisa.

Figura 27 – O mapa com o novo objetivo determinado.

Fonte: Elaborada pelo autor.

Embora o algoritmo D* Lite encontre a solução ótima quando há mudança no ambiente, ele percorre uma maior distância para atingir o objetivo. O algoritmo MMAS-DPP nem sempre encontra a solução ótima, mas pode encontrar uma solução aproximada satisfatória (veja Coluna *Média do melhor caminho* da Tabela 32) e pode alcançar o objetivo rapidamente, considerando a distância percorrida. Essa característica pode ser muito útil em aplicações cujo o objetivo precise ser alcançado rapidamente, como em tarefas de busca e resgate.

Além disso, o algoritmo MMAS-DDP trabalha em uma representação aproximada do ambiente, considerando mapas pouco definidos e localização imprecisa do robô e do objetivo, o que é suficiente para encontrar um caminho navegável. Embora o D* Lite tenha sido aplicado na mesma representação do ambiente, ele depende completamente desse mapa para encontrar o caminho. Durante a navegação, o robô deveria seguir precisamente um conjunto de pontos e, se algo está diferente na representação do mapa, o algoritmo pode falhar.

A navegação do melhor caminho obtido no mapa ilustrado na Figura 23, entre S_1 e G_1 , foi executada no simulador robô V-REP, um simulador robótico realístico (ROHMER S. P. N. SINGH, 2013), utilizando um robô Pioneer P3DX com um sensor 2D Laser Scanner. Um exemplo de um robô no simulador V-REP com seus sensores pode ser visto na Figura 28. O caminho fornecido pela abordagem proposta pode ser representado por uma sequência de ações.

Na navegação desse caminho, o robô começa a caminhar em um corredor. Quando detecta uma mudança de estado, a próxima ação é ativada. O reconhecedor de estado proposto por Santos V. C.; Toledo e Osório (2014) foi aplicado para identificar quando o estado do robô muda. Ele reconhece o estado com base nos sensores do robô.

Figura 28 – Um exemplo de um robô no simulador de robô V-REP. Os pontos vermelhos indicam onde o sensor laser atinge os obstáculos.

Fonte: Elaborada pelo autor.

O vídeo do robô executando o melhor caminho no mapa apresentado na Figura 23, de S_1 a G_1 pode ser visualizado em <<https://youtu.be/9XO2swnY9D4>>. O robô não possui informações sobre o meio ambiente, ele apenas conhece a sequência de ações que deve executar. As sequências de ações são: *virar à leste, virar à oeste, virar à leste, seguir para o norte, seguir para o norte, seguir para o norte, virar à oeste, seguir para o norte, virar à leste, virar à oeste, seguir para o norte, seguir para o norte, virar à leste, virar à oeste, seguir para o norte, seguir para o norte, virar à leste, virar à oeste, virar à leste, seguir para o norte, virar à oeste, virar à leste, virar à oeste, virar à leste, seguir para o norte e virar à oeste.*

4.2.3 Considerações Finais

Neste capítulo foram apresentados os algoritmos propostos para o problema de planejamento de trajetórias de robôs móveis autônomos. É importante destacar que no AG I o robô não precisa mais conhecer o caminho a ser seguido, como proposto por Sales, Osório e Wolf (2011), porque apenas é necessário ativar o próximo comportamento reativo gerado pelo AG quando uma transição de estado é detectada. Assim, as melhorias ao trabalho de Sales, Osório e Wolf (2011) é uma das principais contribuições desta proposta. O sistema proposto mostrou ser uma abordagem promissora para planejamento autônomo de trajetórias e controle de movimento de robôs móveis.

No AG II a representação de mapa foi estendida para permitir o uso de novas estruturas de mapa e executar novas ações. O robô foi capaz de seguir o caminho e alcançar seu objetivo sem conhecimento prévio do ambiente e sem uso de algoritmos de localização precisa. O AG foi capaz de encontrar caminhos satisfatórios para o mapa considerado com diferentes pontos de início e destino. Nesse estágio do projeto, um reconhecedor de estado foi proposto, baseado nas leituras do sensor. Com esse reconhecedor, o robô é capaz de identificar em que estado no ambiente ele se encontra e qual o momento que ele deve ativar uma ação. Além disso, o robô pode identificar quais são os caminhos possíveis que ele pode navegar. O reconhecedor de estado foi capaz de identificar a mudança de estado e ativar as ações planejadas pelo AG.

O AG III apresenta uma função de aptidão que é capaz de avaliar as ações executadas na geração atual e as ações executadas nas gerações anteriores (função de custo realística considerando todos os movimentos do robô). Um indivíduo com sequência de ações que levam o robô a alcançar o objetivo não é substituída e pode tomar parte do processo de reprodução do AG. Entretanto, a evolução continua apenas para sequências de ações relacionadas com robôs que não alcançaram o objetivo. O AG III proposto foi comparado com o algoritmo A^* e um algoritmo proposto, chamado C^* , que modifica o A^* para considerar e avaliar a distância quando retrocessos são executados. A comparação considera a distância percorrida pelos robôs em cada algoritmo. Três mapas foram avaliados e o C^* proposto apresentou desempenho melhor em situação onde o A^* teve que executar vários retrocessos. O AG tem desempenho médio melhor que o A^* e o C^* e as menores distâncias percorridas das soluções retornadas pelo AG depois de 100 execuções são sempre melhores que as soluções retornadas pelos algoritmos A^* e C^* . O método também é capaz de retornar mais que 80% de soluções melhores que o A^* e o C^* na maioria das situações.

Nos algoritmos genéticos, pode-se perceber que houve um amadurecimento tanto do problema quanto dos algoritmos propostos até chegar no AG III, que planeja a trajetória por partes até alcançar o objetivo. Os resultados foram satisfatórios em relação à qualidade das soluções encontradas. Além disso, a análise da distância percorrida para obter a solução final mostra a adequabilidade dos algoritmos propostos em situações reais.

