UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ALLAN COSTA GOMES

IMPLEMENTAÇÃO DE ALGORITMO BASEADO EM OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADO AO PLANEJAMENTO DE CAMINHOS PARA ROBÔ MÓVEL

FORTALEZA

2018

ALLAN COSTA GOMES

IMPLEMENTAÇÃO DE ALGORITMO BASEADO EM OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADO AO PLANEJAMENTO DE CAMINHOS PARA ROBÔ MÓVEL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Arthur Plínio de Souza Braga

FORTALEZA

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G612i Gomes, Allan Costa.
Implementação de algoritmo baseado em otimização por colônia de formigas aplicado ao planejamento de caminhos para robô móvel / Allan Costa Gomes. – 2018.
50 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2018.
Orientação: Prof. Dr. Arthur Plínio de Souza Braga.

1. Planejamento de caminhos. 2. Otimização por Colônia de Formigas. 3. Robótica móvel. 4. Meta-heurísticas. I. Título.

CDD 621.3

ALLAN COSTA GOMES

IMPLEMENTAÇÃO DE ALGORITMO BASEADO EM OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS APLICADO AO PLANEJAMENTO DE CAMINHOS PARA ROBÔ MÓVEL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Arthur Plínio de Souza Braga (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Giovanni Cordeiro Barroso
Universidade Federal do Ceará (UFC)

Eng. Vinicius dos Reis Alves Ferreira
Universidade Federal do Ceará (PPGEE-UFC)
Companhia Siderúrgica do Pécem (CSP)

A minha mãe, Maria Luciete, a pessoa mais importante da minha vida, a quem devo tudo que tenho e sou. Minha gratidão para com a senhora é eterna.

AGRADECIMENTOS

A meus pais e minha família, por formarem toda a base de minha formação como pessoa e pelo suporte durante todos esses anos.

A meus muitos amigos formados durante toda a graduação, em especial a Filipe Marques, Hallison Lima, Juscelino Taleires, Mateus Pinheiro, Rafael Sousa, Raimundo Jackson e Vinícius Alexandre, por todos esses anos de estudo e companheirismo.

Ao meus amigos e colegas Marcus Davi, Thiago Azevedo e Claudio Cézar, por toda a disponibilidade e assistência sem as quais parte dos experimentos presentes neste trabalho não teriam existido.

Ao Prof. Dr. Arthur Plínio De Souza Braga, pela paciência nesses vários anos de orientação nos meus estudos acadêmicos e na elaboração deste trabalho de conclusão de curso.

Ao Prof. Dr. Fabrício Gonzalez Nogueira e Prof. Dr. Bismark Claure Torrico, responsáveis pelo Grupo de Pesquisa em Automação, Controle e Robótica, por toda a estrutura fornecida para que fosse possível a realização deste trabalho.

Ao Departamento de Engenharia Elétrica, Centro de Tecnologia e Universidade Federal do Ceará, pelos anos de estudo em que pude me aproveitar de toda a infraestrutura disponibilizada para que pudesse completar a minha formação.

A Ednardo Moreira Rodrigues e Alan Batista de Oliveira, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

RESUMO

O planejamento de caminhos para robôs móveis é um dos temas centrais na área de robótica móvel, uma vez que espera-se que os robôs sejam capazes de planejar seus deslocamentos autonomamente. Inúmeras estratégias foram e vem sendo desenvolvidas ao longo das últimas décadas. Algumas destas baseiam-se em algum ponto na realização de uma busca por um caminho ótimo em um espaço de possíveis soluções geradas. Neste contexto, com o desenvolvimento nas últimas duas décadas das meta-heurísticas, que são capazes de fornecer métodos eficazes para a realização dessas buscas, estas surgem como uma alternativa em desenvolvimento a utilização de métodos clássicos para fornecer boas soluções ao problema de planejamento de caminhos. Dito isto, este trabalho trata da implementação de dois algoritmos baseados em *Ant Colony Optimization* (ACO) aplicado ao planejamento de caminhos para robôs móveis aliado à estratégia clássica de decomposição em células do espaço de trabalho do robô. São os algoritmos: (1) *Max-Min Ant System* (MMAS) e (2) *Fast Two-Stages Max-Min Ant System* (FTS-MMAS). O objetivo destes é encontrar um caminho ótimo no mapa decomposto. Para verificar a eficiência de toda a estratégia implementada, os algoritmos ACO foram submetidos a uma série de testes simulados, visando mensurar o desempenho destes em relação a tempo de execução e comprimento do caminho obtido tanto em um cenário simples, dois cenários mais complexos e um cenário de maior dimensão. Por final, um experimento em um robô real foi realizado, tendo como principal objetivo atestar se o robô seria capaz de executar o caminho planejado. Os resultados para os testes simulados indicaram que os algoritmos ACO implementados tem desempenhos semelhantes em relação ao comprimento dos caminhos possíveis. Em relação ao tempo de execução, observou-se diferença no desempenho, sendo o algoritmo FTS-MMAS mais veloz. O algoritmo MMAS apresentou um tempo de execução significantemente maior. O experimento realizado no robô real com o algoritmo FTS-MMAS teve desempenho aceitável nos critérios de comprimento de caminho e tempo de planejamento. Observou-se discrepâncias entre o caminho planejado e o executado, principalmente nas curvas, devido as restrições não-holonômicas e o controlador que gera os sinais para os motores. Por fim, este trabalho fornece uma base no Grupo de Pesquisa em Automação e Robótica (GPAR) no DEE/UFC para futuros trabalhos em planejamento de caminhos, que envolvem temas como Controle e Inteligência Computacional.

Palavras-chave: Planejamento de caminhos. Otimização por Colônia de Formigas. Robótica móvel. Meta-heurísticas.

ABSTRACT

Path planning for mobile robots is one of the central themes in mobile robotics, as robots are expected to be able to plan their movements autonomously. Numerous strategies have been developed over the last few decades. Some of these are based at some point on the realization of a search for an optimal path in a space of possible solutions generated. In this context, with the development of metaheuristics in the last two decades, which are able to provide effective methods for the realization of these searches, these arise as a developing alternative to the use of classic methods to provide good solutions to the problem of path planning. That said, this work deals with the implementation of two algorithms based on Ant Colony Optimization (ACO) applied to the path planning problem for mobile robots allied to the classic strategy of cells decomposition of the robot workspace. The algorithms are: (1) Max-Min Ant System (MMAS) and (2) Fast Two-Stages Max-Min Ant System (FTS-MMAS). The purpose of the algorithms is to find an optimal path in the decomposed map. In order to verify the efficiency of all the implemented strategy, the ACO algorithms were submitted to a series of simulated tests, in order to measure its performance in relation to execution time and path length obtained in a simple scenario, two more complex scenarios and one scenario of larger size. At last, an experiment in a real robot was carried out, this one having as main objective to certify if the robot would be able to execute the planned path. The results for the simulated tests indicated that the ACO algorithms implemented have similar performances in relation to the length of the possible paths. In relation to the execution time, a difference in performance was observed, with the FTS-MMAS algorithm being faster. The MMAS algorithm presented a significantly longer execution time. The experiment performed in the real robot with the FTS-MMAS algorithm had acceptable performance in the path length and planning time criteria. Discrepancies between the planned and executed paths were observed, mainly in the curves, due to the non-holonomic constraints and the controller that generates the signals for the motors. Finally, this work provides a basis for the Automation and Robotics Research Group (GPAR - Grupo de Pesquisa em Automação e Robótica) in the DEE/UFC for future path planning work, involving topics such as Control and Computational Intelligence.

Keywords: Path planning. Ant Colony Optimization. Mobile robotics. Meta-heuristics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACO	*Ant Colony Optimization*
FTS-MMAS	*Fast Two-Stages Max-Min Ant System*
GPAR	Grupo de Pesquisa em Automação e Robótica
MMAS	*Max-Min Ant System*
PCM	Problema do Caminho Mínimo
SLAM	Localização e Mapeamento Simultâneos

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos últimos anos, as meta-heurísticas vem se tornando ferramentas cada vez mais utilizadas no campo da inteligência computacional para oferecer soluções alternativas para problemas de otimização combinatória. Na Figura 1, apresenta-se um gráfico que ilustra o número de trabalhos publicados na plataforma do *IEEE Xplorer* que tem somente *metaheuristics* como palavra-chave associada.

Figura 1 – Quantidade de artigos publicado tendo *metaheuristics* como palavra chave na plataforma do *IEEE Xplorer* desde 2000.

Fonte: Autoria própria.

Sobretudo que, se fossem adicionados à pesquisa os termos associados a cada tipo de meta-heurística em si, o número de artigos seria consideravelmente superior.

A principal motivação para o crescimento do uso das meta-heurísticas é que estas fornecem soluções aproximadas para os problemas de otimização combinatória e são em sua maioria flexíveis, podendo a mesma meta-heurística ser adaptada para diversos tipos de problemas. Problemas estes que muitas vezes não possuem soluções exatas, ou se possuem, são impraticáveis devido a alta complexidade associada ao próprio problema.

O termo meta-heurística, por si só, ainda é recente, por isso não se pode afirmar que existe um consenso em torno de uma definição geral para este, embora diversos autores já tenham abordado este tema, muito devido a relevância que este vem ganhando nas últimas duas décadas. De modo geral, uma meta-heurística é uma estrutura de alto nível, independente de qualquer problema, que fornece um conjunto de estratégia e guias para o desenvolvimento

de heurísticas para algoritmos de otimização. O termo também é utilizado para se referir aos algoritmos criados que se baseiam nesta estrutura (SÖRENSEN *et al.*, 2017).

Entre algumas meta-heurísticas de grande destaque, como Algoritmos Genéticos, Otimização por Enxame de Partículas e Computação Evolutiva, a ACO vem sendo aplicada com sucesso a diversos problemas de otimização. Desenvolvida inicialmente por Dorigo ao longo de sua tese (DORIGO, 1992), esta meta-heurística inspira-se no comportamento de forrageamento de algumas espécies de formigas para propor uma estrutura capaz de solucionar problemas de otimização.

Um desses problemas, tratado neste trabalho, é o de planejamento de caminho de robôs móveis, um problema elementar da área de robótica. Uma vez que autonomia é uma característica esperada em robôs móveis, é natural que estes sejam capazes de planejar caminhos no espaço por onde possam se locomover de modo seguro e otimizado.

Devido a essa necessidade de se planejar caminhos, diversas estratégias para solucionar este problema foram elaboradas (RUSSELL; NORVIG, 2010; SICILIANO *et al.*, 2009; CHOSET *et al.*, 2005). E em geral, todas essas estratégias necessitam da utilização de técnicas de otimização, uma vez que os caminhos encontrados devem ser os melhores possíveis de acordo com os parâmetros desejados, como distância, segurança, gasto energético. Por isso é muito comum a utilização de meta-heurísticas, incluindo ACO, como parte da estratégia adotada para se realizar o planejamento de escolher um caminho ótimo.

Visto isto, neste trabalho se realiza a implementação de uma estratégia de planejamento de caminhos baseada em decomposição de células, através da qual gera-se um grafo que representa com boa precisão o espaço de trabalho no qual o robô está inserido. E sobre o grafo gerado realiza-se uma busca por um caminho ótimo em relação ao comprimento através da utilização de um algoritmo ACO. Os testes são realizados tanto em simulação, para verificar o desempenho da estrategia utilizada, assim como em um robô real, que é o objetivo final deste trabalho.

A organização é feita da seguinte forma. No Capítulo 2 estabelece-se e define-se o problema de planejamento de caminhos. Em seguida, descreve-se a estratégia de decomposição em células do espaço de trabalho e como o grafo que é utilizado para se realizar a busca pelo caminho mais adequado é feita. No Capítulo 3 introduz-se a meta-heurística ACO e em seguida descreve-se o algoritmo base utilizado para realizar a busca do melhor caminho no grafo. No Capítulo 4 descreve-se uma alteração introduzida para melhorar o desempenho do algoritmo. Os

testes e os resultados são descritos e apresentados no Capítulo 5. E por final, as conclusões e sugestões para trabalhos futuros baseado nos resultados obtidos são apresentadas no Capítulo 6.

2 PROBLEMA DE PLANEJAMENTO DE CAMINHOS PARA ROBÔS MÓVEIS

No campo da robótica, uma das tarefas essenciais a ser realizada pelos robôs é o planejamento de caminhos de modo autônomo entre uma posição inicial e um destino. Este caminho deve ser otimizado sob o aspecto de certos parâmetros preteridos, podendo-se citar como exemplo: distância percorrida, consumo de energia e velocidade de tráfego. Além disso, este caminho deve atender à condição de ser livre de obstáculos que interfiram no cumprimento desse caminho pelo robô tal qual planejado.

Para que um robô possa planejar um caminho, o conjunto de limitações sobre as quais o robô deve trabalhar, devem ser caracterizadas para que assim se possa determinar quais são as necessidades de recursos que o robô demandará. Ao longo deste capítulo estas limitações sobre as quais o robô será utilizado e o próprio robô serão descritos. Além disso, fornece-se as definições dos conceitos utilizados no decorrer do trabalho.

Este capítulo é organizado da seguinte forma. Na Seção 2.1 descreve-se formalmente o problema de planejamento de caminhos, tendo por base os conceitos de espaço de estados e colisões. Na Seção 2.2 o robô é descrito fisicamente, assim como é apresentado o modelo de relação do mesmo com o espaço de estados e os obstáculos. Na Seção 2.3 descreve-se o método utilizado para se realizar o planejamento. Na Seção 2.4 é mostrado como se captura o mapa utilizado, através do uso de sensores e de algoritmos. O Capítulo 3 é reservado à técnica utilizada para se realizar o planejamento de trajetória.

