UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

GUILHERME SONAGLIO CHAVES

Analisando Soluções de Conflitos em Gestores Distribuídos

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof.ª Dra. Mariana Luderitz Kolberg
Co-orientador: Prof. Dr. Renan de Queiroz Maffei

Porto Alegre
2024

“All effective innovations are breathtakingly simple. Indeed, the greatest praise an innovation can receive is for people to say: ‘This is obvious. Why didn’t I think of it?’”

— PETER DRUCKER

RESUMO

Um gestor de frotas de robôs é uma ferramenta que auxilia na distribuição de tarefas para um conjunto de robôs em um determinado ambiente com o objetivo de realizar essas tarefas no menor tempo possível. Já existem boas ferramentas de gestão de frotas no mercado que funcionam com poucas centenas de robôs. À medida que o tamanho dessas frotas aumenta, tornam-se necessários métodos melhores para a organização dos robôs. Neste trabalho, foram analisados alguns algoritmos e arquiteturas utilizados na gestão de frotas, e foi realizado um estudo aprofundado de gestores de arquiteturas distribuídas, a fim de tentar resolver o problema que ocorre quando dois robôs, andando em sentidos opostos, se encontram em um corredor longo e estreito. Em cima desse estudo, foi implementado, no sistema supervisor de uma arquitetura totalmente distribuída, um algoritmo cotrolador de tráfego para ser acionado somente quando um robô entrar em uma área crítica. Esse algoritmo diminui consideravelmente a quantidade de conflitos e deadlocks que ocorrem nas arquiteturas distribuídas, tornando elas mais confiáveis de serem usadas na indústria.

Palavras-chave: Gestor de frotas de robôs. Arquitetura distribuída. Planejador de caminhos. Corredores estreitos.

Analyzing Conflict Solutions in Distributed Managers

ABSTRACT

A robot fleet manager is a tool that helps to distribute tasks to a set of robots in a given environment with the aim of carrying out these tasks in the shortest possible time. There are already good fleet management tools on the market that work with just a few hundred robots. As the size of these fleets increases, better methods for organizing robots become necessary. In this work, some algorithms and architectures used in fleet management were analyzed, and an in-depth study of distributed architecture managers was carried out, in order to try to solve the problem that occurs when two robots, walking in opposite directions, meet in a long and narrow corridor. Based on this study, a traffic control algorithm was implemented in the supervisor system of a fully distributed architecture to be activated only when a robot enters a critical area. This algorithm considerably reduces the number of conflicts and deadlocks that occur in distributed architectures, making them more reliable to be used in industry.

Keywords: Robot fleet manager. Distributed architectures. Path planner. Narrow corridors.

LISTA DE ABREVIATURAS E SIGLAS

AMCL	Localização de Monte Carlo Adaptativa
IAs	Inteligências Artificiais
IoTs	Internet of Things
ROS	Robot Operating System
SLAM	Localização e Mapeamento Simultâneos

LISTA DE FIGURAS

LISTA DE TABELAS

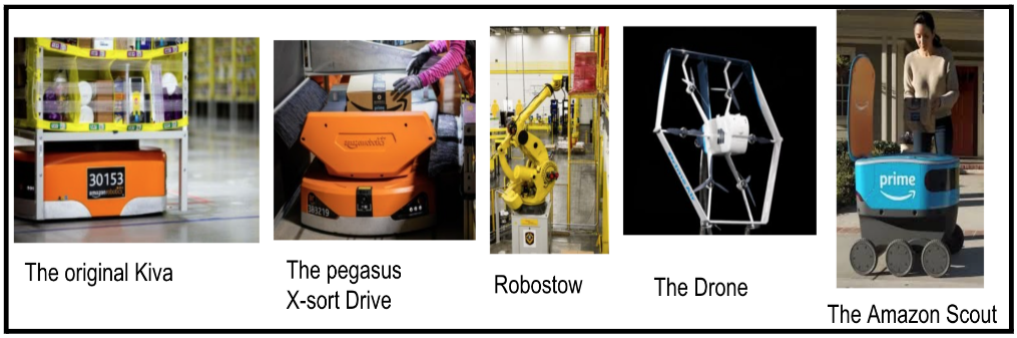
SUMÁRIO

1 INTRODUÇÃO

A indústria tem um papel muito importante para a nossa sociedade, sendo utilizada para a criação de produtos e bens de consumo. Já ocorreram três revoluções industriais até o momento, e à medida que a população cresce, as necessidades de produção tendem a aumentar. Para suprir essa demanda, nós precisamos nos adaptar ao uso de novas tecnologias, o que nos leva a uma nova revolução industrial, chamada de indústria 4.0 (Bicaku et al., 2018).