O uso de ACO para este problema foi bastante eficiente. O ACO possui o diferencial de ter a quantidade de feromônio no ambiente influenciando a decisão para o próximo nó a ser explorado, que reflete a qualidade das soluções encontradas. Graças à influência do feromônio, com esse algoritmo os robôs puderam alcançar o objetivo pela primeira vez percorrendo distância menor que os algoritmos genéticos propostos.

NAVEGAÇÃO TOPOLÓGICA

No capítulo anterior foram discutidos diversos algoritmos que permitem a exploração e descoberta de caminhos de uma posição inicial até uma posição final, através de uma abordagem de planejamento exploratório de trajetórias. No presente capítulo será dada ênfase ao processo de navegação do robô no ambiente, ou seja, serão apresentadas abordagens que permitem que o robô realize a navegação de um ponto a outro dentro de um mapa topológico (grafo composto por nós e suas interconexões que são usados de referência para a navegação). A navegação topológica é baseada na identificação (reconhecimento) dos nós e das transições de um nó para outro do grafo que representa a topologia do ambiente, onde é através desta identificação dos nós e das transições que o robô se localiza, define e navega em suas trajetórias.

Na navegação topológica, conforme a abordagem adotada nessa tese, o robô navega em um corredor e quando identifica uma mudança de estado, executa uma ação. Para identificar o estado atual do robô, algoritmos de reconhecimento de padrões são executados durante toda a navegação. Neste capítulo, os algoritmos de reconhecimento de padrões adotados nesta pesquisa são apresentados. Na Seção 5.1 é apresentada a navegação com Redes Neurais Artificiais, na Seção 5.2 é apresentada a navegação com algoritmo baseado em Distância Normalizada de Compressão e na Seção 5.3 é apresentada a navegação com algoritmo de Reconhecimento de Estados baseado em Ângulos de Referência.

5.1 Navegação com Redes Neurais Artificiais

A motivação inicial deste trabalho foi o sistema de navegação topológica de robôs móveis (*Neural Finite-State Automata - Neuro-FSM*) proposto por Sales, Osório e Wolf (2011), Sales (2012), Sales *et al.* (2012), Sales *et al.* (2014). O Neuro-FSM foi desenvolvido para fornecer o controle e sistema de navegação para robôs móveis baseado em mapas topológicos previamente conhecidos e caminhos previamente planejados e definidos, que são especificados com base em mapas topológicos.

Na abordagem Neuro-FSM, o mapa, o caminho e as ações do robô foram definidos previamente. O sistema FSM - *Finite State Machine* é um sequenciador, ou seja, uma máquina de estados que tem "embutido" na estrutura deste autômato o plano da trajetória, definido pela sequência de estados/ações. O sequenciador do FSM usa o estado atual para associar uma ação específica ao contexto atual, e, quando o estado atual muda, uma nova ação é selecionada para ser executada. Por exemplo, se o robô está seguindo um caminho em um corredor, o estado atual é definido como "corredor" e a ação é mover-se no corredor. Quando o robô chega no fim do corredor, ou encontra outra situação nova representando um contexto diferente (por exemplo, curvas, cruzamentos), o robô deve trocar a ação para outro comportamento que o permita lidar com a nova situação. Às vezes, há mais que uma ação possível quando uma nova situação é encontrada. Por isso, o FSM foi codificado em sequências de ações a serem seguidas cada vez que é preciso mudar de um estado para outro, de modo a executar uma trajetória previamente definida que o leva ao seu destino. Quando o robô chega no fim do corredor com uma ação virar à esquerda, ele reconhece essa situação e troca para o comportamento virar à esquerda. A sequência de ações define o caminho a ser percorrido pelo robô, e permite que ele se localize em relação a trajetória descrita na forma de um caminho definido junto ao grafo que representa o mapa topológico. Em resumo, o FSM define a sequência de estados e ações que robô espera encontrar e busca executar, mas o que causa a decisão de quando mudar de um estado/ação atual para um novo estado/ação são as entradas que o FSM recebe do módulo neural. As entradas são representadas pelas leituras do ambiente obtidas pelos sensores do robô.

Uma Rede Neural Artificial (RNA) foi adotada para detectar os estados (estado atual) e as mudanças de estado (mudança de contexto), baseado nos sensores do robô. A RNA foi usada para reconhecer o estado atual e as mudanças de estado baseada nas entradas sensoriais do robô. O reconhecimento de padrões e a análise das entradas sensoriais são executados por uma RNA que percebe o ambiente e detecta o estado atual. Por exemplo, a RNA é capaz de classificar os padrões de entrada (por exemplo, dados de um sensor laser) que definem um corredor e, quando o robô chega em uma interseção, a RNA também será treinada para identificar esse novo padrão e notificar o FSM que o robô alcançou uma interseção. Por outro lado, quando o FSM é notificado sobre uma mudança de estado, define a ação a ser selecionada e executada nessa nova configuração, por exemplo, curva à esquerda em uma interseção. A Figura 29 apresenta uma visão geral da arquitetura Neuro-FSM.

A abordagem Neuro-FSM é interessante porque permite que um robô móvel navegue baseado apenas com base em um mapa topológico muito simples e um conjunto de ações fornecidos previamente. Embora essa abordagem mostre-se confiável, o usuário precisa fornecer todo o conhecimento sobre o problema: mapa, estados possíveis, ações e caminho. A Figura 30 mostra um exemplo de alguns estados, percepções, mapa topológico e caminho (sequência de nós no grafo). Uma das tarefas mais complexas é treinar a RNA para reconhecer estados diferentes por se tratar de um algoritmo de aprendizado supervisionado. Foi necessário treinar a RNA para reconhecer os estados diferentes (por exemplo, corredor, curva à esquerda, curva à direita,

Figura 29 – Visão geral da arquitetura do sistema Neuro-FSM.