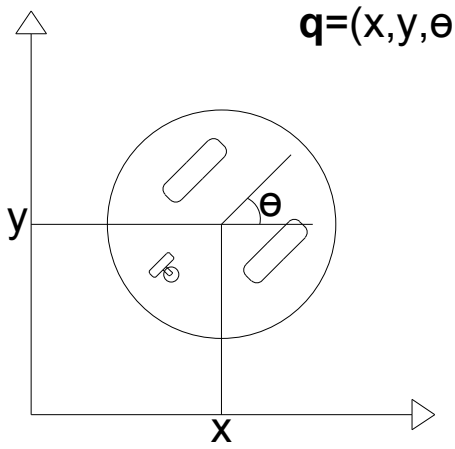
2.1 Problema de planejamento de caminhos

Afim de definir o problema de planejamento de caminhos, faz-se necessário estabelecer conceitos preliminares. Define-se como espaço de trabalho $\mathbf{W}$ o ambiente real no qual o robô está inserido. Denomina-se como estado, ou configuração, a localização do robô no espaço de trabalho, representada por um vetor de variáveis $\mathbf{q}$ de dimensão igual ao número de graus de liberdade de movimentação do robô. Ao conjunto de todos os estados possíveis que o robô pode assumir chama-se de espaço de estados $\mathbf{Q}$ (SPONG *et al.*, 2005).

Como exemplo, para a articulação robótica que pode girar sobre o próprio eixo, o estado desta pode ser representada pelo vetor $\mathbf{q} = (\theta)$ e o espaço de estados $\mathbf{Q}$ é o circulo unitário, comumente representado como $\mathbf{SO(2)}$. Para um braço robótico cartesiano, livre para se mover pelo plano, o vetor de variáveis é $\mathbf{q} = (x, y)$ e o espaço de estados é $\mathbf{Q} = \mathbb{R}^2$.

Para um robô móvel de tração diferencial, em que se considera que este se movimenta apenas no plano, o vetor de variáveis é $\mathbf{q} = (x, y, \theta)$ e o espaço de estados é representado como $\mathbf{Q} = \mathbb{R}^2 \times \mathbf{SO(2)}$, ilustrado na Figura 2.

Figura 2 – Espaço de estados para robô de tração diferencial



Fonte: Autoria própria.

Outro conceito preliminar a ser tratado é o de colisão. Entende-se que uma colisão ocorre quando um robô entra em contato com um objeto no espaço de trabalho no qual estes estão inseridos. Um dos requisitos de um algoritmo planejador de caminhos é que os caminhos sejam livres de colisões a fim de que o robô chegue à posição de destino sem colidir com obstáculos.

Formalmente, para um robô **A** em uma estado **q**, tem-se por $\mathbf{A}(\mathbf{q})$ o espaço ocupado pelo robô no espaço de trabalho. Seja também denominado por **O** o conjunto de obstáculos localizado no espaço de trabalho. A fim de se planejar um caminho livre de colisões, deve-se assegurar que o robô não se mova para qualquer estado **q** que faça com que o espaço ocupado pelo robô $\mathbf{A}(\mathbf{q})$ seja o mesmo ocupado por qualquer obstáculo pertencente a **O**.

O conjunto de estados no qual o robô colide com um obstáculo **QO** é definido por: $\mathbf{QO} = \{\mathbf{q} \in \mathbf{Q} \mid \mathbf{A}(\mathbf{q}) \cap \mathbf{O} \neq \emptyset\}$ e é denominado de espaço de estados de obstáculos. O espaço de estados livre de colisões $\mathbf{Q_{free}}$ pode então ser determinado como $\mathbf{Q_{free}} = \mathbf{Q} - \mathbf{QO}$.

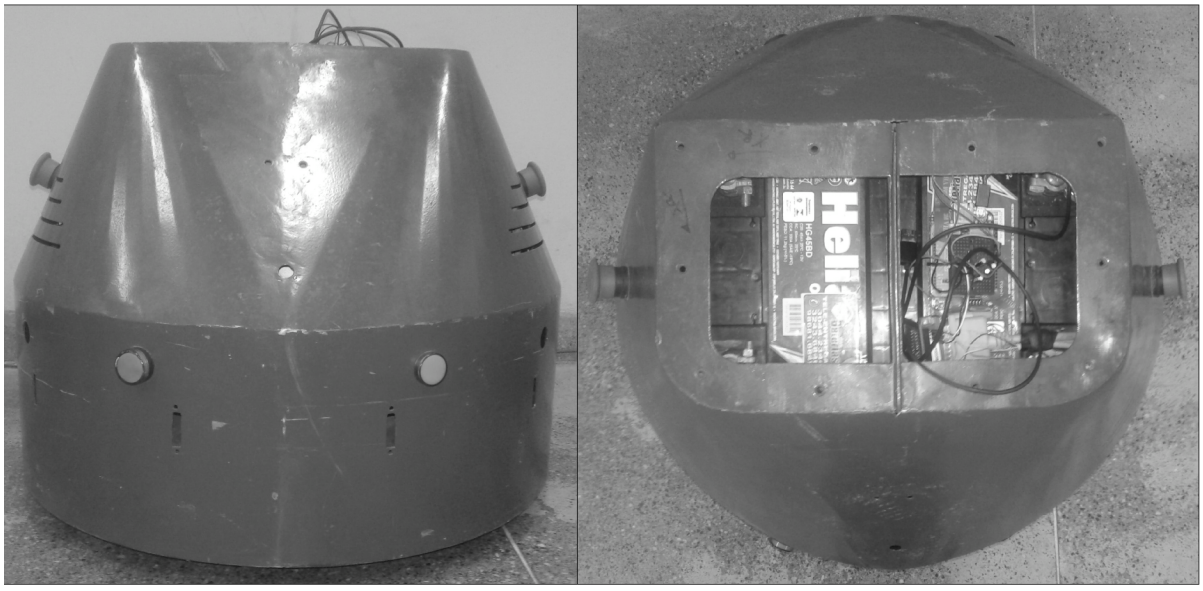
Com isso define-se o problema de planejamento de caminho como o problema de encontrar a partir de um estado inicial $\mathbf{q_{inicial}}$ um caminho no espaço de estados livre de colisões $\mathbf{Q_{free}}$ até um estado objetivo $\mathbf{q_{final}}$. Caracteriza-se um caminho livre de colisões como um conjunto de pontos localizados no espaço de estados livres do robô representado pela função $\tau : [0,1] \rightarrow \mathbf{Q_{free}}$, em que $\tau(0) = \mathbf{q_{inicial}}$ e $\tau(1) = \mathbf{q_{final}}$. É importante aqui diferenciar o conceito de caminho e trajetória, enquanto o primeiro é definido acima neste parágrafo, o segundo é definido como uma função do tempo sobre o caminho, sobre o qual se pode obter velocidades

e acelerações que devem ser utilizadas como referência para o robô. Neste trabalhado, o planejamento é realizado somente a nível de se obter caminhos (SPONG *et al.*, 2005).

2.2 Descrição física e modelagem do robô

Tomou-se como base para a elaboração deste trabalho o robô Hulk (Figura 3), visto que um dos objetivos é a realização do experimento real no robô. O robô apresenta geometria aproximadamente circular, sendo tracionado por duas rodas acionadas por motores DC, além da presença de duas rodas guia para fornecimento de equilíbrio, podendo essas rodas se moverem sem restrição em qualquer direção. O controle de velocidade de cada roda é feito via *driver*, modelo HDC2450 da RoboteQ^{TM}. O robô conta ainda com controle de seguimento de trajetória, implementado em (FORTE *et al.*, 2018).

Figura 3 – Visão frontal e superior do robô Hulk.



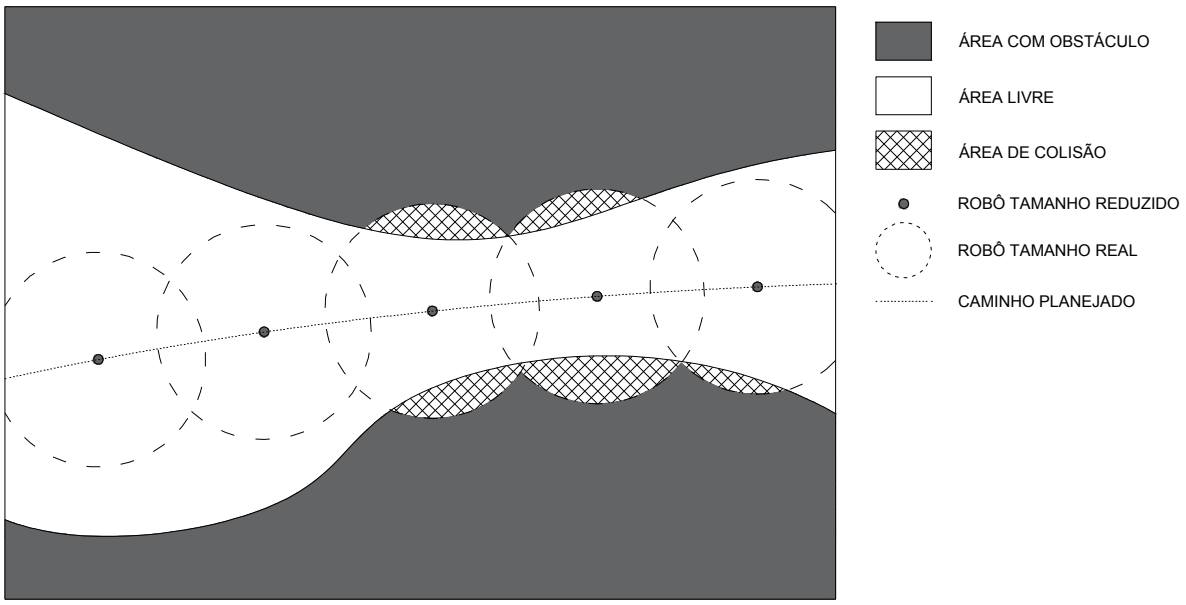
Fonte: Autoria própria.

Para fins de planejamento, visto que o robô apresenta geometria circular, utilizou-se a modelagem descrita em (CHOSET *et al.*, 2005). Neste modelo, o robô circular é reduzido a um ponto no espaço, sendo este ponto o centro geométrico do robô. Considera-se ainda para fins de planejamento de caminho que o robô não apresenta restrições cinemáticas e dinâmicas, sendo livre para se movimentar em todas as direções do espaço. Levando-se em conta esses fatores, a tarefa de planejamento de caminho tem sua complexidade reduzida ao nível geométrico (OTTONI, 2000). Devido a adoção dessas prerrogativas, não há restrições no modelo de movimento utilizado para se realizar o planejamento, e somente os obstáculos podem se interpor

entre o robô e a posição de destino.

Um dos problemas resultantes da consideração do robô ser reduzido a um ponto é que, ao se desprezar as dimensões reais do robô, existe a possibilidade de colisão com um obstáculo mesmo que o caminho planejado seja livre destes. Essa consequência é ilustrada na Figura 4, onde pode-se observar que caso o caminho planejado para um robô pontual leve o robô real a passar a uma distância menor que o raio deste em relação ao obstáculo, haverá como consequência uma colisão.

Figura 4 – Representação de colisão em caminho livre devido a não consideração das dimensões do robô.



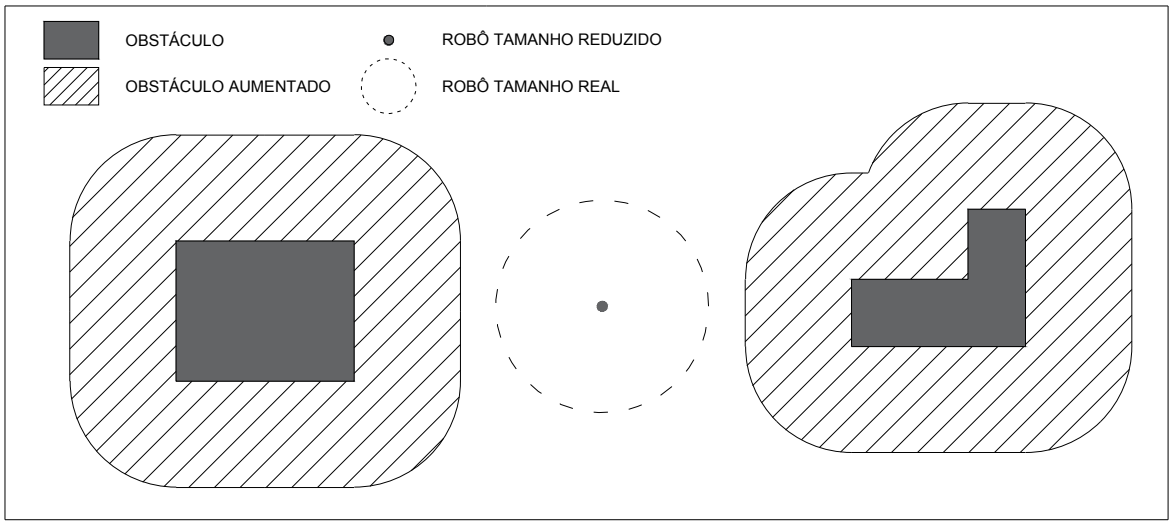
Fonte: Autoria própria.

Para se corrigir este problema, aproveitando-se do fato de o robô ter formato circular, realiza-se um aumento nas dimensões dos obstáculos (CHOSET *et al.*, 2005). Esta tarefa é feita ao se aumentar as dimensões dos obstáculos em todas as direções por um tamanho igual ou superior ao raio do robô (Figura 5). Ao se fazer isso, garante-se que o caminho planejado não se aproximará de uma distância inferior ao raio do robô, garantindo a não ocorrência de colisões por falta do planejador.

2.3 Metodologia de planejamento de caminho adotada

De modo geral são diversas as abordagens utilizadas para se solucionar o problema de planejar caminhos para robôs. Entre as abordagens mais utilizadas pode-se citar o método de planejamento via utilização de campos potenciais artificiais. Métodos ligados à esqueletização

Figura 5 – Representação do processo de aumento dos obstáculos em função das dimensões de um robô circular.



Fonte: Autoria própria.