A indústria 4.0 tem o propósito de criar um ecossistema industrial totalmente digital, incorporando várias tecnologias que se comunicam entre si. Essa indústria utiliza tecnologias que vão desde Internet das Coisas (IoTs) e Inteligências Artificiais (IAs), até o uso de robôs, que podem ser fixos ou móveis (Colombo; Bangemann; Karnouskos, 2014). Esses robôs móveis são utilizados para transportar produtos dentro do ambiente de galpões logísticos, sendo esses galpões locais que armazenam produtos que aguardam uma nova fase no processo de comercialização. (Girija Patil; Mareena; Kaewkhiaolueang, 2021) comenta que a Amazon, por exemplo, utiliza o robô Kiva e o robô da linha Pegasus para realizar o transporte de prateleiras dentro dos galpões logísticos, sendo que o Pegasus consegue levantar mais peso e realizar o processamento das informações das tarefas de maneira mais rápida. Além disso, drones e outros tipos de robôs também podem ser utilizados para realizar entregas diretamente para o cliente. Alguns dos tipos de robôs utilizados pela Amazon podem ser visualizados na Figura 1.1.

Figura 1.1 – Linha de robôs utilizada pela Amazon.



Fonte: (Girija Patil; Mareena; Kaewkhiaolueang, 2021)

Dentro dos galpões logísticos, os robôs móveis precisam ser capazes de se mover de maneira autônoma em um ambiente que muda dinamicamente. Para isso, esses robôs precisam ser supervisionados por um sistema externo capaz de garantir a segurança deles e, consequentemente, do ambiente em que eles estão operando (Hazik et al., 2022). Esse

sistema supervisor faz parte da arquitetura de um gestor de frotas de robôs e tem como principal objetivo, comum a todas as arquiteturas, distribuir um conjunto de tarefas para essa frota. Essas tarefas devem ser distribuídas de tal forma que elas sejam finalizadas no menor tempo possível.

Dos problemas fundamentais trabalhados na robótica móvel (Figura 1.2), o mais importante para um gestor de frotas é o planejamento de movimento. Nesse problema, o sistema encarregado do controle do robô deve realizar um planejamento de rotas de acordo com o objetivo fornecido para o robô em particular. O planejamento tem o propósito de gerar o menor caminho livre de obstáculos possível entre o ponto em que o robô se encontra e o objetivo da tarefa (Bruno; Khatib, 2016). Para tanto, o coordenador já deve possuir o mapa do ambiente de antemão.

Figura 1.2 – Problemas fundamentais da Robótica Móvel

Fonte: Adaptada de (Makarenko et al., 2002)

Existem basicamente dois tipos principais de algoritmos e arquiteturas para gestores de frotas de robôs: distribuídos e centralizados. No caso dos gestores distribuídos, cada robô fica responsável pelo seu próprio controle e planejamento de caminhos. Já nos gestores centralizados, o sistema supervisor externo tem como objetivos cuidar do planejamento da frota inteira de robôs no ambiente e controlar os movimentos de cada um dos robôs. Atualmente, os gestores centralizados tem preferência sobre os distribuídos para o uso na indústria, uma vez que eles apresentam maior segurança e previsibilidade durante sua execução (Palleschi et al., 2020).

Essa nova indústria, por questões de segurança e desempenho, possui alguns requisitos com relação ao uso de robôs autônomos dentro de galpões logísticos. Um dos requisitos mais importantes é com relação ao tempo de comunicação entre os agentes envolvidos no controle dos robôs. No caso dos robôs estarem sendo controlados por um gestor de frotas centralizado, o delay presente entre as trocas de mensagens, se muito grande, pode gerar uma dessincronização entre os robôs da frota. Esse atraso no tempo

de comunicação entre os sistemas envolvidos em um gestor centralizado se torna um problema cada vez mais preocupante à medida que o tamanho da frota tende a se tornar maior (Hazik et al., 2022).