Fonte: Adaptada de Soares *et al.* (2016).

interseção), apresentando um conjunto de exemplos rotulados. Essa tarefa pode ser difícil de ser executada.

Figura 30 – Exemplos de percepção e estado do robô (a) até (e) e o grafo representando um mapa topológico com um caminho pré-definido.

Fonte: Adaptada de Soares *et al.* (2016).

O mapa topológico, plano do caminho (FSM) e conjunto de ações (comportamentos reativos) podem ser definido pelo usuário e normalmente, a tarefa mais difícil é encontrar o caminho da origem para o destino, quando estamos navegando em um mapa de ambiente mais complexo. O conjunto de ações é um conjunto de comportamentos reativos simples (seguir adiante, seguir a parede, virar n graus para esquerda/direita, virar para a esquerda/direita até que haja um espaço livre na frente do robô). Concluindo, as duas tarefas mais complexas são: aprender e reconhecer as configurações diferentes do ambiente e descobrir o caminho da origem para o destino (planejamento de trajetórias).

O algoritmo AG I foi proposto com o objetivo de encontrar a sequência de ações para

alcançar o objetivo de forma autônoma, no contexto desta abordagem adotada pelo sistema Neuro-FSM. Para a navegação da melhor solução encontrada pelo AG, uma rede neural foi treinada para reconhecer os estados possíveis para o mapa adotado. Para treinar a RNA, foi feita uma navegação prévia do robô no ambiente para coletar os dados de cada estado, que são rotulados manualmente. Durante a navegação, a RNA reconhece o estado atual do robô e uma ação correspondente é ativada, de acordo com a mudança detectada. O algoritmo AG I com a navegação baseada no sistema Neuro-FSM é descrito no artigo (SANTOS *et al.*, 2014) e o vídeo pode ser visualizado em <<https://youtu.be/rWA8gdt5kj0>>.

5.2 Navegação com Distância Normalizada de Compressão

Um desafio da navegação topológica, conforme indicado na seção anterior, é como descobrir a mudança de um estado e reconhecer os diferentes estados. Em função disso, foram investigadas técnicas que permitissem analisar dados, no caso os dados das leituras dos sensores, e tentar de forma automática identificar padrões similares. Essas técnicas de análise de dados devem identificar dados similares, quando ocorrem mudança nos padrões de dados, e também, buscar agrupar os padrões similares em grupos (permitindo um reconhecimento dos mesmos).

Distância Normalizada de Compressão (*Normalized Compression Distance* (NCD)), da teoria da informação (CILIBRASI; VITANYI, 2005), é uma métrica que pode encontrar relacionamentos entre dados, determinando similaridade entre variáveis por medir o tamanho de seus dados comprimidos. Essa abordagem não requer qualquer conhecimento específico do domínio da aplicação. De acordo com Cilibrasi e Vitanyi (2005), NCD é uma métrica universal e robusta que tem sido aplicada com sucesso em áreas como genética, literatura, música e astronomia.

NCD é baseada na Distância de Informação Normalizada (*Normalized Information Distance* (NID)) (TERWIJN; TORENVLIET; VITANYI, 2011), que considera a similaridade entre amostras (variáveis, arquivos, etc), de acordo com as características dominantes compartilhadas por eles. O NID usa o conceito de complexidade de Kolmogorov (LI; VITNYI, 1997) para estimar distâncias, entretanto, o cálculo correspondente é computacionalmente intratável.

O NCD realiza uma aproximação do NID baseado nos resultados de um algoritmo de compressão. Na prática, a distância entre duas amostras (armazenadas em arquivos) calculada pelo NCD é um número positivo que varia entre $[0; 1 + \epsilon]$. Esse número representa o quanto dois arquivos são diferentes, onde ϵ é o limite superior para o erro do compressor usado. O valor de distância entre dois arquivos x e y é dado pela Equação 5.1.

$$NCD(x, y) = \frac{C(xy) - \min \{C(x), C(y)\}}{\max \{C(x), C(y)\}} \quad (5.1)$$

xy é o arquivo concatenado de x e y , $C(xy)$ é o tamanho do arquivo xy depois da compressão e $C(x)$ e $C(y)$ são os tamanhos dos arquivos x e y depois da compressão, respectivamente.

O nível das compressões afeta os valores de distância determinados pelo NCD. Assim, a escolha de um compressor pode ser baseada na taxa de compressão dos arquivos envolvidos, considerando vários compressores encontrados na literatura (ITO; ZEUGMANN; ZHU, 2010). Os compressores usados com NCD mais comuns são *bzip*, *gzip*, *lzm* and *ppmz*. O compressor *bzip* apresentou as melhores taxas de compressão para os dados usados nesse trabalho.

A abordagem inicial adotada no Neuro-FSM requer o treinamento prévio da RNA para que ela seja capaz de reconhecer cada estado diferente (contexto atual), representado pelos padrões dos dados do sensor. A RNA adotada na abordagem Neuro-FSM foi treinada baseada em um algoritmo de aprendizado supervisionado, que significa que o usuário deve preparar previamente um conjunto de exemplos de treinamento (padrões rotulados) para treinar a rede neural. Neste trabalho, foi proposto o Método de Detecção de Novidade com NCD (DN-NCD), publicado em (SOARES *et al.*, 2015; SOARES *et al.*, 2016), que busca evitar a necessidade desse passo prévio de treinamento.

A maior parte do caminho é normalmente composta por corredores, onde o robô pode seguir reto. Quando o estado muda, ou seja, quando o estado não é corredor, esse evento deve ser detectado e uma nova ação é requerida. Assim, o método ND-NCD (SOARES *et al.*, 2015; SOARES *et al.*, 2016) é proposto para reconhecer o momento que o robô saiu do corredor.