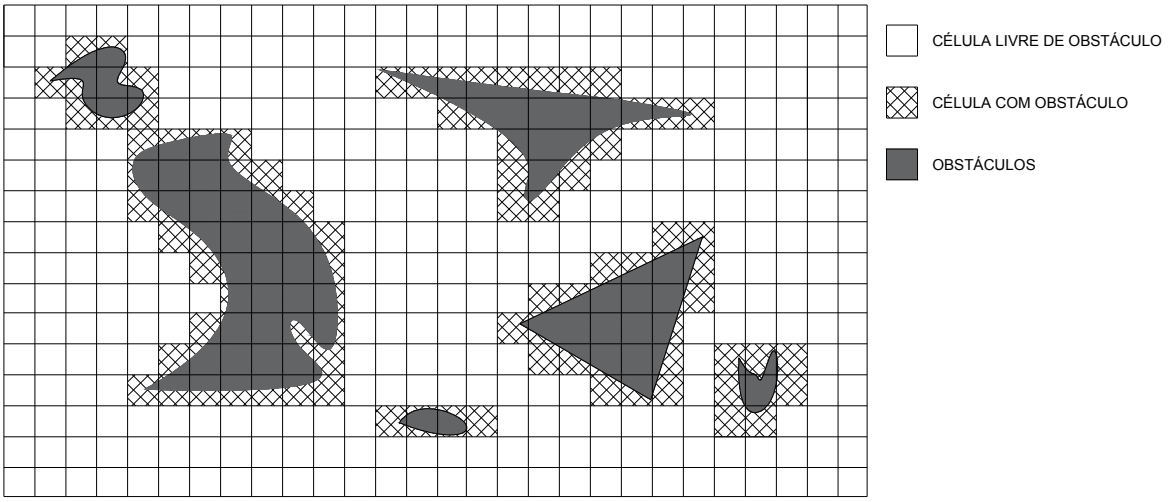
do espaço de trabalho, como construção de grafos de Voronoi e utilização de rotas probabilísticas. Além de métodos baseados na decomposição em células do espaço de trabalho (SICILIANO *et al.*, 2009; RUSSELL; NORVIG, 2010), que é o utilizado neste trabalho.

Através do método de decomposição em células decompõe-se o espaço de trabalho, que é contínuo, em um número finito de regiões que contém dados relativos à localização da célula e a presença ou não de obstáculos. Essas regiões podem apresentar quaisquer tipos de formato, porém, por questão de simplicidade optou-se pela formação de uma malha de células quadradas de dimensões iguais entre si (Figura 6).

No mapa dividido em células, como na 6, o processo de aumento dos obstáculos é feito de modo que, baseado na relação p entre o raio do robô e a resolução das células, para cada célula ocupada c, altera-se o estado de ocupação de todas as células ao redor de c em um raio de p tanto no eixo horizontal como no eixo vertical para células ocupadas. Ou seja, para cada célula c ocupada, toda as células dentro de um quadrado de lado $2p$ cujo centro é c passam para o estado de ocupadas.

Devido a utilização do método de decomposição em células e acrescentando isto ao fato de que não se levará em conta as restrições mecânicas do robô para se realizar o planejamento, o problema se reduz a encontrar um caminho em um grafo discreto, que pode ser obtido a partir do mapa de células. No caso, os vértices do grafo passam a ser representados pelas coordenadas centrais das células e os arcos são representados pela estrutura de vizinhança entre as células, onde neste grafo o robô somente pode se mover entre células vizinhas.

Figura 6 – Exemplo de decomposição do espaço de trabalho através do método de decomposição em células.



Fonte: Autoria própria.

Em particular, o tipo de estrutura de vizinhança utilizada é um ponto crítico na construção do grafo (SOUZA, 2008). Uma vizinhança simples, ou seja, poucos arcos por vértice, indica um número reduzido de buscas e uma possível queda da qualidade da solução, ao contrário, uma vizinhança muito grande, que leva a muitos arcos por vértice, indica um número alto de buscas levando a um alto tempo de processamento.

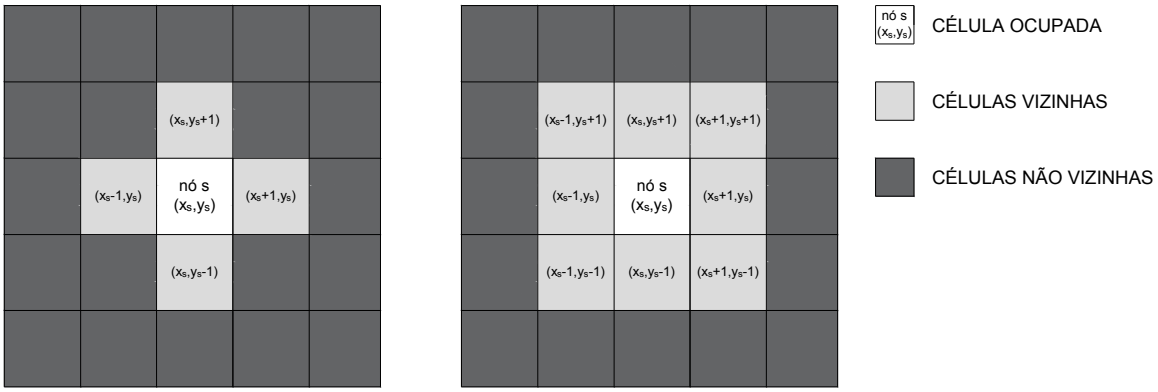
A fim de exemplificar a questão acima, tem-se na Figura 7 estruturas de vizinhança definidas como NP4(s) e NP8(s) (SOUZA, 2008). O custo para se mover ao vértice t a partir do vértice s para a estrutura NP4(s) é de, no mínimo, 3 e para a estrutura NP8(p) é de, no mínimo, $\sqrt{5}$. Porém, para esses casos, para a estrutura NP4(s), realiza-se no máximo 12 e no mínimo 9 buscas em 3 iterações, enquanto que para a estrutura NP8(s) pode-se realizar até 16 e no mínimo 9 buscas em 2 iterações, dependendo do modo como o código é estruturado. Devido a simplicidade, a estrutura utilizada ao longo deste trabalho foi a estrutura de vizinhança NP4(s).

2.4 Percepção do espaço de trabalho

Para que o robô faça uma leitura do espaço de trabalho, a fim de se gerar o mapa sobre o qual se realizará o planejamento de caminho utiliza-se um sensor de varredura a laser, modelo URG-04LX-UG01 da Hokuyo^{TM}. O sensor produz como saída 683 sinais que representam a varredura do espaço sobre uma área de 240° com profundidade de até 4 metros a uma frequência de 10 ciclos por segundo (HOKUYO, 2009).

Para realizar a coleta e tratamento desses dados, utiliza-se o *framework Robot*

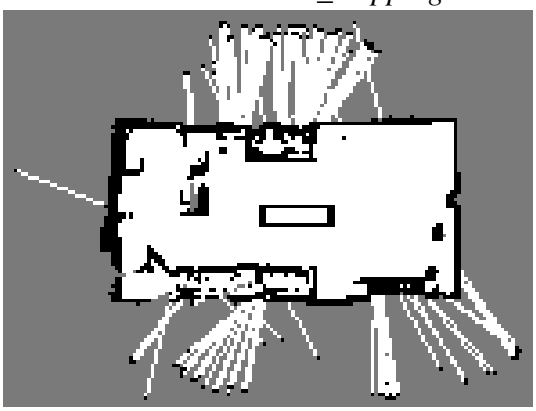
Figura 7 – Estruturas de vizinhanças NP4(s) e NP8(s).



Fonte: Autoria própria.

Operation System (ROS), em específico, o pacote *hector_mapping* (KOHLBRECHER *et al.*, 2011). Este pacote se baseia em um algoritmo capaz de realizar o processo de Localização e Mapeamento Simultâneos (SLAM), além de salvar o mapa completo do espaço de trabalho decomposto em células. A Figura 8 foi gerada a partir dos dados salvos após a execução do algoritmo, onde cria-se um mapa de resolução de 255x255 e cada célula tem dimensão de 5 centímetros de lado. As áreas em branco representam células livres de obstáculos, as células em preto representam células ocupadas por obstáculos e as células em cinza representam áreas não identificadas ou não exploradas, que são tratadas neste trabalho como células ocupadas.

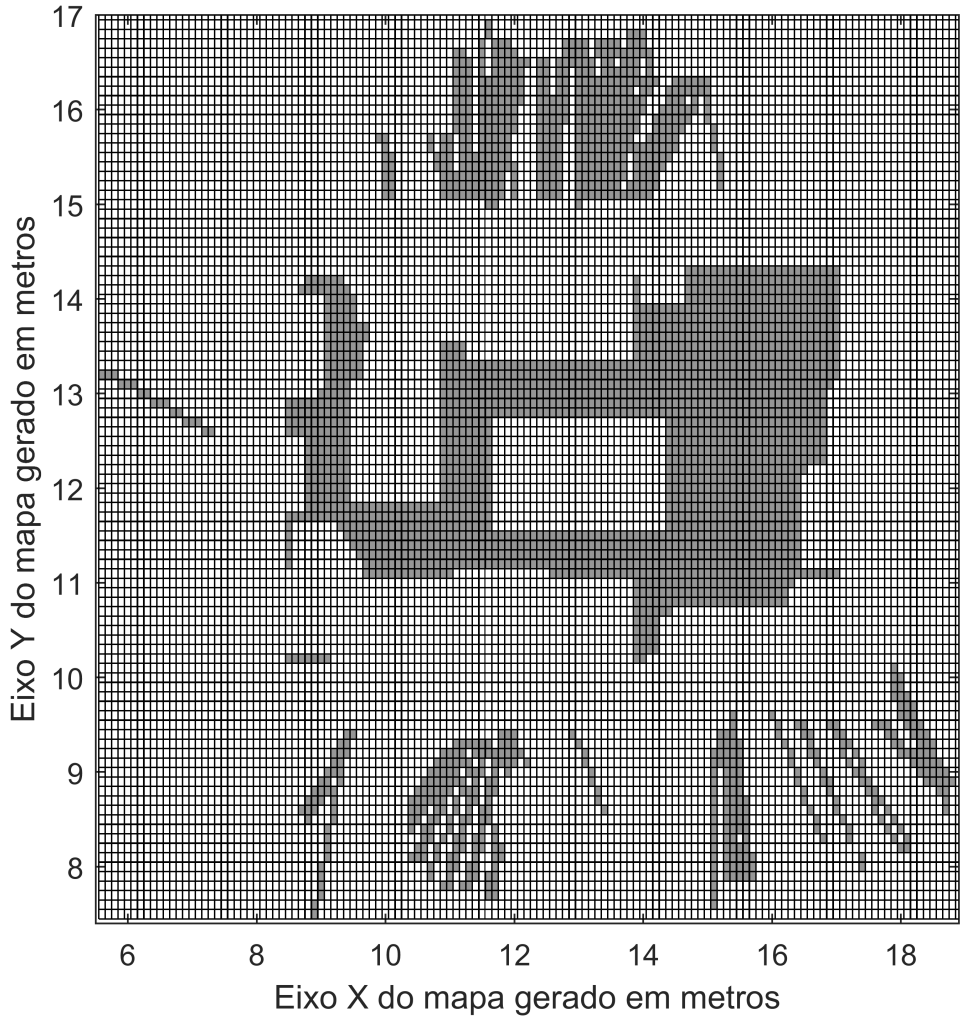
Figura 8 – Mapa produzido a partir dos dados gerados pela execução dos algoritmos do pacote *hector_mapping*.



Fonte: Autoria própria.

A partir desses dados, constrói-se, via software, o grafo utilizado para se realizar o planejamento, como ilustrado na Figura 9, de acordo como estabelecido na Seção 2.3, incluindo o processo de aumento dos obstáculos. A partir deste grafo se aplica o algoritmo planejador de caminhos.

Figura 9 – Exemplo de grafo gerado a partir dos dados colhidos no mapeamento do espaço de trabalho.



Fonte: Autoria própria.

Neste capítulo definiu-se o problema de planejamento de caminhos e a estratégia adotada para soluciona-lo. A estratégia baseia na utilização do método de decomposição em células do espaço de trabalho no qual o robô está inserido. Com o espaço de trabalho decomposto, pode-se criar um grafo sobre o qual realiza-se uma busca para encontrar um caminho. Essa busca pode ser feita utilizando-se diversos algoritmos. O Capítulo 3 é destinado a apresentar o algoritmo baseado em ACO utilizado neste trabalho.

3 ALGORITMO DE PLANEJAMENTO DE CAMINHOS BASEADO EM OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

O planejamento de caminhos baseado na técnica de decomposição em células pode ser dividido em duas etapas. Na primeira etapa, se realiza a decomposição do espaço de trabalho em células a partir da qual se pode obter uma estrutura em forma de grafo, como descrito no Capítulo 2. Na segunda etapa, de posse do grafo, aplica-se um algoritmo capaz de encontrar um caminho de custo mínimo. Em particular, se a função de custo for baseada somente na distância percorrida, como adotado neste trabalho, reduz-se o problema ao Problema do Caminho Mínimo (PCM), muito comum na área de otimização combinatória (KORTE; VYGEN, 2008), e definido na Seção 3.1.

Ainda na Seção 3.1, após definido o PCM, aborda-se estratégias utilizadas para obter a solução deste. A estratégia utilizada neste trabalho consiste da utilização de um algoritmo da meta-heurística de ACO, que tem seus conceitos introduzidos a Seção 3.2. O algoritmo utilizado é descrito na seção 3.3.

3.1 Definição e estratégia para a solução do PCM

Para um grafo qualquer $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ cujos vértices $v \in \mathbf{V}$ possuem coordenadas $(x, y) \in \mathbb{R}^2$, associa-se a cada um dos arcos $e_{ij} \in \mathbf{E}$, em que i e j são vértices vizinhos do grafo, um valor de custo $c(e_{ij})$ correspondente à distância euclidiana entre os vértices i e j, tal que $c(e_{ij}) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Um caminho $\mathbf{G}_{st} = (\mathbf{V}_{st}, \mathbf{E}_{st})$ conectando dois vértices quaisquer s e t de $\mathbf{G}$, é definido aqui, como um subgrafo de $\mathbf{G}$, em que $\mathbf{V}_{st}$ é um conjunto de vértices em que s e t são, respectivamente, o primeiro e o último vértice de $\mathbf{V}_{st}$, e todos os vértices de $\mathbf{V}_{st}$ pertencem a $\mathbf{V}$. Esses vértices são conectados através dos arcos $\mathbf{E}_{st}$, em que todos os arcos de $\mathbf{E}_{st}$ também pertencem a $\mathbf{E}$.