(Bertilsson et al., 2022) ao fazer uma comparação entre uma arquitetura de gestor centralizado e um distribuído, usando um modelo de controle preditivo não linear de planejamento de trajetórias, chegou a conclusão que o tempo que leva para um gestor centralizado completar as tarefas da frota tende a escalar quadraticamente com relação à quantidade de robôs, enquanto o tempo, para um gestor distribuído, aumenta linearmente com o tamanho da frota (Tabela 1.1). Levando-se em consideração que a quantidade de robôs utilizada em galpões logísticos, como os da Amazon, chegam à casa das centenas, e que essa quantidade pode vir a aumentar com o passar dos anos, o uso de arquiteturas distribuídas nesses ambientes se torna cada vez mais necessário.

Tabela 1.1 – Tempo de execução de tarefas em diferentes condições, para arquiteturas centralizadas e distribuídas.

Case	**Centralized [ms]**	**Distributed [ms]**
2 robots, collision	3.6	5.6
2 robots, crossing (case 1)	3.6	3.0
2 robots, dynamic object	3.5	5.0
5 robots, joining the path (case 2)	6.7	3.0
5 robots, crossing	7.2	3.2
5 robots, dynamic object	6.7	3.6
10 robots, crossing	21.9	3.7
10 robots, dynamic object (case 3)	20.5	3.6

Fonte: (Bertilsson et al., 2022)

As arquiteturas distribuídas, por sua vez, apresentam alguns desafios para uso em galpões logísticos. O principal desafio refere-se ao fato de que elas podem não funcionar no caso em que dois robôs estejam andando em lados opostos de um corredor estreito e longo (Palleschi et al., 2020), algo muito comum em galpões logísticos. Esse caso causará colisão ou parada dos robôs naquele local, uma vez que não existe espaço para manobra entre eles.

Logo, nesse trabalho de conclusão de curso, será efetuado um estudo sobre algoritmos e arquiteturas utilizadas em frotas de robôs atualmente, realizando, em cima disso, um estudo aprofundado sobre diferentes técnicas existentes para evitar o encontro de robôs em passagens estreitas com arquiteturas descentralizadas. Além disso, será implementado, utilizando o ROS, um algoritmo de planejamento de caminhos capaz de evitar que esse problema com corredores estreitos ocorra.

Este trabalho está organizado da seguinte forma. No capítulo 2, será realizada uma

revisão da literatura abordando, com maior nível de detalhes, as arquiteturas centralizadas e distribuídas de frotas de robôs. O capítulo 3 buscará solucionar o problema do tráfego de robôs nos corredores estreitos com base em estudos relacionados sobre o assunto. O capítulo 4 conterá dados sobre o ambiente de teste dos experimentos, enquanto o capítulo 5 discutirá os resultados dos testes. Por fim, serão discutidas as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, será realizada uma revisão bibliográfica onde serão discutidos, com maior nível de detalhes, os dois principais tipos de arquiteturas usadas em frotas de robôs na indústria 4.0. Na seção 2.1, será apresentada uma revisão da literatura referente às arquiteturas centralizadas, já na seção 2.2, será feita uma revisão sobre as arquiteturas distribuídas.

2.1 Arquiteturas Centralizadas

Gestores centralizados são aqueles que utilizam um sistema externo supervisor que realiza o planejamento e sincronização de todos os robôs da frota durante a execução deles. Eles são os mais utilizados no ambiente de galpões logísticos, pois trazem maior segurança e previsibilidade durante a execução das tarefas. Além disso, por realizar o planejamento de todos os robôs simultaneamente, é possível conseguir um tempo ótimo na execução de tarefas (Berndt et al., 2021).

Esse tipo de gestor é bastante usado na indústria atualmente, porém ele apresenta alguns problemas que o tornam menos atrativo à medida que o uso de uma quantidade elevada de robôs dentro de galpões logísticos se faz necessário. Pelo fato do sistema supervisor ter que ficar responsável pela sincronização dos robôs, o maior número de robôs acaba aumentando demais a transmissão de mensagens entre o supervisor e eles. Essa quantidade elevada de mensagens pode levar a um engarrafamento ou atraso muito grande nessas comunicações, o que leva a interrupção das atividades dos robôs para evitar possíveis acidentes. É preciso um estudo maior no uso de tecnologias de rede como o Wi-Fi dentro da indústria para tentar mitigar esse problema de comunicação (Hazik et al., 2022).