O método é dividido em três procedimentos distintos:

1. Coleta de dados. Nessa etapa, o ambiente é percorrido durante uma travessia para coletar dados de corredor que irão compor a base de conhecimento. A varredura é feita pelo robô em um controle automático (por exemplo, comportamento de seguir parede). Não é necessário visitar todos os corredores, apenas uma amostra suficiente que represente os principais aspectos de um corredor.
2. Cálculo da sensibilidade de ativação. O cálculo é executado usando NCD para gerar um vetor U_k de distâncias para cada amostra do vetor V_k da base de conhecimento. A maior mediana $m_k = \text{median}_k\{U_k(i)\}$ define o valor de sensibilidade da base, chamada Sensibilidade de Ativação (SA). Isso significa que a amostra com valor de mediana menor que SA pode ser considerada um corredor, como resumido na Figura 31.
3. Execução. O NCD calcula as distâncias entre cada amostra V_k obtida por um sensor laser acoplado ao robô e as amostras da base de conhecimento. Então, cada amostra capturada resulta em um vetor de distâncias U_K com um valor de mediana correspondente. Se esse valor for maior do que SA, a amostra não é reconhecida como corredor (novo estado), como descrito no diagrama da Figura 32.

Figura 31 – Diagrama do cálculo da Sensibilidade de Ativação.

Fonte: Adaptada de Soares *et al.* (2016).

Para evitar erros na rotulação, um filtro foi aplicado para uma distribuição de medianas para o cálculo de SA. O filtro usado no ND-NCD é a média aritmética das últimas 10 saídas de 50 medianas. Consequentemente, a classificação de um sinal externo pode ocorrer apenas depois do cálculo de uma janela de amostra de tamanho 10. A Figura 33 mostra a distribuição das médias com e sem filtro. Por fim, é importante notar que o ND-NCD é desabilitado enquanto o robô está executando uma ação.

Esse trabalho obteve resultados promissores, em que foi preciso apenas reconhecer o estado corredor para a realização da navegação do robô. Os resultados desse método foram publicados em (SOARES *et al.*, 2015; SOARES *et al.*, 2016). A navegação com o uso do ND-NCD no simulador Player/Stage e em ambientes reais podem ser visualizados em <<https://goo.gl/5ZfqPU>>, <<https://goo.gl/npvhfU>>, <<https://goo.gl/lveF8Q>>, <<https://goo.gl/HyKPpi>>.

5.3 Navegação com Reconhecimento de Estados baseado em Ângulos de Referência

Na nova estrutura de ambiente proposta no desenvolvimento do AG II, novas possibilidades de estado surgiram, com bifurcações em diferentes angulações (além das curvas somente em 90 graus), dificultando o treinamento de uma RNA. Com essa motivação, um algoritmo simples

Figura 32 – Processo de ativação das ações.

Fonte: Adaptada de Soares *et al.* (2016).

de reconhecimento de estado foi proposto, que usa apenas a leitura do sensor do robô. O sensor laser adotado neste trabalho possui uma varredura de 0 a 180 graus. Para cada ângulo, é possível obter a distância do robô para os obstáculos, sendo que a distância máxima é de 8 metros, quando o laser não intercepta um obstáculo. O algoritmo identifica quais os ângulos que não interceptam obstáculos, denominados ângulos livres, e aproxima o valor médio desses ângulos aos ângulos de referência.

Na estrutura de ambiente usado neste trabalho, cinco áreas livres são possíveis, de acordo com as leituras do sensor: leste (0°), nordeste (45°), norte (90°), noroeste (135°) and oeste (180°). Na Figura 34, é possível observar o laser e os ângulos usados como referência para identificar as áreas livres.

Para cada ângulo, o algoritmo verifica se ele é um ângulo livre. Se for, o algoritmo começa a calcular uma soma de cada valor de ângulo até encontrar um valor que indique que um obstáculo foi encontrado (distância menor que a distância máxima considerada, no caso oito metros). Para essa soma, o valor médio desses ângulos livres é calculado, considerando o número de ângulos lidos. Depois, a distância desse valor médio para cada ângulo de referência é encontrado. A nova área livre é classificada como a mais próxima ao ângulo de referência.

Por exemplo, na Figura 34, a partir do ângulo 0° até o ângulo 40°, identificado pelos círculos vermelhos, não há obstáculos. A média desse intervalo é 20, que é mais próximo de

Figura 33 – Filtros diferentes para o ND-NCD.

Fonte: Soares *et al.* (2016).

leste (0°). Então, a região leste é rotulada como livre e o robô pode executar a ação *virar à leste*. Nesse exemplo, o mesmo processo é feito para as regiões norte e oeste.

Figura 34 – Figura ilustrando o sensor laser e os ângulos de referência utilizados pelo reconhecedor de estados.

Fonte: Elaborada pelo autor.

Essa abordagem é bem simples em termos de implementação e custo computacional e não exige treinamento prévio dos possíveis estados para ser executada. Apenas é preciso conhecer a estrutura do ambiente para informar os ângulos de referência. Além disso, apresentou bom

desempenho para os experimentos realizados. A navegação da trajetória com esse método pode ser vista nas publicações (SANTOS V. C.; TOLEDO; OSÓRIO, 2014; SANTOS; TOLEDO; OSÓRIO, 2015; SANTOS *et al.*, 2016).

5.4 Considerações Finais

Neste capítulo foram apresentadas as abordagens utilizadas para a identificação do estado durante a navegação do robô. As três abordagens são eficientes na tarefa de identificar o estado e cada uma apresenta vantagens e desvantagens.

A abordagem baseada em Redes Neurais Artificiais é robusta, porém, por ser um método de classificação supervisionado, é necessária a intervenção de um especialista para rotular os estados (por exemplo, curva à direita, corredor, etc.) e realizar o treinamento prévio.