Para o caminho $\mathbf{G}_{st}$, o comprimento total $C(\mathbf{G}_{st})$ é calculado através do somatório dos custos de todos os arcos e_{ij} pertencentes a $\mathbf{E}_{st}$ e que conectam os vértices de $\mathbf{V}_{st}$ de modo a formar o caminho, conforme:

$$C(\mathbf{G}_{st}) = \sum c(e_{ij}),\ \forall\ e_{ij} \in \mathbf{E}_{st}. \tag{3.1}$$

Com isso, é possível definir formalmente o PCM, com base em (KORTE; VYGEN, 2008), como a tarefa de se encontrar, caso exista, um caminho $\mathbf{G}_{st}$ em $\mathbf{G}$ de comprimento $C(\mathbf{G}_{st})$

mínimo, entre os vértices s e t. Caso não haja um caminho, o algoritmo deve retornar esta informação.

Para resolver o PCM, diversos algoritmos podem ser encontrados na literatura. Algoritmos clássicos, como os algoritmos de Bellman-Ford e de Dijkstra, retornam, caso exista, o caminho de menor distância existente. Porém, algoritmos deste tipo tem o desempenho em termos de tempo de execução comprometidos para grafos de altas dimensões e de estrutura complexa (CORMEN *et al.*, 2009). Estruturado de maneira semelhante ao algoritmo de Dijkstra, o algoritmo *A-Star* acrescenta a utilização de funções heurísticas para direcionar o processo de busca, podendo acelerar o tempo de execução do algoritmo (MILLINGTON; FUNGE, 2009). Porém, embora tenda a ser mais veloz, também tem o desempenho depreciado frente a grafos complexos e que apresentam altas dimensões.

Como alternativa para solucionar o PCM, métodos baseados em meta-heurísticas (Algoritmos Genéticos, Otimização por Enxame de Partículas, entre outros) apresentam-se como alternativa e são cada vez mais utilizados para essa tarefa (LI; WANG, 2010; CHEN *et al.*, 2013; HUANG *et al.*, 2014; SABRI *et al.*, 2018) por serem capazes de retornar soluções ótimas ou sub-ótimas em tempos mais praticáveis para grafos complexos e de alta dimensão. Entre estas, está a meta-heurística ACO, que é utilizada neste trabalho para a solução do PCM.

ACO é uma meta-heurística na qual uma colônia de agentes artificias, comumente chamados de formigas, cooperam entre si para encontrar soluções de problemas de otimização combinatória (DORIGO; STüTZLE, 2004), como o PCM. Essa cooperação é feita através de comunicação indireta baseada no estado do ambiente que é alterado pelas próprias formigas, através do depósito de feromônios, mecanismo esse chamado de estigmergia. Com base na estrutura da meta-heurística ACO diversos algoritmos foram e vem sendo desenvolvidos a fim de se solucionar quaisquer problemas de otimização discreta.

Entre esses algoritmos pode-se citar o *Ant System* (DORIGO *et al.*, 1996), um dos primeiros algoritmos desenvolvidos e base para o algoritmo utilizado neste trabalho, o MMAS (STüTZLE; HOOS, 2000). O MMAS foi desenvolvido inicialmente para tratar do Problema do Caixeiro Viajante e do Problema Quadrático de Alocação, porém devido a sua adaptabilidade ele pode ser aplicado a diversos problemas, como agendamento de projetos de software (CRAWFORD *et al.*, 2014) e planejamento de projetos baseado em finanças (AL-SHIHABI; ALDURGAM, 2017). Devido a essa flexibilidade, optou-se pela sua utilização, adaptando-o ao PCM.

3.2 Introdução à Otimização por Colônia de Formigas

A meta-heurística ACO tem como principal inspiração para formação de sua estrutura o comportamento de forrageamento, que é a busca e exploração de recursos alimentares, apresentado por algumas espécies de formigas. Estas depositam uma trilha de feromônios no percurso entre formigueiro e alimento cuja intensidade é dependente da quantidade e qualidade do alimento encontrado (DORIGO; STüTZLE, 2004).

Essa trilha de feromônio é o que serve de guia para levar outras formigas até o alimento, que por sua vez, também depositam feromônios. Quando a fonte de alimento se esgota, e as formigas param de depositar feromônios, devido a um processo natural de evaporação, os feromônios perdem sua intensidade e as formigas passam a explorar outros locais. Tendo por base este mecanismo, pode-se descrever um algoritmo ACO generalizado (DORIGO; BLUM, 2005).

Inicialmente, o algoritmo recebe como entrada um modelo do problema a ser solucionado. Este modelo deve conter o espaço de busca sobre o qual as formigas devem procurar as soluções além de uma função de custo, por onde se possa avaliar a qualidade da solução. Pode-se ainda inserir restrições que as soluções devem atender.

Em seguida, inicializa-se os feromônios, com valores positivos e iguais. Após isso, um número M de formigas, em cada iteração, constroem soluções para o problema. Por fim, as soluções obtidas são utilizadas para atualizar os feromônios, que influenciarão no processo de construção de soluções da iteração seguinte. O processo de construção de soluções e atualização de feromônios se repete até que algum critério de convergência seja atingido, ou um número limite de iterações seja ultrapassado.

A construção da solução para um determinado problema é feita através de uma série de decisões tomadas de modo probabilístico pelas formigas. A probabilidade p_o^m de uma formigas m em um processo de decisão escolher uma opção o dentre um conjunto de opções candidatadas $\mathbf{O}$ é dependente da quantidade de feromônios τ_o associado a cada opção e de um fator heurístico η. Logo, a probabilidade p_o^m é dada por:

$$p_o^m = \frac{\tau_o^\alpha \cdot \eta^\beta}{\sum_{o \in \mathbf{O}} (\tau_o^\alpha \cdot \eta^\beta)}. \tag{3.2}$$

Em que os parâmetros α e β são números reais positivos que são utilizados para determinar a importância relativa entre a quantidade de feromônio presente e a informação heurística.

O fator heurístico η deve ser determinado com base no tipo de problema que o algoritmo ACO deve solucionar e tem por função direcionar as formigas a certas soluções. A má utilização deste fator pode resultar na diminuição da qualidade ou limitação do tipo de solução que as formigas podem gerar. Em casos em que não se pode determinar uma heurística que auxilie positivamente na construção de boas soluções, pode-se tomar η pela unidade, e a construção da solução passa a ser baseada somente na distribuição de feromônios.

A atualização dos feromônios, de modo geral, é realizada através da aplicação de:

$$\tau_o(n+1) = \rho\,\tau_o(n) + \Delta\tau. \tag{3.3}$$

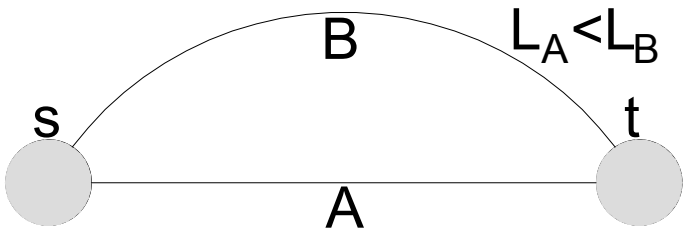
Nesta, o valor de feromônio a ser utilizado na próxima iteração $\tau_o(n+1)$ é calculado a partir do valor atual do feromônio $\tau_o(n)$ amortizado pelo fator de evaporação ρ e incrementado de um valor $\Delta\tau$ que é calculado baseado na qualidade da solução gerada avaliada pela função de custo e na estratégia adotada pelo algoritmo.

3.2.1 Experimento das duas pontes

A fim de introduzir os conceitos básicos de ACO descritos e exemplificar a utilização, reproduz-se nesta seção o experimento das duas pontes, descrito em (DORIGO; STüTZLE, 2004) e reproduzido aqui com adaptações.

Considere duas pontes, A e B, de comprimentos L_A e L_B, em que $L_A < L_B$ ligando o ponto s ao ponto t, conforme a Figura 10. Inicializa-se então uma colônia de formiga (artificiais) com M formigas no ponto s cujo objetivo é determinar qual ramo tem menor comprimento para se chegar em t. Inicialmente nos dois ramos não há feromônio algum depositado neles. Afirma-se ainda que as formigas tem capacidade de memorizar a distância percorrida entre o s e t.

Figura 10 – Representação do experimento das duas pontes.



Fonte: Autoria própria.

Inicialmente, uma vez que não há a presença de feromônio, o processo de decisão

sobre para qual ramo seguir é feito com probabilidades iguais $p_A = p_B$ de cada ramo ser escolhido. Neste caso, a esperança estatística é de que um número igual de formigas escolha cada ramo $n_A = n_B$. Ao final, quando todas as formigas chegam ao ponto t, cada uma delas deposita feromônio baseada no valor registrado de distância percorrida.

Uma vez que o objetivo é minimizar a distância, a quantidade de feromônios depositado é proporcional ao inverso da distância percorrida. Logo, espera-se que a quantidade de feromônio depositada no ramo A ($f_A = \frac{n_A}{d_A}$) seja maior que a quantidade de feromônio no ramo B ($f_B = \frac{n_B}{d_B}$), uma vez que como $d_A < d_B$ e $n_A = n_B$ tem-se que $\frac{n_A}{d_A} > \frac{n_B}{d_B}$.

Consequentemente, na iteração seguinte as formigas usarão essa informação de feromônio para determinar probabilisticamente qual ramo seguir, com $f_A > f_B$ tem-se que $p_A > p_B$. Com isso, a esperança estatística é de que um número maior de formigas escolha o ramo A, logo $n_A > n_B$. Com isso, uma quantidade ainda maior de feromônio tende a ser depositado em A em relação ao feromônio depositado em B, já que $n_A > n_B$ e $d_A < d_B$ levando a $\frac{n_A}{d_A} > \frac{n_B}{d_B}$, aumentando ainda mais a probabilidade de A ser escolhido.

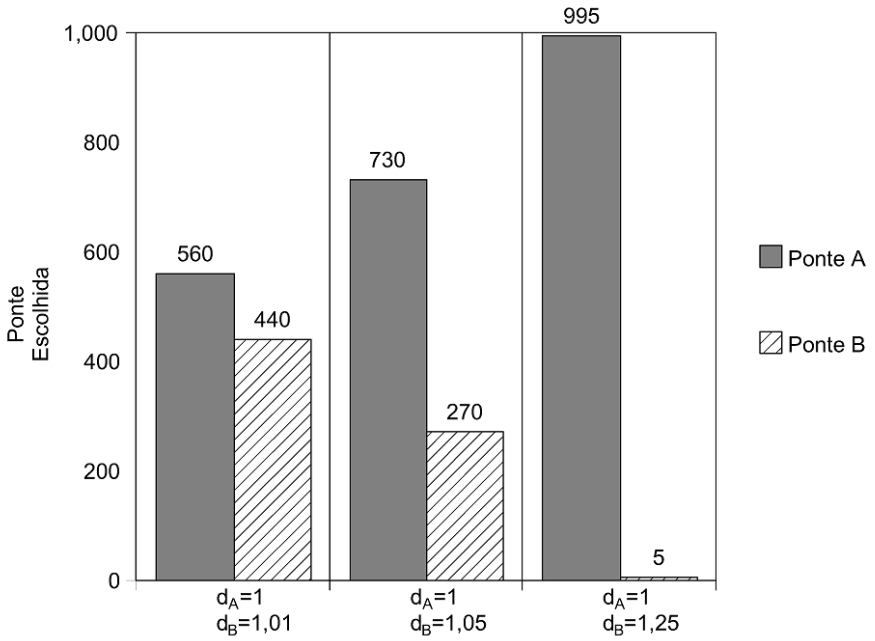
Visto isso, é simples concluir que com o passar das iterações, a probabilidade de A ser escolhido tende a ser muito maior, $p_A \gg p_B$, indicando uma convergência para o ramo A, que é o de menor comprimento. Com isso pode-se observar a característica de realimentação positiva do uso dos feromônios, em que as próprias formigas geram estímulo para melhorar sua tomada de decisão.

Obviamente, é possível que ao inicio do algoritmo uma quantidade suficientemente grande de formigas escolha o ramo B, visto que processo de decisão é probabilístico, e que isso continue a se repetir nas iterações futuras, levando as formigas a indicar incorretamente o ramo B como o de menor distância.

Para verificar esta situação, na Figura 11 apresenta-se os resultados do experimento das duas pontes realizados para três casos, variando-se o comprimento da ponte B em relação a A por um fator de 1%, 5% e 25%. Para cada caso, foram realizados 1000 execuções (nE) do processo descrito nesta seção para um número de formigas $M = 10$. O critério de parada adotado foi $p_A > 0,9$, $p_B > 0,9$ ou número de iterações para todas as M formigas maior que 10000.

Como se pode observar da Figura 11, no caso em que a diferença de comprimento é pequena (1%), o algoritmo retorna como solução as duas opções com probabilidades semelhantes. É possível também notar que quanto maior da diferença de comprimento, maior a probabilidade do algoritmo retornar a ponte A. E mesmo em casos que a diferença de comprimento é

Figura 11 – Experimento das duas pontes. nE = 1000, M = 10.



Fonte: Autoria própria.

considerável (25%) ainda é possível que o algoritmo retorne como solução a ponte *B*.

Por isso, algoritmos ACO não garantem que a solução será sempre ótima, porém, a tendência é que o caminho retornado seja sempre ótimo ou sub-ótimo, principalmente quando se adiciona as outras características comuns de algoritmos ACO, como o uso de uma heurística no processo de decisão e o efeito de evaporação dos feromônios.

3.3 Descrição do algoritmo MMAS aplicado ao PCM

O algoritmo MMAS, utilizado aqui para solucionar o problema do PCM é um algoritmo caracterizado pela inserção de valores máximos e mínimos para a quantidade de feromônio depositado. Essa característica tem como objetivo encorajar uma maior exploração pelas formigas (STüTZLE; HOOS, 2000).

Outra característica do algoritmo MMAS é que a atualização dos feromônios é elitista, isto é, utiliza-se uma regra para se determinar qual ou quais formigas podem atualizar os feromônios. O objetivo é evitar que soluções ruins sejam incluídas no processo de atualização de feromônios.