Além disso, pelo fato do controle dos robôs ser exercido pelo sistema central supervisor, para evitar acidentes, todos os robôs devem interromper suas atividades quando um deles encontra um obstáculo no meio do caminho. Isso serve para evitar problemas de sincronização entre os robôs. Por segurança, as atividades só devem retornar ao normal quando o obstáculo for removido e o usuário ordenar a volta da execução das tarefas ao sistema supervisor (Berndt et al., 2021).

Outro problema na utilização desses coordenadores é que, quando um robô termina a sua tarefa, ele deve ficar parado esperando todos os outros terminarem suas res-

pectivas atividades. Isso acontece pelo fato de que o supervisor realiza o processo de planejamento de todos os robôs simultaneamente. Esse problema pode ser evitado ao parar todos os robôs e iniciar um novo planejamento de caminhos. Porém, realizar essa tarefa de maneira otimizada, em que os robôs não alterem muito a trajetória que eles já estavam seguindo, não é trivial.

(Hazik et al., 2022) focou no desenvolvimento de um algoritmo de gestor de frotas que tivesse compatibilidade com os requisitos esperados pela indústria 4.0. A solução do autor utilizou o A*, um planejador de caminhos simples, e manteve o módulo de sincronização dos robôs separado do módulo do planejador. Essas modificações, de acordo com o autor, reduziram consideravelmente o tempo de comunicação entre o robô e o sistema supervisor. Esse feito é um passo importante para que a solução se enquadre melhor dentro dos requisitos da indústria 4.0. Porém, esse gestor acaba trazendo algumas limitações que poderiam ser evitadas. Ele leva em conta o uso de caminhos unidirecionais e permite caminhos bidirecionais apenas em pontos críticos e curtos para facilitar o trabalho do planejador. O uso de caminhos unidirecionais acaba limitando a trajetória que o robô pode seguir.

(Pecora et al., 2018), (Čáp; Gregoire; Frazzoli, 2016), entre outros trabalhos relacionados a esse tipo de gestor foram encontrados, porém todos possuem os problemas característicos citados anteriormente. A solução proposta pelo presente trabalho evitará a maior parte desses problemas, uma vez que se trata de um sistema mais distribuído.

2.2 Arquiteturas Distribuídas

Controladores distribuídos de frotas de robôs tendem a escalar com maior facilidade com relação à quantidade de robôs suportados (Bertilsson et al., 2022). Essa facilidade no escalamento é devido ao fato de que, diferente dos gestores centralizados, o sistema supervisor não controla os movimentos dos robôs. Cada robô tem total autonomia para decidir o caminho que percorrerá, assim como cada um deve ter seu próprio sistema de desvio de obstáculos. A única função do sistema supervisor é decidir qual tarefa será distribuída para qual robô. Como o coordenador de tarefas não realiza o planejamento de caminhos da frota, não é possível alcançar uma solução ótima para a distribuição das tarefas. Mas é possível conseguir uma solução subótima, com cada robô conseguindo um planejamento local ótimo (Berndt et al., 2021).

Para decidir a qual robô atribuir determinada tarefa, uma série de aspectos devem

ser analisados pelo gestor: o robô selecionado precisa ter bateria suficiente para realizar a tarefa; no caso de uma frota heterogênea, o robô precisa ter características que o tornam capaz de completar a tarefa; o controlador deve escolher o robô capaz de completar a tarefa no menor tempo possível (Rodrigues et al., 2022).

Devido ao fato de que o supervisor não precisa se preocupar com a sincronização entre os robôs, o tráfego de comunicação entre o supervisor e os robôs é demasiadamente reduzido. Os robôs apenas precisam reportar o status da missão periodicamente para o supervisor ter conhecimento de como está indo a execução da tarefa e como está a bateria do robô no momento. Essa diminuição na comunicação permite que a frota possa vir a aumentar, adequando-se aos requisitos da indústria 4.0.