O método DN-NCD é mais flexível. Como os mapas avaliados nesta pesquisa são formados por corredores e bifurcações, a ideia desse método é identificar apenas se o robô está em um corredor ou não. Quando ele não está no corredor significa que houve uma mudança de estado, então é o momento de executar uma ação. Esse método precisa ser aprimorado para reconhecer qual é o estado que o robô se encontra quando não está no corredor para que possa ser executado em aplicações de tempo real.

A abordagem de Navegação com Reconhecimento de Estados baseado em Ângulos de Referência não exige treinamento prévio para identificar os estados. É preciso apenas identificar os ângulos de referência para fazer o reconhecimento. Essa abordagem mostrou-se bastante eficiente nos experimentos realizados.

CONCLUSÕES E TRABALHOS FUTUROS

Nessa pesquisa, uma abordagem híbrida para planejamento e controle de navegação de robôs móveis em ambiente *indoor* foi apresentada. A abordagem é dita híbrida porque há um compromisso não somente para a obtenção da trajetória (etapa de planejamento), mas também para permitir que essa trajetória possa ser cumprida de forma adequada por robôs em aplicações reais, considerando portanto a navegação do robô.

O ambiente considerado é um mapa do tipo labirinto, *indoor*, desconhecido pelos robôs, onde o caminho até o objetivo final é descoberto pelo robô, representado de forma topológica. Como os robôs não possuem conhecimento prévio do ambiente, este é considerado um problema de planejamento exploratório de trajetórias, no qual eles descobrem o ambiente à medida que planejam sua rota para um objetivo. Além disso, os métodos não dependem de localização precisa ou de representação precisa do ambiente.

Na etapa de planejamento, foram investigados os algoritmos genéticos e algoritmos de otimização por colônia de formigas no mesmo contexto, em ambientes estáticos e dinâmicos. Os algoritmos genéticos foram desenvolvidos até a obtenção do AG III, que planeja a trajetória por partes até alcançar o objetivo. Os resultados foram satisfatórios em relação à qualidade das soluções encontradas. Além disso, a análise da distância percorrida para obter a solução final mostra a adequabilidade dos algoritmos propostos em situações reais.

O uso de ACO para este problema foi bastante eficiente. O ACO possui o diferencial de ter a quantidade de feromônio no ambiente influenciando a decisão para o próximo nó a ser explorado, que reflete a qualidade das soluções encontradas. Graças à influência do feromônio, com esse algoritmo os robôs puderam alcançar o objetivo pela primeira vez percorrendo distância menor que os algoritmos genéticos propostos. Embora a presença de feromônio possa ser uma dificuldade em implementações simuladas ou reais, essa abordagem apresentou melhor desempenho nos experimentos realizados nesta pesquisa.

A comparação com os algoritmos A* e D* em relação à distância percorrida mostra que

os métodos propostos são mais adequados do que esses algoritmos em situações que se deseja alcançar o objetivo o mais rápido possível, com a execução de uma trajetória satisfatória, não necessariamente a ótima.

Na etapa de navegação da trajetória foram utilizados três métodos, baseado em Redes Neurais, baseado em Distância Normalizada de Compressão e um algoritmo de Reconhecimento de Estados baseado em Ângulos de Referência.

6.1 Contribuições

A principal contribuição dessa tese é a obtenção de um método de planejamento de trajetória comprometido com sua execução em ambientes com robôs móveis e na definição da trajetória como um conjunto de ações (comportamentos reativos) que levam o robô da posição inicial ao objetivo. Essa definição permite que o planejamento e a navegação da trajetória sejam realizados sem a necessidade de representação precisa do ambiente ou localização precisa do robô.

Além disso, outras contribuições e resultados relevantes foram fornecidos:

- Desenvolvimento de algoritmos genéticos para o planejamento exploratório de trajetórias em mapas do tipo labirinto, no qual os robôs não tem conhecimento prévio do ambiente;
- Desenvolvimento de algoritmos de otimização por colônia de formigas no mesmo contexto, em ambientes estáticos e dinâmicos;
- Desenvolvimento de algoritmos de reconhecimento de padrões para identificar o estado atual do robô durante a execução da navegação.

6.2 Limitações e Dificuldades

Apesar dos algoritmos genéticos e algoritmos de otimização por colônia de formigas apresentarem bom desempenho para as avaliações realizadas, esses métodos não garantem que a solução ótima seja sempre alcançada, não sendo adequados para situações onde encontrar a solução ótima seja indispensável junto a uma determinada tarefa.

O uso da representação de grafo para avaliação dos algoritmos foi adotado porque a avaliação no ambiente robótico seria muito lenta. Porém, a navegação das soluções obtidas foram realizadas em simuladores robóticos e em ambiente real, mostrando que a representação em grafo é equivalente à execução em ambientes reais.

6.3 Trabalhos Futuros

Os algoritmos AG III e MMAS-DPP serão aprimorados para gerar melhores resultados e serão avaliados em outros mapas para confirmar sua eficiência.

Para a navegação da trajetória, serão investigados algoritmos de aprendizado não-supervisionado, como o K-Médias (MACQUEEN, 1967) e o DBSCAN (ESTER *et al.*, 1996), que não exigem conhecimento sobre os possíveis estados do ambiente.