Ao longo desta Seção, descreve-se o algoritmo MMAS, aplicado ao PCM, de acordo com o a estrutura base descrita na Seção 3.2 e com as características inseridas pelo algoritmo MMAS.

3.3.1 Inicialização dos feromônio

O modelo do problema fornecido ao algoritmo consiste, para o problema do PCM como definido na Seção 3.1, de um grafo **G** que é o espaço de busca do algoritmo e uma função de custo, que para o PCM é o comprimento total da solução gerada $C(\mathbf{G}_{st})$ entre o ponto de partida (vértice s), e o ponto objetivo (vértice t), calculada conforme Equação 3.1.

Uma vez que o espaço de busca é um grafo, os feromônios podem ser inicializados tanto nos vértices como nos arcos. A vantagem de se inicializar os feromônios nos arcos é que, dados dois vértices i e j, pode-se associar diferentes feromônios aos arcos que ligam i a j e j a i. Devido a isso, a direção do movimento também é levada em consideração de modo inerente à estrutura de feromônios do algoritmo.

Por isso, adota-se a inicialização de feromônios nos arcos de **G**. O processo de inicialização dos feromônios é feito então associando a cada arco do grafo **G** um valor de feromônio inicial τ_0.

3.3.2 Construção da solução

Após a inicialização dos feromônios, as formigas podem iniciar o processo de construção da solução, no caso, construção do caminho. Para o PCM, baseado no modelo fornecido para o algoritmo MMAS, o caminho formado entre s e t é um subgrafo $\mathbf{G}_{st}$ de **G** formado por um conjunto de vértices e arcos, conforme definido na Seção 3.1.

A construção desse caminho é feita através do movimento das formigas entre os vértices que são vizinhos, ou seja, que são conectados por arcos. Partindo do vértice s até chegar ao vértice t, cada formiga se move entre vértices vizinhos pelos arcos, construindo assim os conjuntos $\mathbf{V}_{st}$ e $\mathbf{E}_{st}$.

Cada movimento das formigas é determinado de maneira probabilística, em que, para uma formiga m localizada em um vértice qualquer i, esta formiga só pode se movimentar para um vértice j pertencente à vizinhança **J** de i. Então, a probabilidade p_{ij}^{m} de m se mover de i para qualquer vértice $j \in \mathbf{J}$, para o algoritmo MMAS aplicado ao PCM, é dada por:

$$p_{ij}^{m} = \frac{\tau_{ij}^{\alpha} \cdot \eta^{\beta}}{\sum_{j \in \mathbf{J}} (\tau_{ij}^{\alpha} \cdot \eta^{\beta})}. \tag{3.4}$$

A estratégia utilizada para determinar o parâmetro η é baseada no cálculo do inverso da distância euclidiana entre os vértices j e t, uma vez que com isso a probabilidade de uma

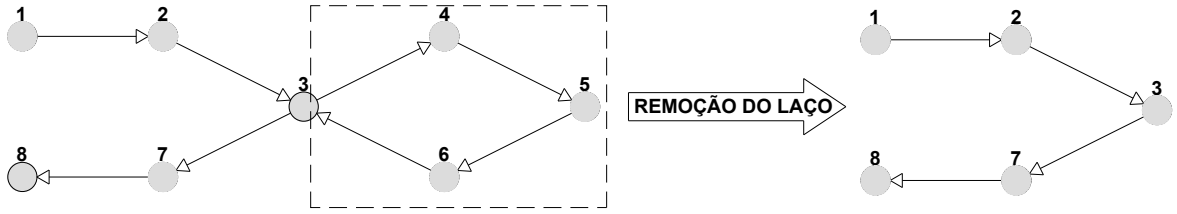
formiga escolher um vértice mais próximo de t será maior. Logo, η na Equação 3.4 pode ser determinado como:

$$\eta = \frac{1}{\sqrt{(x_j - x_t)^2 + (y_j - y_t)^2}}. \tag{3.5}$$

Durante a construção da solução, as formigas podem se deslocar entre os vértices sem restrições. Por causa disso, vértices anteriormente visitados podem ser escolhidos novamente. Quando isso ocorre, forma-se uma laço no caminho, que se removido, resulta na diminuição do comprimento total do caminho gerado. Ou seja, mesmo que uma formiga realize um alto número de movimentos, o caminho final pode ter uma distância pequena. Por isso, faz-se a remoção de laços como uma ferramenta para melhorar a qualidade dos caminhos gerados.

Esse processo de formação de laços é ilustrado na Figura 12. Nesta, apresenta-se um caminho construído, partindo do vértice 1 até o vértice 8. Neste caminho, há um laço que se forma no vértice 3, que quando removido, retira do caminho, os vértices 4, 5 e 6. Com isso, o caminho removido do laço tem seu comprimento reduzido.

Figura 12 – Exemplo de caminho que apresenta um laço antes e após a remoção deste.



Fonte: Autoria própria.

A estratégia utilizada para realizar a remoção destes laços também pode afetar a qualidade do caminho gerado. Devido a simplicidade de implementação, neste trabalho, os laços gerados são removidos no momento em que se formam. Uma outra estratégia seria esperar a formiga construir todo o caminho e então, uma vez que laços podem se formar dentro de laços, remover seletivamente os laços de modo que o caminho final seja o de menor distância possível.

Ao final do processo de construção das soluções, todas as M formigas devem ter chegado ao vértice objetivo t e armazenado na memória a solução construída.

3.3.3 Atualização dos feromônios

Como o MMAS utiliza uma estratégia elitista de atualização dos feromônios, somente a melhor solução gerada $\mathbf{G}_{st}^{best} = (\mathbf{V}_{st}, \mathbf{E}_{st})$ na iteração é utilizada na atualização dos feromônios.

Logo, para um arco e_{ab} qualquer que conecta os vértices a a b e que apresenta um valor de feromônio τ_{ab} associado, a atualização dos feromônios, é dada por:

$$\tau_{ab}(n+1) = \begin{cases} \rho\,\tau_{ab}(n) + \Delta\tau & \text{se } e_{ab} \in \mathbf{E}_{st} \\ \rho\,\tau_{ab}(n) & \text{se } e_{ab} \notin \mathbf{E}_{st} \end{cases} . \tag{3.6}$$

E $\Delta\tau$ é calculado por:

$$\Delta\tau = 1/C(\mathbf{G}_{st}^{best}). \tag{3.7}$$

Em que $C(\mathbf{G}_{st}^{best})$ o comprimento do melhor caminho obtido.

Após a atualização dos feromônios, realiza-se a checagem para se observar se os feromônios estão dentro dos limites impostos pelos parâmetros máximos e mínimos, de acordo com:

$$\tau_{ab}(n+1) = \begin{cases} \tau_{min} & \text{se } \tau_{ab}(n+1) \leq \tau_{min} \\ \tau_{ab}(n+1) & \text{se } \tau_{min} \leq \tau_{ab}(n+1) \leq \tau_{max} \\ \tau_{max} & \text{se } \tau_{max} \leq \tau_{ab}(n+1) \end{cases} . \tag{3.8}$$

A determinação dos limites de feromônios τ_{max} e τ_{min} são de alta importância para a boa execução do algoritmo. O objetivo da utilização destes limites é incentivar um leve aumento na exploração em torno de boas soluções, uma vez que τ_{min} garante a existência de uma probabilidade, mesmo que mínima, de qualquer decisão ser tomada. Enquanto isso, τ_{max} serve ao propósito de limitar a probabilidade de que um caminho específico seja muita alta, evitando uma convergência completa.

Porém, se τ_{max} e τ_{min} forem mal determinados pode-se ter duas situações extremas. Na primeira, em que τ_{max} e τ_{min} possuem magnitudes semelhantes, mesmo que o melhor caminho seja encontrado, a probabilidade de escolha deste pelas formigas, nunca será suficientemente alta para que as formigas rapidamente o escolham, tornando o processo de construção de solução lento. No caso em que τ_{max} e τ_{min} tem alta diferença de intensidade, o impacto da utilização desses limites no algoritmo é muito pequena.

Por isso, em (PELLEGRINI *et al.*, 2006), recomenda-se a utilização das seguintes expressões para se determinar τ_{max} e τ_{min}:

$$\tau_{max} = \frac{1}{C(\mathbf{G}_{st}^{BEST})}, \tag{3.9a}$$

$$\tau_{min} = \frac{\tau_{max}(1 - \sqrt[n(\mathbf{V})]{0.05})}{(\frac{n}{2} - 1)\sqrt[n(\mathbf{V})]{0.05}}. \tag{3.9b}$$

Em que $\mathbf{G}_{st}^{BEST}$ é o melhor caminho gerado durante toda a execução do algoritmo e $n(\mathbf{V})$ é o número de vértices no grafo $\mathbf{G}$.

3.3.4 Término do algoritmo

Para declarar que o algoritmo convergiu e que se pode encerrar a execução do mesmo, deve-se realizar um teste de convergência. Para o algoritmo MMAS, considera-se que o algoritmo convergiu quando, se para uma dada solução, todos os arcos dessa solução tem associado τ_{max} ao seu valor de feromônio, enquanto que todos os outros possíveis arcos que poderiam ser escolhidos durante a construção da solução tem associado τ_{min} ao seu valor de feromônio (STüTZLE; HOOS, 2000).

Além deste teste de convergência, pode-se utilizar também outros critérios de parada, como a quantidade de iterações realizada pelo algoritmo. Isso porque, em problemas de planejamento de caminho para robôs móveis, o tempo de resposta do algoritmo deve ser baixo e a convergência do algoritmo, de acordo com a definição, pode requerer do algoritmo um número elevado de iterações.

Ao encerramento do algoritmo, retorna-se o melhor caminho obtido $\mathbf{G}_{st}^{BEST}$ durante toda a execução do algoritmo.

4 APRIMORAMENTO DO ALGORITMO ACO UTILIZADO

Devido a característica estocástica do MMAS, principalmente no inicio da execução do algoritmo em que os feromônios estão distribuídos igualmente, as formigas podem demorar bastante até alcançarem o ponto de destino. Por isso, a fim de acelerar o processo, optou-se pela utilização de uma heurística proposta em (CHEN *et al.*, 2013) ao invés da heurística adotada na Seção 3.3. O algoritmo utilizando esta nova heurística será referenciado neste trabalho como FTS-MMAS.

Este Capítulo é organizado da seguinte forma. Na Seção 4.1 descreve-se essa heurística e como ela é implementada no algoritmo, fazendo-se ainda uma breve análise do funcionamento do algoritmo após a inserção desta nova heurística.

4.1 Descrição da nova heurística utilizada

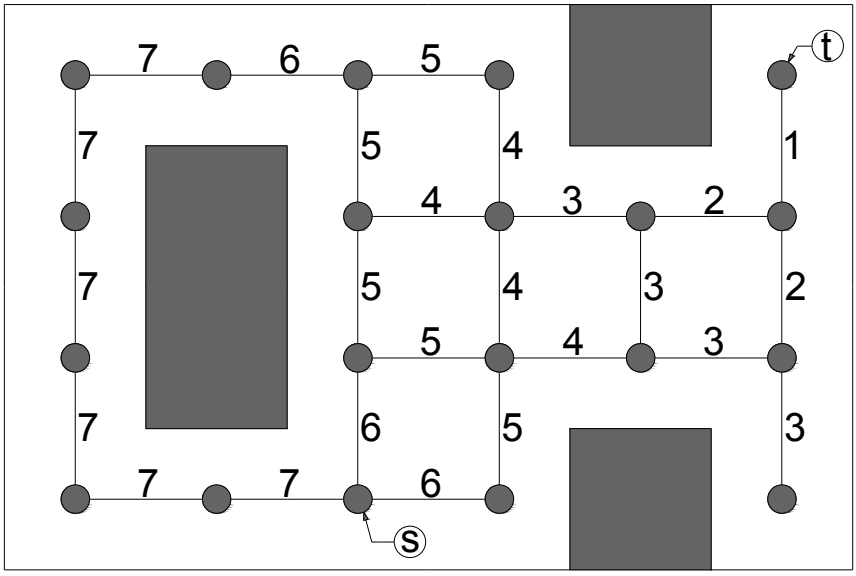
Resumidamente, a heurística utilizada associa a cada aresta um valor inteiro, chamado aqui de nível de conectividade μ. Este processo é feito antes da construção da solução pelas formigas. A partir deste nível de conectividade, aplica-se um nova equação para se determinar o fator heurístico η da Equação 3.4, reproduzida aqui:

$$p_{ij}^{m} = \frac{\tau_{ij}^{\alpha} \cdot \eta^{\beta}}{\sum_{j \in \mathbf{J}} (\tau_{ij}^{\alpha} \cdot \eta^{\beta})}.$$

O nível de conectividade $\mu(e_{ij})$ associado a cada aresta e_{ij} é determinado como descrito a seguir. A partir do vértice objetivo t, associa-se as arestas que conectam este aos seus vizinhos o valor igual a 1. Em seguida, repete-se este processo para os vizinhos de t, incrementando-se o valor associado às arestas pela unidade. Esse processo continua até que pelo menos uma das arestas do vértice de partida s tenha um valor de nível associado a si. Se durante o processo uma mesma aresta apresentar dois valores diferentes de nível de conectividade, mantém-se o menor valor. Ao final, caso ainda existam arestas sem nível de conectividade associado, associa-se o último nível obtido incrementado pela unidade a esse valor. O grafo da Figura 13 exemplifica este processo para um grafo qualquer, neste o número ao lado das arestas é o nível de conectividade.

Os níveis de conectividade são então utilizados para determinar o fator heurístico η durante o processo de construção da solução. Para uma formiga qualquer localizada no vértice i, com vértice vizinho $j \in \mathbf{J}$, em que i e j são conectados por uma aresta e_{ij} que apresenta nível de

Figura 13 – Exemplo de grafo com níveis de conectividade associado aos vértices.