Apesar do sistema distribuído poder parecer melhor do que o centralizado, ele apresenta alguns desafios para uso em galpões logísticos. O principal desafio refere-se ao fato de que ele não funciona no caso em que dois robôs estejam andando em lados opostos de um corredor estreito e longo (Palleschi et al., 2020), algo muito comum em galpões logísticos. Esse caso irá causar colisão ou parada dos robôs naquele local, uma vez que não existe espaço para manobra entre eles. Esse problema ocorre porque os robôs não tem conhecimento sobre o caminho que os outros estão percorrendo, logo, eles entram no corredor mesmo que outro robô já esteja passando por ele naquele momento.

(Bertilsson et al., 2022) fez um estudo comparativo onde ele chega a conclusão de que, enquanto a quantidade de robôs em galpões for compatível com as capacidades de controle de um sistema centralizado, um gestor centralizado deve ser preferível sobre um sistema distribuído, especialmente, devido a essa situação que ocorre com corredores estreitos. Nesse estudo, ele também leva em consideração o fato de que os robôs, em um sistema distribuído, podem demorar mais para chegar aos seus objetivos, principalmente, quando ocorre um desvio de caminho para evitar um obstáculo.

3 O PROBLEMA DOS CORREDORES ESTREITOS

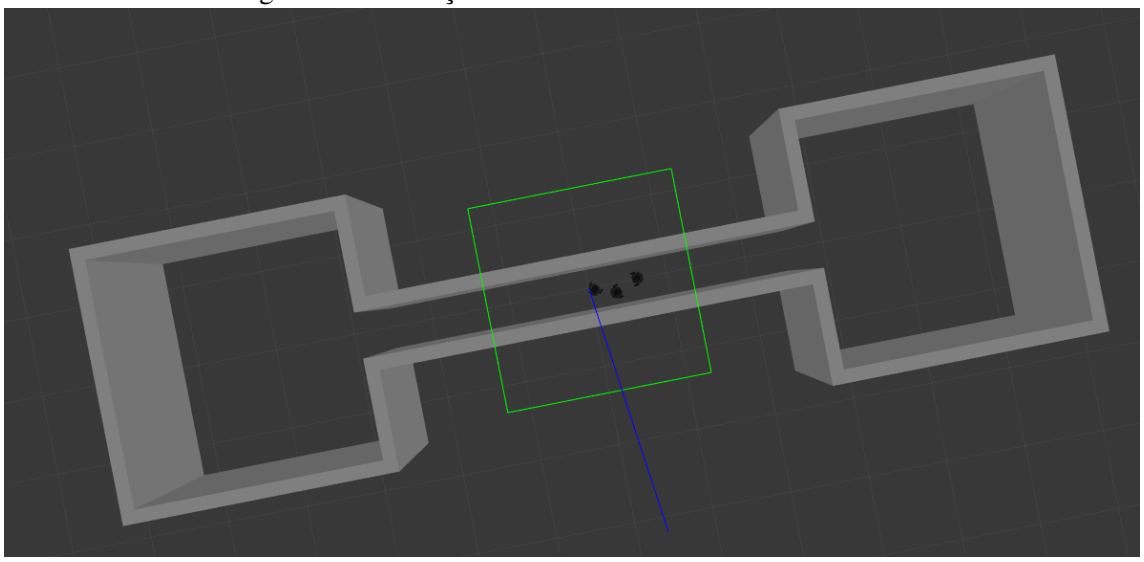
Neste capítulo, será feita uma análise sobre soluções já existentes para o problema que ocorre quando dois robôs entram em conflito ao passarem por um corredor longo e estreito em uma arquitetura distribuída. E, ao final do capítulo, será apresentada uma proposta para evitar que esse problema ocorra.

3.1 Definição do Problema

Quando utilizamos gestores de robôs com arquiteturas distribuídas, podemos nos deparar com diversas ocorrências de conflitos e deadlocks entre robôs, uma vez que, nesse tipo de arquitetura, um robô não tem conhecimento da localização ou do caminho por onde os outros robôs estão passando.

Um exemplo de conflito que pode ocorrer é quando dois robôs, vindos de sentidos opostos, tentam passar ao mesmo tempo por um corredor longo e estreito. Como um não tem conhecimento do outro, eles acabam se encontrando dentro do corredor. Não havendo espaço para manobras, eles se encontram em uma situação de deadlock.

Figura 3.1 – Situação de conflito em um corredor estreito.