6.4 Lista de Publicações

- SANTOS, V. C.; OTERO, F. S. B.; JOHNSON, C. G.; OSORIO, F. S.; TOLEDO, C. F. M. **Exploratory Path Planning Using the Max-Min Ant System Algorithm**. In: IEEE World Congress on Computational Intelligence (IEEE WCCI), 2016, Vancouver. 2016 IEEE Congress on Evolutionary Computation, 2016. p. 4229-4235.
- Soares, Antônio; SANTOS, VALÉRIA; Toledo, Cláudio; Osório, Fernando; DELBEM, ALEXANDRE . **ND-NCD: Environmental Characteristics Recognition and Novelty Detection for Mobile Robots Control and Navigation**. Communications in Computer and Information Science. 1ed.: Springer International Publishing, 2016, v. , p. 192-209.
- DE CARVALHO SANTOS, VALERIA; TOLEDO, CLAUDIO FABIANO MOTTA ; OSORIO, FERNANDO SANTOS . **An exploratory path planning method based on genetic algorithm for autonomous mobile robots**. In: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, Sendai. 2015 IEEE Congress on Evolutionary Computation (CEC). p. 62-69.
- SOARES, ANTONIO ; SANTOS, VALERIA ; TOLEDO, CLAUDIO ; OSORIO, FERNANDO ; DELBEM, ALEXANDRE . **Novelty Detection Based on NCD for Navigation Control of Mobile Robots**. In: 2015 12th Latin American Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARSSBR), 2015, Uberlândia. 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR). v. 1. p. 246-252.
- DE CARVALHO SANTOS, VALÉRIA; SALES, DANIEL OLIVA ; TOLEDO, CLÁUDIO FABIANO MOTTA ; OSÓRIO, FERNANDO SANTOS . **A hybrid GA-ANN approach for autonomous robots topological navigation**. In: the 29th Annual ACM Symposium, 2014, Gyeongju. Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14. p. 148-153.
- SANTOS, VALERIA DE CARVALHO; TOLEDO, CLAUDIO FABIANO MOTTA ; OSORIO, FERNANDO SANTOS. **A Hybrid Approach for Path Planning and Execution**

for Autonomous Mobile Robots. In: 2014 Joint Conference on Robotics: SBRLARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014, Sao Carlos. 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol. p. 124-129.

6.5 Artigo Submetido

- SANTOS, V. de C.; OTERO, F. E. B.; JOHNSON, C. G.; OSÓRIO, F. S.; TOLEDO, C. F. M. An Ant Colony Optimization Approach for Exploratory Path Planning for Mobile Robots. In: Applied Soft Computing.

REFERÊNCIAS

ADORNO, B. V.; BORGES, G. A. Planejamento de rotas. In: ROMERO, R. A. F.; PRESTES, E.; OSÓRIO, F.; WOLF, D. (Ed.). **Robótica Móvel**. Rio de Janeiro: Editora LTC, 2014. cap. 6, p. 84–110. ISBN 978-85-216-2303-8. Citado nas páginas 27 e 40.

ATMAR, W. On the rules and nature of simulate evolutionary programming. **Proc. of the First Ann. Conf. On Evolutionary Programming**, p. 17–26, 1992. Citado nas páginas 31 e 45.

AUTERE, A. Hierarchical a* based path planning — a case study. **Knowledge-Based Systems**, v. 15, n. 1–2, p. 53 – 66, 2002. ISSN 0950-7051. Citado nas páginas 40 e 56.

CAGIGAS, D. Hierarchical d* algorithm with materialization of costs for robot path planning. **Robotics and Autonomous Systems**, v. 52, n. 2-3, p. 190–208, 2005. Citado nas páginas 40 e 56.

CHOSSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G.; BURGARD, W.; KAVRAKI, L. E.; THRUN, S. **Principles of Robot Motion: Theory, Algorithms, and Implementations**. Cambridge, MA: MIT Press, 2005. Citado na página 23.

CILIBRASI, R.; VITANYI, P. Clustering by compression. **Information Theory, IEEE Transactions on**, v. 51, n. 4, p. 1523–1545, April 2005. ISSN 0018-9448. Citado na página 96.

COLLETT, T. H.; MACDONALD, B. A.; GERKEY, B. P. Player 2.0: Toward a practical robot programming framework. In: **Proc. of the Australasian Conf. on Robotics and Automation (ACRA)**. Sydney, Australia: [s.n.], 2005. Citado nas páginas 49, 52, 76 e 80.

COSTA, A. H. R.; SELVATICI, A. H. P. **Localização usando representação do ambiente**. [S.l.]: in press, 2014. Citado nas páginas 28 e 29.

DAVOODI, M.; PANAHI, F.; MOHADES, A.; HASHEMI, S. N. Multi-objective path planning in discrete space. **Appl. Soft Comput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 13, n. 1, p. 709–720, jan 2013. ISSN 1568-4946. Citado nas páginas 34, 35 e 43.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast elitist multi-objective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, p. 182–197, 2000. Citado na página 34.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. Scituate, MA, USA: Bradford Company, 2004. ISBN 0262042193. Citado nas páginas 31, 34, 46, 57, 58, 59 e 61.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Proc. KDD**. [S.l.: s.n.], 1996. p. 226–231. Citado na página 105.

GARCIA, M. P.; MONTIEL, O.; CASTILLO, O.; SEPÚLVEDA, R.; MELIN, P. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. **Applied Soft Computing**, v. 9, n. 3, p. 1102 – 1110, 2009. ISSN 1568-4946. Citado nas páginas 36, 38 e 43.

HART, P.; NILSSON, N.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **Systems Science and Cybernetics, IEEE Transactions on**, v. 4, n. 2, p. 100–107, July 1968. ISSN 0536-1567. Citado nas páginas 40 e 78.

HAWA, M. Light-assisted a* path planning. **Engineering Applications of Artificial Intelligence**, v. 26, n. 2, p. 888 – 898, 2013. ISSN 0952-1976. Citado nas páginas 40 e 56.

HU, Y.; YANG, S. X. A knowledge based genetic algorithm for path planning of a mobile robot. In: **ICRA'04**. [S.l.: s.n.], 2004. p. 4350–4355. Citado nas páginas 31, 35 e 43.