Fonte: Autoria própria.

conectividade $\mu(e_{ij})$, o valor de η_{ij} é determinado pela equação:

$$\eta_{ij} = \begin{cases} 1 & \text{se } \mu(e_{ij}) = \mu_{min} \\ 0 & \text{se } \mu(e_{ij}) > \mu_{min} \end{cases} . \tag{4.1}$$

Em que μ_{min} é o menor nível de conectividade entre as arestas que conectam o vértice i aos seus vizinhos $j \in \mathbf{J}$. Devido a isto, as formigas só se movimentam entre arestas de níveis cada vez menor.

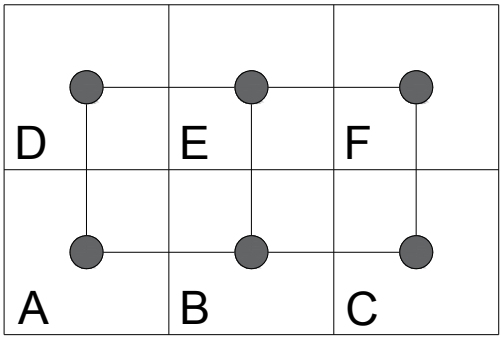
A utilização desta heurística garante que após $N = \mu_{MAX} - 1$ iterações, em que μ_{MAX} é o maior nível de conectividade em todo o grafo, a formiga chegará ao vértice de destino, ou seja, garante-se a existência de um caminho entre s e t e garante-se ainda que este caminho será ótimo (CHEN *et al.*, 2013).

A heurística utilizada aproveita-se da estrutura de mapa adotada neste trabalho, uma grade de células quadradas. Devido a este tipo de estrutura, todas as arestas do grafo $\mathbf{G}$ formado a partir do mapa, possuem o mesmo comprimento. Por causa dessa propriedade, podem existir no mesmo grafo, diversos caminhos $\mathbf{G}_{st}$ entre dois pontos, que apresentam o mesmo comprimento $C(\mathbf{G}_{st})$.

Para ilustrar essa afirmação, faz-se uso da Figura 14. Nesta tem-se um simples grafo construído a partir de uma grade de células quadradas conforme explanado na Seção 2.3. Partindo do vértice A, para se chegar ao vértice F, como se pode facilmente observar, a menor distância possível que se pode obter através do grafo é de três unidades, e neste caso, existem ao todo

três caminhos que apresentam este comprimento, a saber, os caminhos (A-B-C-F), (A-B-E-F) e (A-D-E-F). O outro caminho possível (A-D-E-B-C-F) apresenta cinco unidades de comprimento.

Figura 14 – Exemplo de grafo construído baseado em grade de células quadradas.



Fonte: Autoria própria.

Devido a heurística utilizada, para este tipo de grafo, os caminhos formados pelas formigas serão sempre os mais otimizados. Logo, a utilização desta heurística implica na redução do espaço de busca das formigas a somente as soluções ótimas em termos de comprimento. No caso do grafo da Figura 14, as formigas só construirão os caminhos (A-B-C-F), (A-B-E-F) e (A-D-E-F).

Por causa desta característica, a fim de melhor utilizar os recursos do algoritmo, algumas mudanças na estrutura deste podem ser realizadas. Já que o único parâmetro que se busca minimizar pelas formigas é o comprimento do caminho e o caminho será sempre ótimo em relação a isto, não há necessidade de se utilizar diversas formigas e nem de se repetir o processo de construção por varias iterações até que haja a convergência do algoritmo.

Por isso, para a versão modificada cujo objetivo é somente encontrar o caminho de menor comprimento utiliza-se somente uma única formiga em uma única iteração. Caso existissem outros parâmetros a serem minimizados pelas formigas, esta estratégia não deveria ser utilizada.

A adoção desta estratégia implica em outra mudança no algoritmo. Como o processo todo ocorre em somente uma única iteração, todo o procedimento de atualização, e consequentemente, de utilização dos feromônios deixam de ter impacto, podendo estes procedimentos serem retirados do algoritmo. Essas alterações fazem com que o algoritmo passe a ter um comportamento determinístico, uma vez que elementos característicos de algoritmos ACO passam a não interferir ou interferir pouco na solução do problema. Porém, caso adicione-se ao algoritmo outros parâmetros a serem otimizados, como a proximidade de obstáculos ao longo do caminho, essas alterações não seriam aplicáveis.

Devido a estas mudanças, espera-se que haja um ganho de tempo consideravelmente maior em relação ao algoritmo MMAS original, tornando esta versão modificada do algoritmo mais apta a ser utilizada em aplicações de robótica, que exigem decisões tomadas o mais rapidamente possível.

Visto isto, o principal limitador de velocidade do algoritmo com a utilização desta heurística, especialmente para grafos de alta dimensão, passa a ser o processo de associação de níveis às arestas do grafo, visto que para se realizar este processo necessita-se realizar uma varredura no grafo.

5 EXPERIMENTAÇÃO E RESULTADOS

A fim de verificar o desempenho do algoritmo, na Seção 5.1 submete-se este a uma série de testes simulados. O experimento realizado no robô é tratado na Seção 5.2.

5.1 Experimento Simulado

Foram realizadas simulações a fim de observar o comportamento de ambos os algoritmos MMAS e FTS-MMAS em três cenários distintos. Ao longo desta Seção cada um desses cenários é descrito, assim como os resultados obtidos são apresentados e analisados.

Os algoritmos foram implementados em MATLAB. Para se realizar medições do tempo de execução do algoritmo e produzir gráficos utilizou-se as funções deste. Os testes foram realizados em um computador equipado com um processador Intel Core I3-2328 e 6GB de memória RAM.

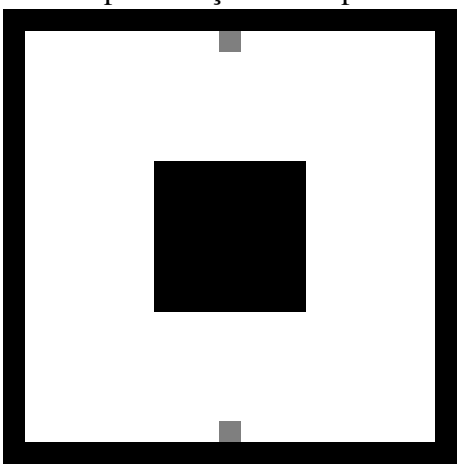
5.1.1 Cenário A

O Cenário A é o mais simples dentre todos os cenários analisados. O objetivo deste é verificar se o algoritmo é capaz de encontrar caminhos livres de obstáculos, que representam o objetivo central do planejamento de caminhos. As simulações foram feitas para o algoritmo MMAS descrito no Capitulo 3 e para o algoritmo FTS-MMAS cuja modificações em relação ao MMAS são descritas no Capítulo 4. O intuito é comparar o desempenho dos dois algoritmos quanto a tempo de execução e comprimento do caminho obtido.

O mapa utilizado como base desta simulação consta de um simples obstáculo retangular no centro de uma área livre, conforme Figura 15. Nesta figura os pixels em cinza inferior e superior representam, respectivamente, os pontos de partida e de chegada adotados. O grafo formado a partir deste mapa, apresenta ao todo 312 vértices, que representa um mapa de baixa resolução, porém é suficiente para se verificar o objetivo deste cenário. A distância ótima para este cenário é de 26 unidades.

Foram realizadas ao todo 1000 execuções de ambos os algoritmos. Para o algoritmo MMAS adotou-se dois critérios de parada. O primeiro foi a adoção de um número máximo de iterações. O segundo critério adotado consta de verificar se a variância do comprimento dos caminhos formados é menor do que um limite adotado. O algoritmo é encerrado assim que um desses critérios é satisfeito. Para o FTS-MMAS o algoritmo é executado em somente uma

Figura 15 – Representação do mapa do Cenário A.



Fonte: Autoria própria.

iteração.

Os parâmetros utilizados para ambos os algoritmos MMAS e FTS-MMAS são apresentados na Tabela 1. Os valores adotados para os fatores de evaporação e heurísticos do algoritmo MMAS foram escolhidos utilizando-se os valores recomendados na literatura (PELLEGRINI *et al.*, 2006). Para o FTS-MMAS, só se utiliza uma formiga e uma iteração, e os outros parâmetros não são inicializados, como afirmado no Capítulo 4.

Tabela 1 – Parâmetros utilizados para o algoritmo MMAS e FTS-MMAS.

Parâmetro	Algoritmo MMAS	Algoritmo FTS-MMAS
Número de formigas	10	1
Número máximo de iterações	100	1
Variância mínima	0,001	–
Fator de evaporação ρ	0,98	–
Fator heurístico α	2,5	–
Fator heurístico β	1	–

Fonte: Autoria própria.

Na Tabela 2 são apresentados os resultados para a execução dos algoritmos. Como esperado, o algoritmo FTS-MMAS apresenta desempenho superior tanto em comprimento de caminho como de tempo de execução. O comprimento médio é exatamente igual ao comprimento ótimo e o tempo de execução de aproximadamente um centésimo de segundo são satisfatórios.

Tabela 2 – Resultados do teste para o Cenário A.

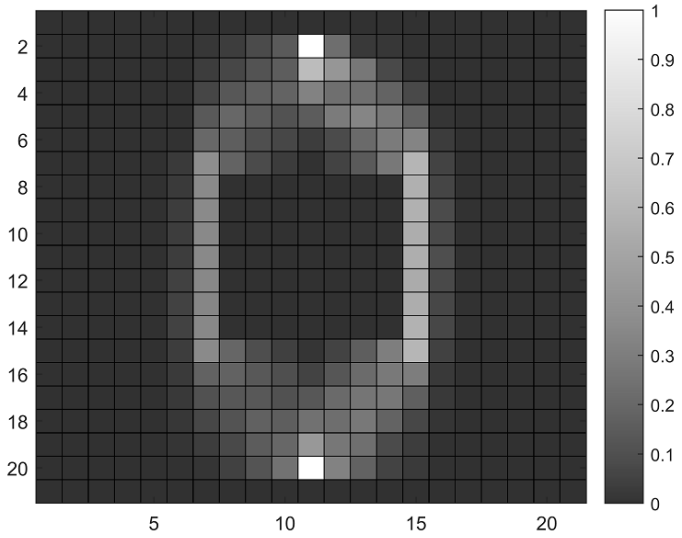
Parâmetro	Algoritmo MMAS	Algoritmo FTS-MMAS
Comprimento médio	26,3340 u	26 u
Tempo médio de execução	1,0800 s	0,0994 s

Fonte: Autoria própria.

Na Figura 16 é apresentado um mapa de calor dos melhores caminhos obtidos em

cada execução do algoritmo MMAS. Os pontos de partida e de chegada, por necessariamente fazerem parte de todos os caminhos apresentam valor máximo associado.

Figura 16 – Mapa de calor da distribuição no espaço dos melhores caminhos formados em cada execução para o mapa do Cenário A para o algoritmo MMAS.



Fonte: Autoria própria.

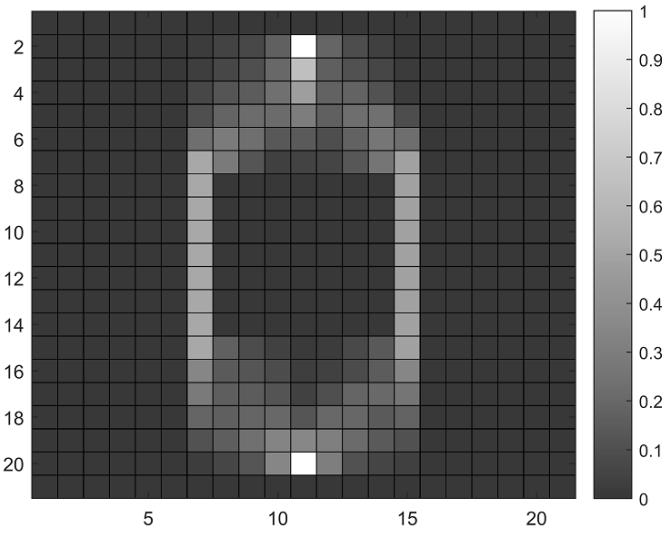
Da mesma figura, pode-se notar uma certa diversidade de caminhos formados, identificada principalmente pelas regiões logo acima do ponto de partida e abaixo do ponto de chegada. Isso porque nessas duas regiões as formigas possuem um número maior de opções pelas quais se deslocar. O mesmo já não ocorre nas laterais do obstáculo, uma vez que contornar o obstáculo mantendo-se junto a este resulta em um caminho de menor comprimento, fato este comprovado pela intensidade maior no mapa de calor presente nessas regiões.

Na Figura 17 apresenta-se o mapa de calor dos melhores caminhos obtidos para o algoritmo FTS-MMAS. De modo semelhante ao mapa de calor da Figura 16, mesmo todos os caminhos sendo ótimos, ainda existe uma alta diversidade nos caminhos formados. Porém, diferentemente do mapa de calor da Figura 16, pode-se observar que não há deslocamentos a mais do que necessário no eixo horizontal, refletindo a melhor qualidade dos caminhos construídos em relação ao comprimento destes.

5.1.2 Cenário B

O Cenário B utiliza-se de dois mapas de baixa resolução semelhantes ao mapa do Cenário A. Porém, estes mapas apresentam a característica de possuírem mínimos locais. O objetivo então é verificar se os algoritmos são capazes de escapar destes mínimos locais e

Figura 17 – Mapa de calor da distribuição no espaço dos melhores caminhos formados em cada execução para o mapa do Cenário A para o algoritmo FTS-MMAS.

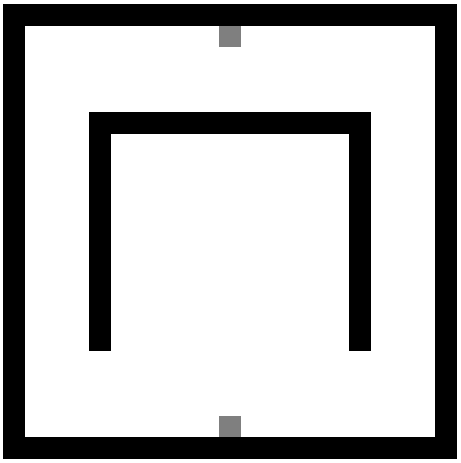


Fonte: Autoria própria.

construir uma solução com o mínimo de impacto no desempenho do algoritmo. Assim como para o Cenário A, os testes foram feitos utilizando-se as duas versões do algoritmo. Os parâmetros utilizados para ambos os algoritmos são os mesmos apresentados na Tabela 1. Cada algoritmo foi executado 200 vezes.