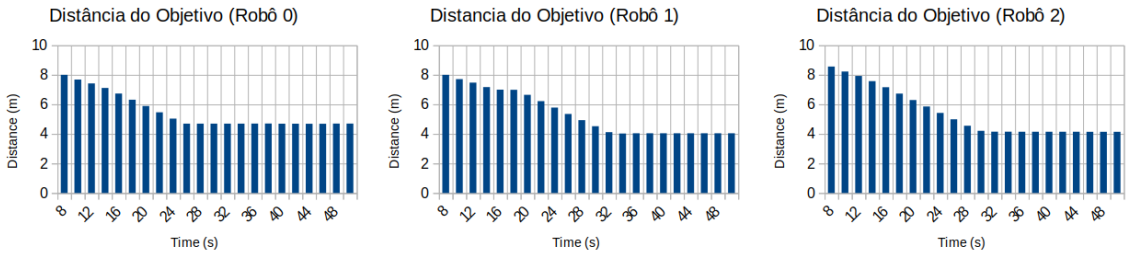


Fonte: De autoria própria.

Foi executada uma simulação que demonstra a situação de conflito citada. Na figura 3.1, é possível ver o resultado final da simulação, e na figura 3.2, temos gráficos mostrando a evolução do caminho percorrido pelos robôs através da distância euclidiana deles até seus respectivos objetivos. É possível ver pelos gráficos, que nenhum robô

conseguiu chegar até seu objetivo. Isso se deve ao fato de ter ocorrido um deadlock que impossibilitou que os robôs avançassem.

Figura 3.2 – Gráficos demonstrando o caminho percorrido pelos robôs durante a execução da simulação.



Fonte: De autoria própria.

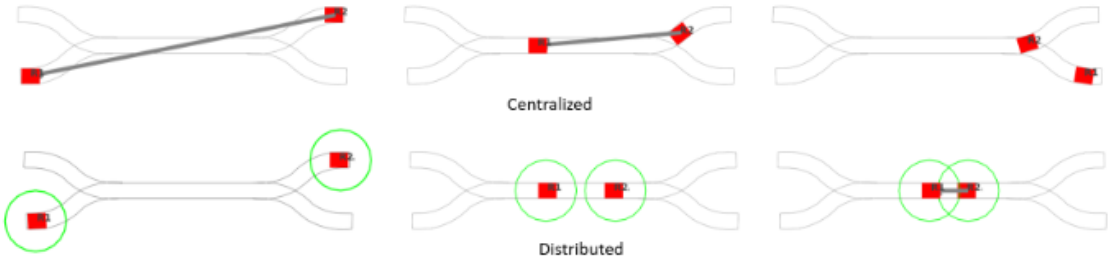
Este trabalho tem o objetivo de estudar algumas soluções existentes hoje em dia que tentam evitar que esses conflitos ocorram. E, levando como base o estudo realizado, foi desenvolvida uma proposta nova de solução de conflitos, que será apresentada nesse capítulo.

3.2 Análise de Soluções de Conflito

(Palleschi et al., 2020) compara três tipos de arquiteturas em diferentes situações. Ele faz a comparação entre uma arquitetura centralizada, uma arquitetura totalmente distribuída e uma outra parcialmente distribuída. A arquitetura totalmente distribuída utilizada em seu trabalho assume que os robôs conseguem se comunicar entre si, mas apenas se um robô estiver dentro do raio de comunicação do outro. Essa arquitetura distribuída acaba causando o erro dos corredores longos, comentado anteriormente. Pensando em resolver esse problema, o autor resolveu desenvolver uma arquitetura parcialmente distribuída. Essa arquitetura funciona de igual maneira a centralizada, porém quando os robôs se aproximam de uma área crítica, como a de um corredor estreito, é permitido que eles se comuniquem entre si para se coordenarem. Isso requer um sistema de comunicação ou coordenador local a parte. Ao realizar o teste em uma área crítica com os robôs vindo de sentidos opostos, ele constatou que sua implementação parcialmente distribuída resolveu o problema dos corredores longos, comportando-se de maneira similar a uma arquitetura totalmente centralizada. A figura 3.3 mostra a execução do teste em que dois robôs, partindo de lados opostos, precisam passar por um corredor longo.