IOANNIDIS, K.; SIRAKOULIS, G.; ANDREADIS, I. Cellular ants: A method to create collision free trajectories for a cooperative robot team. **Robotics and Autonomous Systems**, v. 59, n. 2, p. 113 – 127, 2011. ISSN 0921-8890. Citado nas páginas 37, 38 e 43.

ITO, K.; ZEUGMANN, T.; ZHU, Y. Clustering the Normalized Compression Distance for Influenza Virus Data. **Algorithms and Applications**, Springer, p. 130–146, 2010. Citado na página 97.

KALA, R. Multi-robot path planning using co-evolutionary genetic programming. **Expert Syst. Appl.**, v. 39, n. 3, p. 3817–3831, 2012. Citado nas páginas 33, 35, 43 e 63.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Neural Networks, 1995. Proceedings., IEEE International Conference on**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4. Citado na página 35.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **Int. J. Rob. Res.**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 5, n. 1, p. 90–98, apr 1986. ISSN 0278-3649. Citado na página 39.

KOENIG, S.; LIKHACHEV, M. Improved fast replanning for robot navigation in unknown terrain. In: **Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)**. [S.l.: s.n.], 2002. v. 1, p. 968–975 vol.1. Citado nas páginas 83 e 88.

KOENIG, S.; LIKHACHEV, M.; FURCY, D. Lifelong planning A*. **Artificial Intelligence**, v. 155, n. 1, p. 93 – 146, 2004. ISSN 0004-3702. Citado na página 40.

LI, M.; VITNYI, P. M. **An Introduction to Kolmogorov Complexity and Its Applications**. 2. ed. [S.l.]: Springer-Verlag New York, Inc., 1997. Citado na página 96.

LUMELSKY, V.; STEPANOV, A. Dynamic path planning for a mobile automaton with limited information on the environment. **IEEE Transactions on Automatic Control**, v. 31, n. 11, p. 1058–1063, Nov 1986. ISSN 0018-9286. Citado na página 39.

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: CAM, L. M. L.; NEYMAN, J. (Ed.). **Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability**. [S.l.]: University of California Press, 1967. v. 1, p. 281–297. Citado na página 105.

MICHEL, O. Webots: Professional mobile robot simulation. **Journal of Advanced Robotics Systems**, v. 1, n. 1, p. 39–42, 2004. Citado na página 37.

MITCHELL, M. Genetic algorithms: An overview. **Complexity**, v. 1, n. 1, p. 31–39, 1995. Citado nas páginas 31, 45 e 46.

MOHANTA, J. C.; PARHI, D. R.; PATEL, S. K. Path planning strategy for autonomous mobile robot navigation using petri-ga optimisation. **Comput. Electr. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 37, n. 6, p. 1058–1070, nov 2011. ISSN 0045-7906. Citado nas páginas 32, 35 e 43.

MONDADA, F.; BONANI, M.; RAEMY, X.; PUGH, J.; CIANCI, C.; KLAPTOCZ, A.; MAGNENAT, S.; ZUFFEREY, J.-C.; FLOREANO, D.; MARTINOLI, A. The e-puck, a robot designed for education in engineering. In: IPCB: INSTITUTO POLITÉCNICO DE CASTELO BRANCO. **Proceedings of the 9th conference on autonomous robot systems and competitions**. [S.l.], 2009. v. 1, n. LIS-CONF-2009-004, p. 59–65. Citado na página 37.

NELSON, A. L.; BARLOW, G. J.; DOITSIDIS, L. Fitness functions in evolutionary robotics: A survey and analysis. **Robot. Auton. Syst.**, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, v. 57, n. 4, p. 345–370, apr 2009. ISSN 0921-8890. Citado na página 23.

PETRI, C. A. **Kommunikation mit Automaten**. Tese (Doutorado) — Institut für instrumentelle Mathematik, Bonn, 1962. Citado na página 32.

QU, H.; XING, K.; ALEXANDER, T. An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. **Neurocomputing**, v. 120, n. 0, p. 509 – 517, 2013. ISSN 0925-2312. Citado nas páginas 33, 35 e 43.

REZENDE, S. O. **Sistemas Inteligentes: Fundamentos e Aplicação**. Barueri, SP: Manole, 2003. 1ª edição. Citado na página 45.

ROHMER S. P. N. SINGH, M. F. E. V-rep: a versatile and scalable robot simulation framework. In: **Proc. of The International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2013. Citado na página 89.

ROMERO, R.; PRESTES, E.; OSORIO, F.; WOLF, D. **Robótica Móvel**. [S.l.]: LTC, 2014. ISBN 9788521623038. Citado na página 25.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach (3rd Edition)**. 3. ed. [S.l.]: Pearson, 2009. Hardcover. ISBN 0136042597. Citado nas páginas 35, 40, 56 e 78.

SALES, D.; CORREA, D.; OSÓRIO, F. S.; WOLF, D. F. 3d vision-based autonomous navigation system using ann and kinect sensor. In: _____. **Engineering Applications of Neural Networks: 13th International Conference, EANN 2012, London, UK, September 20-23, 2012. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 305–314. ISBN 978-3-642-32909-8. Citado nas páginas 24, 25 e 93.

SALES, D. O. **NeuroFSM: Aprendizado de Autômatos Finitos através do uso de Redes Neurais Artificiais aplicadas à Robôs Móveis e Veículos Autônomos (Master Dissertation in Portuguese)**. São Carlos, Brasil: Universidade de Sao Paulo - ICMC - CCMC, 2012. Citado nas páginas 24, 25 e 93.

SALES, D. O.; CORREA, D.; FERNANDES, L.; WOLF, D. F.; OSÓRIO, F. S. Adaptive finite state machine based visual autonomous navigation system. **Engineering Applications of Artificial Intelligence**, Elsevier, p. 152–162, 2014. Citado nas páginas 24, 25 e 93.