O primeiro mapa apresenta um obstáculo simples em forma de "U"(Figura 18) que funciona como um mínimo local para algoritmos que tentam construir caminhos através de seu lado côncavo. Assim como para o mapa do Cenário A, os pixels em cinza inferior e superior representam, respectivamente, os pontos de partida e de chegada adotados. O grafo formado apresenta ao todo 328 vértices. A distância ótima para este cenário é de 32 unidades.

Figura 18 – Representação do primeiro mapa do Cenário B.



Fonte: Autoria própria.

Na Tabela 3 são apresentados os resultados para execução dos algoritmos para o

primeiro mapa do Cenário B. Em relação ao comprimento médio obtido pelos algoritmos, tem-se uma situação semelhante àquela apresentada no Cenário A, com o algoritmo FTS-MMAS tendo um desempenho melhor em relação ao algoritmo MMAS para os parâmetros avaliados.

Tabela 3 – Resultados do teste para o primeiro mapa do Cenário B.

Parâmetro	Algoritmo MMAS	Algoritmo FTS-MMAS
Comprimento médio	32,3700 u	32 u
Tempo médio de execução	2,3918 s	0,0948 s

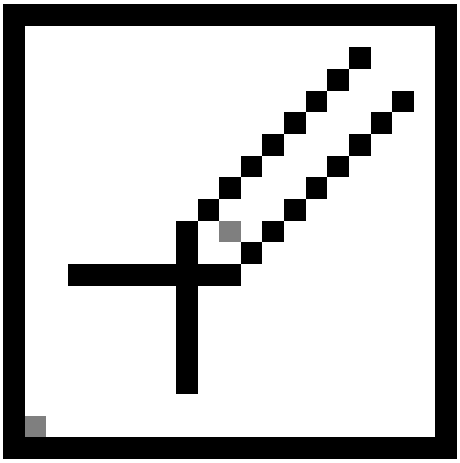
Fonte: Autoria própria.

Porém, pode-se notar uma piora no tempo médio de execução para o algoritmo MMAS. Isto indica que, apesar de as formigas conseguirem escapar do mínimo local durante a construção do caminho, a existência deste por si só apresenta um impacto negativo na velocidade em que as formigas encontram a solução.

Enquanto isto, para o algoritmo FTS-MMAS não há mudanças significativas no tempo de execução. Fato este que também é esperado, pois ambos, mapa e caminhos, apresentam tamanhos semelhantes ao caso do Cenário A. Sendo que são estes os principais limitantes em relação à velocidade de execução do algoritmo.

O segundo mapa apresenta um obstáculo em forma de tesoura (Figura 19), possuindo vários pontos de mínimos locais, sendo um mapa mais complexo em comparação ao primeiro. Para este mapa, o pixel no canto inferior esquerdo representa o local de partida e o pixel localizado no centro do mapa representa o local de chegada. O grafo formado apresenta ao todo 330 vértices. A distância ótima para este cenário é de 50 unidades.

Figura 19 – Representação do segundo mapa do Cenário B.



Fonte: Autoria própria.

Na Tabela 4 são apresentados os resultados para execução dos algoritmos para o

segundo mapa do Cenário B. Neste caso, ambos os algoritmos obtiveram desempenho ótimo em relação ao comprimento dos caminhos.

Tabela 4 – Resultados do teste para o segundo mapa do Cenário B.

Parâmetro	Algoritmo MMAS	Algoritmo FTS-MMAS
Comprimento médio	50 u	50 u
Tempo médio de execução	18,1141 s	0,1355 s

Fonte: Autoria própria.

Porém, as principais diferenças são notadas no tempo médio de execução dos algoritmos. Para o algoritmo MMAS, mesmo com os mapas apresentando dimensões semelhantes e o caminho ótimo apresentando comprimento pouco maior, o impacto da utilização de um mapa mais complexo, apresenta um impacto grande no tempo de execução, reduzindo a efetividade do mesmo e tornando-o ineficiente em relação à velocidade de execução para mapas complexos. Por outro lado, o algoritmo FTS-MMAS pouco é afetado pela modificação do mapa, mantendo sua consistência e justificando sua utilização.

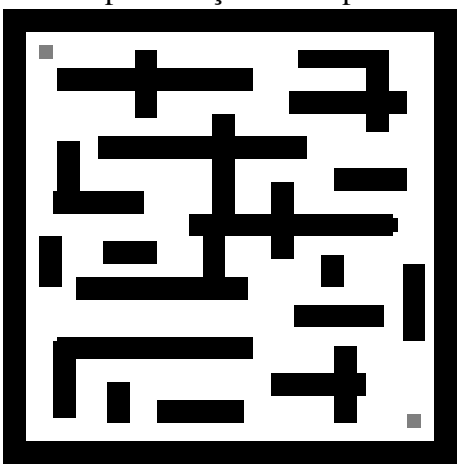
5.1.3 Cenário C

No Cenário C tem-se por objetivo testar o desempenho do algoritmo FTS-MMAS para um mapa de alta dimensão com diversos obstáculos. O mapa utilizado para este cenário é apresentado na Figura 20. O ponto de partida para este mapa é localizado no canto superior esquerdo e o ponto de chegada é localizado no canto inferior direito. O grafo formado apresenta ao todo 5583 vértices. A distância ótima para este cenário é de 160 unidades.

Neste cenário, somente o algoritmo FTS-MMAS foi utilizado para se realizar o teste. O algoritmo foi executado 200 vezes e os parâmetros utilizados para o FTS-MMAS foram os mesmos apresentados na Tabela 1.

Na Tabela 5 são apresentados os resultados do teste do algoritmo FTS-MMAS para o Cenário C. Como esperado o algoritmo obtém sempre o caminhos de distância ótima. Porém, percebe-se para o FTS-MMAS que para um mapa de maior dimensão o tempo de execução passa a ser muito maior do que para mapas mais simples. Isso é justificado pelo fato de que o processo de atribuição de níveis às arestas é dependente do tamanho do grafo, logo, quanto maior o grafo, maior a quantidade de vértices entre o ponto de partida e o ponto de chegada, assim mais lento será o processo de atribuição de níveis. Para todas as execuções que foram feitas do algoritmo, em média, cerca de 96,29% do tempo foi utilizado para realizar o processo de atribuição de níveis.

Figura 20 – Representação do mapa do Cenário C.



Fonte: Autoria própria.

Tabela 5 – Resultados do teste para o Cenário C.

Parâmetro	Algoritmo FTS-MMAS
Comprimento médio	160 u
Tempo médio de execução	35,3229 s

Fonte: Autoria própria.

Na Figura 21 apresenta-se o mapa de calor dos melhores caminhos obtidos. Pode-se notar mais claramente que a variação apresentada nos caminhos aparece basicamente nos pontos em que há uma curva no caminho. O que era de se esperar, uma vez que devido ao tipo de vizinhança adotada, as formigas não fazem curvas na diagonal, somente em ângulos retos.

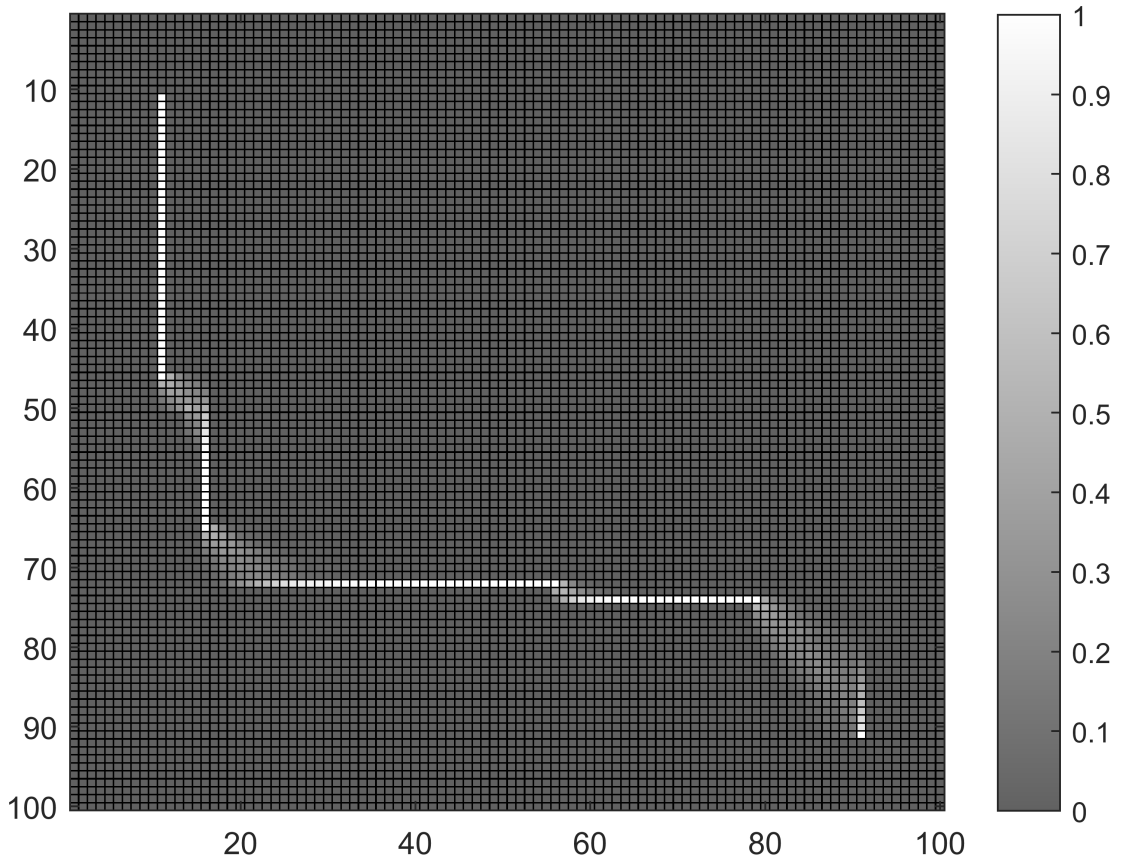
5.2 Experimento no robô real

O principal objetivo deste trabalho é a realização de um experimento de planejamento de caminho para o robô real apresentado na Seção 2.2. Os resultados deste experimento visam atestar se o caminho planejado através da estratégia de planejamento adotada é suficientemente adequado para que o robô possa executá-lo com o máximo de fidelidade possível.

É importante lembrar que a estratégia adotada consta da utilização de uma técnica de decomposição em células do espaço de trabalho aliada ao uso de um algoritmo ACO destinado a encontrar um caminho livre de obstáculos e de comprimento mínimo. E que durante a utilização desta estratégia, as restrições não-holonômicas apresentadas pelo robô não foram consideradas, restrições essas existentes devido ao fato de o robô apresentar tração diferencial (SIEGWART *et al.*, 2011).

Devido a este fato, espera-se que o robô tenha dificuldades em seguir o caminho formado, mesmo este apresentando um controlador destinado a realizar o seguimento da trajetória

Figura 21 – Mapa de calor da distribuição no espaço dos melhores caminhos formados em cada execução para o mapa do Cenário C para o algoritmo FTS-MMAS.



Fonte: Autoria própria.

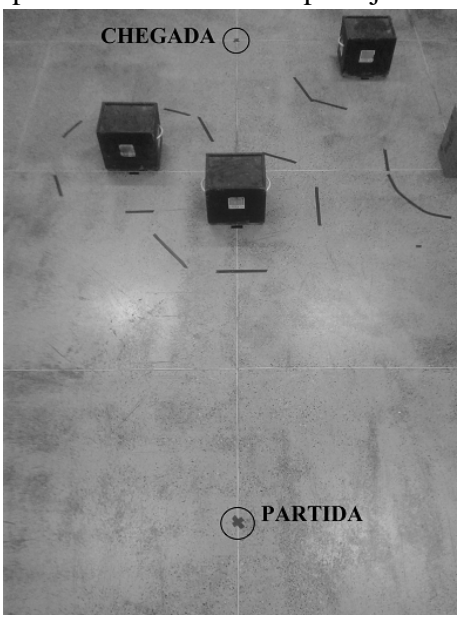
(FORTE *et al.*, 2018). Isto porque o caminho formado apresenta somente curvas ortogonais, que não são de fácil realização para o robô, visto que este apresenta tração diferencial, que implica em restrições não-holonômicas.

Visto isto, a fim de realizar o experimento com o robô real, o cenário apresentado na Figura 22 foi montado. Este cenário apresenta configuração de obstáculos simples, porém suficiente para verificar o objetivo, que é atestar se o robô consegue planejar e executar o caminho planejado. Os pontos de partida e chegada adotados são indicados nesta figura.

O espaço de trabalho é mapeado de acordo com o processo descrito na Seção 2.4, onde se utiliza um algoritmo de SLAM para gerar o mapa apresentado na Figura 23. O mapa gerado apresenta dimensão de 128x128 células que possuem dimensão de 5 centímetros.

Após isto, esse mapa é processado para que se possa gerar o grafo da Figura 24, conforme descrito na Seção 2.3. Neste grafo, as áreas mais claras representam o espaço livre pelo qual garante-se que o robô não colidirá com os obstáculos. Durante o processo de criação do grafo, especificamente na tarefa de realizar o crescimento dos obstáculos, cada obstáculo foi aumentado em 40 centímetros em todas as dimensões, valor este determinado tomando-se

Figura 22 – Cenário montado para realizar o teste de planejamento de caminho para o robô real.



Fonte: Autoria própria.