SALES, D. O.; OSÓRIO, F. S.; WOLF, D. F. Topological autonomous navigation for mobile robots in indoor environments using ann and fsm. In: **I CBSEC: Conferência Brasileira em Sistemas Embarcados Críticos**. São Carlos, Brasil: [s.n.], 2011. Citado nas páginas 24, 25, 49, 66, 90 e 93.

SANTOS, V. C.; SALES, D. O.; TOLEDO, C. F. M.; OSÓRIO, F. S. A hybrid ga-ann approach for autonomous robots topological navigation. In: **Proceedings of the 29th Symposium On Applied Computing**. [S.l.]: ACM, 2014. p. 148–153. Citado nas páginas 46, 49, 63 e 96.

SANTOS V. C.; TOLEDO, C. F. M.; OSÓRIO, F. S. A hybrid approach for path planning and execution for autonomous mobile robots. In: **Proceedings of 2nd Brazilian Symposium on Robotics and 11th Latin American Robotics Symposium**. [S.l.]: ACM, 2014. p. 124–129. Citado nas páginas 49, 52, 63, 76, 80, 90 e 101.

SANTOS, V. de C.; OSÓRIO, F. S.; TOLEDO, C. F. M.; OTERO, F. E. B.; JOHNSON, C. G. Exploratory path planning using the max-min ant system algorithm. In: **IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016**. [S.l.: s.n.], 2016. p. 4229–4235. Citado nas páginas 57, 59 e 101.

SANTOS, V. de C.; TOLEDO, C. F. M.; OSORIO, F. S. An exploratory path planning method based on genetic algorithm for autonomous mobile robots. In: **Evolutionary Computation (CEC), 2015 IEEE Congress on**. [S.l.: s.n.], 2015. p. 62–69. Citado nas páginas 53, 56, 63 e 101.

SEDIGHI, K.; ASHENAYI, K.; MANIKAS, T.; WAINWRIGHT, R.; TAI, H.-M. Autonomous local path planning for a mobile robot using a genetic algorithm. In: **Evolutionary Computation, 2004. CEC2004. Congress on**. [S.l.: s.n.], 2004. v. 2, p. 1338–1345 Vol.2. Citado nas páginas 31, 35 e 43.

SIEGWART, R.; NOURBAKSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X. Citado na página 28.

SOARES, A.; SANTOS, V.; TOLEDO, C.; OSÓRIO, F.; DELBEM, A. Novelty detection based on ncd for navigation control of mobile robots. In: **2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)**. [S.l.: s.n.], 2015. p. 246–252. Citado nas páginas 97 e 98.

SOARES, A.; SANTOS, V.; TOLEDO, C.; OSÓRIO, F.; DELBEM, A. Nd-ncd: Environmental characteristics recognition and novelty detection for mobile robots control and navigation. In: _____. **Robotics: 12th Latin American Robotics Symposium and Third Brazilian Symposium on Robotics, LARS 2015/SBR 2015, Uberlândia, Brazil, October 28 - November 1, 2015, Revised Selected Papers**. Cham: Springer International Publishing, 2016. p. 192–209. ISBN 978-3-319-47247-8. Citado nas páginas 95, 97, 98, 99 e 100.

STÜTZLE, T.; HOOS, H. H. Max-min ant system. **Future Gener. Comput. Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 16, n. 9, p. 889–914, jun 2000. ISSN 0167-739X. Citado na página 59.

TERWIJN, S. A.; TORENVLIET, L.; VITÁNYI, P. M. Nonapproximability of the normalized information distance. **Journal of Computer and System Sciences**, v. 77, n. 4, p. 738 – 742, 2011. ISSN 0022-0000. {JCSS} {IEEE} {AINA} 2009. Citado na página 96.

THRUN, S. Robotic mapping: A survey. In: **Exploring Artificial Intelligence in the New Millenium**. [S.l.]: Morgan Kaufmann, 2002. Citado na página 28.

TUNCER, A.; YILDIRIM, M. Dynamic path planning of mobile robots with improved genetic algorithm. **Comput. Electr. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 6, p. 1564–1572, nov 2012. ISSN 0045-7906. Citado nas páginas 32, 35 e 43.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: **In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation**. [S.l.: s.n.], 1997. p. 146–151. Citado na página 38.

_____. Frontier-based exploration using multiple robots. In: **Proceedings of the Second International Conference on Autonomous Agents**. New York, NY, USA: ACM, 1998. (AGENTS '98), p. 47–53. ISBN 0-89791-983-1. Citado na página 38.

YAO, Y.; NI, Q.; LV, Q.; HUANG, K. A novel heterogeneous feature ant colony optimization and its application on robot path planning. In: **Evolutionary Computation (CEC), 2015 IEEE Congress on**. [S.l.: s.n.], 2015. p. 522–528. Citado nas páginas 37, 38 e 43.

YUN, S. C.; GANAPATHY, V.; CHONG, L. O. Improved genetic algorithms based optimum path planning for mobile robot. In: **Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on**. [S.l.: s.n.], 2010. p. 1565–1570. Citado nas páginas 31, 35 e 43.

ZHU, Q.; HU, J.; CAI, W.; HENSCHEN, L. A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm. **Applied Soft Computing**, v. 11, n. 8, p. 4667 – 4676, 2011. ISSN 1568-4946. Citado nas páginas 36, 37, 38 e 43.

ZHU, Q.; HU, J.; HENSCHEN, L. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. **Applied Soft Computing**, v. 13, n. 1, p. 539 – 549, 2013. ISSN 1568-4946. Citado nas páginas 37, 38 e 43.

ZLOCHIN, M.; BIRATTARI, M.; MEULEAU, N.; DORIGO, M. Model-based search for combinatorial optimization: A critical survey. **Annals of Operations Research**, v. 131, n. 1, p. 373–395, 2004. ISSN 1572-9338. Citado nas páginas 36, 45 e 46.