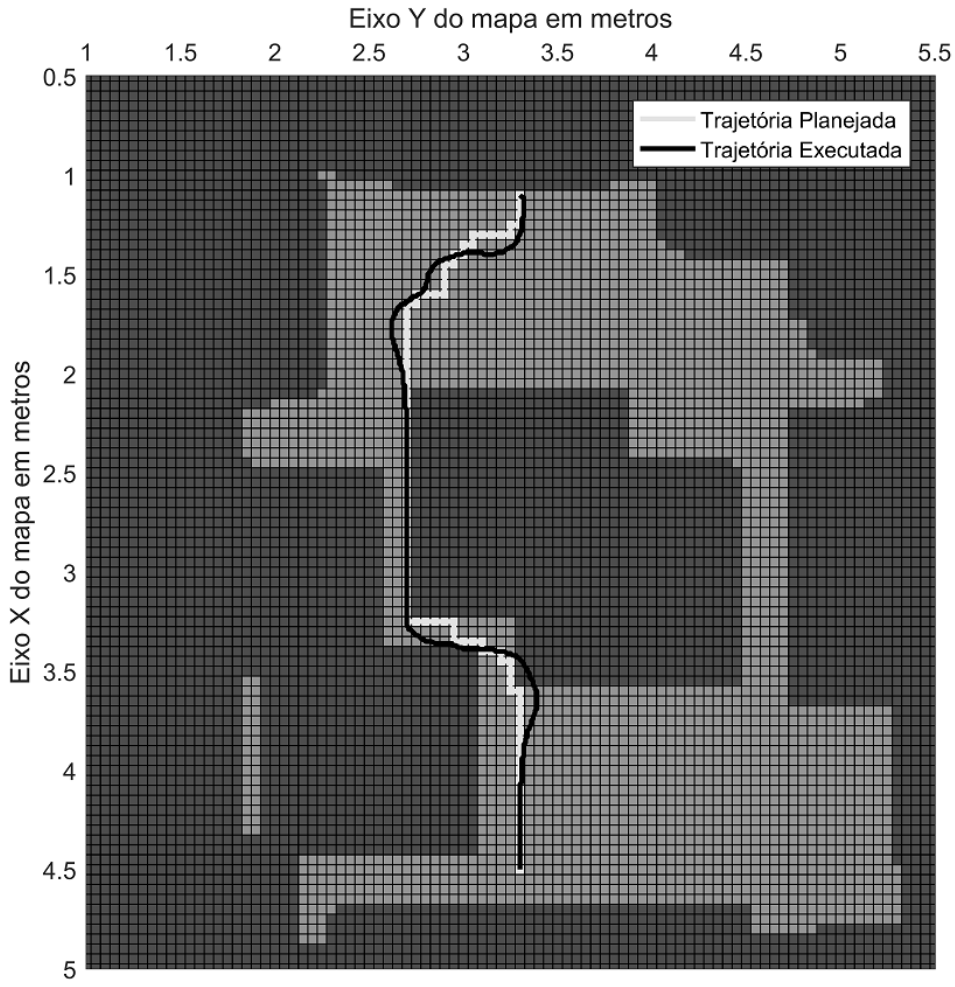
Figura 23 – Mapa gerado para o ambiente montado na Figura 22.

Fonte: Autoria própria.

como base o raio do robô de aproximadamente 30 centímetros. Fez-se isso para garantir-se uma margem de segurança de 10 centímetros para o robô, caso este não seja capaz de executar o caminho planejado com total precisão. E por fim, de posse do grafo que representa o espaço de trabalho, executa-se a versão FTS-MMAS do algoritmo de ACO para se encontrar um caminho, representado pela curva em branco na Figura 24.

Com o caminho gerado, realizou-se o teste de seguimento da trajetória. Primeiramente, com base no caminho gerou-se uma trajetória para ser utilizada como referência para o controlador. A trajetória gerada consta basicamente das posições nos eixos x e y e o ângulo de orientação do robô fornecidas de modo a se adequar a frequência de atualização de referência do controlador de 10 ciclos por segundo. Considerou-se uma velocidade linear constante de 10

Figura 24 – Grafo gerado para o mapa com um caminho formado entre os pontos de partida e chegada.



Fonte: Autoria própria.

centímetros por segundo. O método simples de construção dessa trajetória foi adotado devido a não implementação de uma estratégia de planejamento de trajetória junto a estratégia de planejamento de caminhos.

A trajetória percorrida pelo robô também é apresentada na Figura 24 pela curva em preto. Nesta, é possível observar que o robô foi capaz de seguir o caminho com um certo grau de dificuldade, evidenciado pelas diferenças entre as duas curvas. Nota-se ainda que o robô durante seu deslocamento percorreu parte da área marcada como ocupada por obstáculos, porém não houve de fato colisão, visto a utilização de uma margem de segurança de 10 centímetros no processo de aumento dos obstáculos. Já era esperado que o robô não conseguiria seguir plenamente a trajetória fornecida, principalmente nos locais em que existem curvas, que é onde apresentam-se os maiores desvios em relação à trajetória esperada, como é possível ver na Figura 24.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi estudada e implementada uma estratégia de planejamento de caminhos baseada em decomposição do espaço de trabalho em células aliada à utilização de algoritmos ACO responsáveis por realizar a busca por um caminho no espaço de trabalho que permita a um robô real se descolar seguramente em um ambiente com presença de obstáculos.

O método de decomposição em células consiste de um método clássico utilizado para representar o espaço de trabalho através da criação de um mapa. Em posse deste mapa, para se realizar o planejamento é suficiente fazer uma busca por um caminho que atenda aos critérios desejados da melhor forma possível. Neste trabalho, o único objetivo para o algoritmo de otimização adotado era a obtenção de caminhos de menor comprimento possível.

Para realizar esta busca, dentre as diversas estratégias existentes, optou-se pela utilização de algoritmos baseados em ACO. Foram implementados um algoritmo clássico desta meta-heurística, MMAS - Max-Min Ant System, e uma versão modificada, em que utiliza-se uma heurística mais adequada à situação presente, FTS-MMAS - Fast Two-Stages Max-Min Ant System.

Dito isto, a fim de avaliar as duas versões do algoritmo ACO, testes simulados foram realizados com o objetivo de verificar o desempenho de ambos os algoritmos em relação ao tempo de execução e comprimento do caminho gerado. Para isto, três cenários diferentes foram criados, o primeiro, de estrutura simples, teve por objetivo verificar se os algoritmos eram capazes de encontrar um caminho. O segundo cenário tinha por principal objetivo verificar se os algoritmos eram capazes de escapar de mínimos locais sem apresentar grande perda de desempenho. Em ambos os casos, os dois algoritmos foram capazes de encontrar caminhos, porém o algoritmo FTS-MMAS mostrou-se bastante superior em relação ao MMAS, principalmente no segundo cenário, em que havia presença de mínimos locais, como já era esperado.

Com a confirmação do algoritmo FTS-MMAS como sendo aquele que apresenta menor tempo de execução para os dois primeiros cenário apresentados, submeteu-se somente este algoritmo a um teste em um terceiro cenário simulado. Este cenário, em relação aos anteriores, apresenta maior número de células, e o objetivo foi testar se o algoritmo FTS-MMAS teria bom desempenho neste cenário. Em relação ao comprimento de caminho, o desempenho foi ótimo, como já esperado, devido a própria característica do algoritmo discutida no Capitulo 4. Porém, o tempo de execução médio próximo de 35 segundos foi considerado elevado para uma aplicação de robótica, o que qualifica este resultado como não satisfatório.

Por fim, realizou-se um experimento no robô real. O objetivo deste experimento é verificar se o robô dotado de um controlador de seguimento de trajetórias seria capaz de executar com o máximo de fidelidade o caminho planejado. Uma vez que durante o processo de planejamento as restrições mecânicas do robô não são um fator levado em consideração e já que não se realiza um planejamento de trajetória em cima do caminho planejado, era de se esperar que o robô tivesse dificuldades em seguir o caminho planejado sem nenhum erro. Isto de fato se verificou, o robô real apresentou dificuldades em seguir o caminho planejado, porém, o resultado indicou que os erros não foram muito significativos a ponto de impedir completamente que o robô completasse a trajetória.

Posto isto, destacou-se alguns pontos para possível análise em trabalhos futuros relacionados a este trabalho. Primeiramente, o número de estratégias destinadas a solucionar o problema de planejamento de caminhos para robôs móveis na literatura é grande, de modo que é sempre interessante haver uma comparação entre as diversas estratégias, a fim de avaliar o que cada uma tem de bom e com isso se procurar melhores soluções para o problema.

Outro tema interessante a ser tratado, seria a inclusão das restrições mecânicas do robô no processo de planejamento, a fim de garantir que o caminho planejado seja executado com o máximo de precisão pelo robô.

Neste trabalho, a trajetória para o caminho planejado foi elaborada de modo simples. Um outro problema na área da robótica, muito relacionado ao planejamento de caminho que poderia ser abordado seria o planejamento do movimento do robô como um todo, que incluiria, além do planejamento do caminho, o planejamento de como o robô percorreria esse caminho através da determinação de posições, velocidades e acelerações que seriam utilizadas como referência para os controladores.

E por último, neste trabalho, foi utilizado um algoritmo ACO para realização de uma busca em um espaço que objetivava encontrar um caminho de comprimento mínimo, independente de outros parâmetros que poderiam ser de interesse. Muitas vezes quando se busca uma solução ótima, diversos objetivos devem ser levados em consideração. Por isso, uma busca por caminhos que sejam ótimos em relação a múltiplos objetivos também se destacam como um tema de interesse.

REFERÊNCIAS

AL-SHIHABI, S. T.; ALDURGAM, M. M. A max-min ant system for the finance-based scheduling problem. **Computers & Industrial Engineering**, v. 110, p. 264 – 276, 2017. ISSN 0360-8352.

CHEN, X.; KONG, Y.; FANG, X.; WU, Q. A fast two-stage aco algorithm for robotic path planning. **Neural Computing and Applications**, v. 22, n. 2, p. 313–319, Feb 2013.

CHOSET, H.; LYNCH, L.; HUTCHINSON, S.; KANTOR, G.; BURGARD, W.; KAVRAKI, L.; THRUN, S. **Principles of Robot Motion: Theory, Algorithms, and Implementations**. 1st. ed. [S.l.]: The MIT Press, 2005. (Intelligent Robotics and Autonomous Agents series). ISBN 9780262033275.

CORMEN, T.; LEISERSON, C.; RIVEST, R.; STEIN, C. **Introduction to Algorithms**. 3rd. ed. London, England: The MIT Press, 2009. ISBN 9780262033848.

CRAWFORD, B.; SOTO, R.; JOHNSON, F.; MONFROY, E.; PAREDES, F. A max-min ant system algorithm to solve the software project scheduling problem. **Expert Systems with Applications**, v. 41, n. 15, p. 6634 – 6645, 2014. ISSN 0957-4174.

DORIGO, M. **Optimization, Learning and Natural Algorithms (in Italian)**. Tese (Doutorado) — Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

DORIGO, M.; BLUM, C. Ant colony optimization theory: A survey. **Theoretical Computer Science**, v. 344, n. 2, p. 243 – 278, 2005.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 26, n. 1, p. 29–41, Feb 1996.

DORIGO, M.; STüTZLE, T. **Ant Colony Optimization**. Scituate, MA, USA: Bradford Company, 2004. ISBN 9780262042192.

FORTE, M. D. N.; CORREIA, W. B.; NOGUEIRA, F. G.; TORRICO, B. C. Reference tracking of a nonholonomic mobile robot using sensor fusion techniques and linear control. **IFAC-PapersOnLine**, v. 51, n. 4, p. 364 – 369, 2018. ISSN 2405-8963.

HOKUYO. **Scanning Laser Range Finder URG-04LX-UG01 Specifications**. 2009. Disponível em: <https://www.robotshop.com/media/files/pdf/hokuyo-urg-04lx-ug01-specifications.pdf>. Acesso em: 16 aug. 2018.

HUANG, H. C.; WU, T. F.; HUANG, C. Y. A metaheuristic artificial immune system algorithm for mobile robot navigation. p. 803–806, Aug 2014.

KOHLBRECHER, S.; MEYER, J.; STRYK, O. von; KLINGAUF, U. A flexible and scalable slam system with full 3d motion estimation. November 2011.

KORTE, B.; VYGEN, J. **Combinatorial Optimization: Theory and Algorithms**. Berlin, Heidelberg: Springer, 2008. ISBN 9783540718444.

LI, W.; WANG, G.-Y. Application of improved pso in mobile robotic path planning. p. 45–48, Oct 2010.

MILLINGTON, I.; FUNGE, J. **Artificial Intelligence for Games**. 2nd. ed. Burlington, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN 9780123747310.

OTTONI, G. **Planejamento de trajetórias para robôs móveis**. 2000. Monografia (Bacharel em Engenharia da Computação), FURG (Universidade Federal do Rio Grande), Rio Grande, Brasil.

PELLEGRINI, P.; FAVARETTO, D.; MORETTI, E. On max-min ant system's parameters. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 203–214, 2006.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3th. ed. [S.l.]: Pearson, 2010. ISBN 9780133001983.

SABRI, A. N.; RADZI, N. H. M.; SAMAH, A. A. A study on bee algorithm and a* algorithm for pathfinding in games. p. 224–229, April 2018.

SICILIANO, B.; SCIAVICCO, L.; VILLANI, L.; ORIOLO, G. **Robotics: Modelling, Planning and Control**. 1st. ed. [S.l.]: Springer-Verlag London, 2009. (Advanced Textbooks in Control and Signal Processing). ISBN 9781846286421.

SIEGWART, R.; NOURBAKHSH, I.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots**. 2nd. ed. [S.l.]: The MIT Press, 2011. (Intelligent Robotics and Autonomous Agents series). ISBN 9780262015356.

SÖRENSEN, K.; SEVAUX, M.; GLOVER, F. A history of metaheuristics. **CoRR**, abs/1704.00853, 2017. Disponível em: <http://arxiv.org/abs/1704.00853>.

SOUZA, S. **Planejamento de Trajetória para um Robô Móvel com duas Rodas Utilizando um Algoritmo A-Estrela Modificado**. 2008. Dissertação (Mestrado em Engenharia da Elétrica), UFRJ/COPPE (Universidade Federal do Rio de Janeiro), Rio de Janeiro, Brasil.

SPONG, M.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot Modeling and Control**. [S.l.]: Wiley, 2005. ISBN 9780471649908.

STüTZLE, T.; HOOS, H. H. Max–min ant system. **Future Generation Computer Systems**, v. 16, n. 8, p. 889 – 914, 2000.