(Azarm; Schmidt, 1997) se utiliza da troca de informações entre robôs com o objetivo de evitar deadlocks e resolver conflitos como o da figura 3.4. Os robôs devem

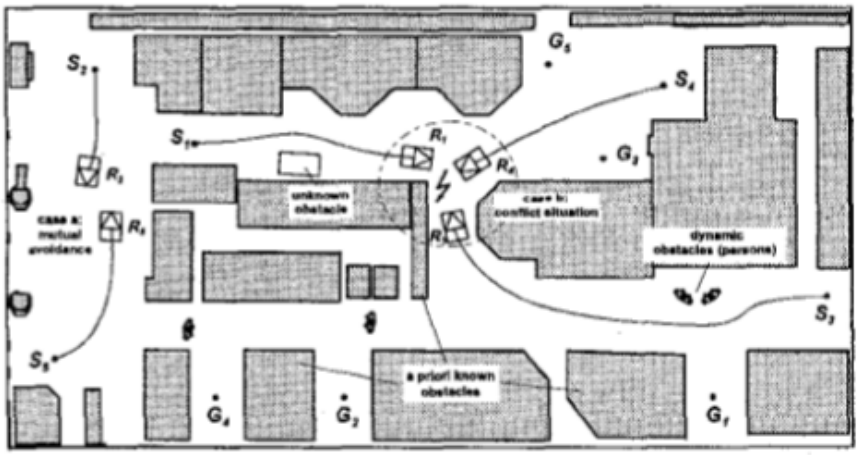
Figura 3.3 – Simulação de arquitetura centralizada e totalmente distribuída, respectivamente. A parcialmente distribuída foi mesclada com a centralizada por apresentar mesma performance.



Fonte: Adaptada de (Palleschi et al., 2020)

compartilhar entre si informações como estado atual e intenções de ações a serem tomadas. Caso o robô se encontre em um conflito com outro robô, eles devem começar a trocar mensagens entre si para poder achar uma solução para o problema. Para tal, um dos robôs envolvidos aplica um algoritmo de planejamento de caminhos de múltiplos robôs, integrando dentro dele informações de mapas, sensores e informações de planejamentos de caminhos pretendidos pelo outro robô envolvido no conflito.

Figura 3.4 – Exemplo de deadlock ocorrendo entre três robôs.

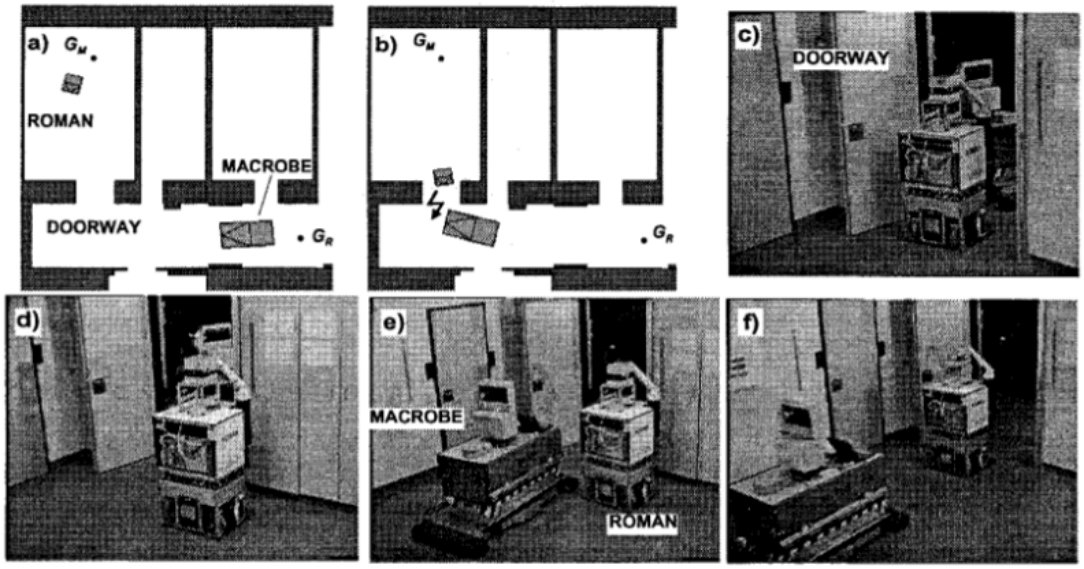


Fonte: (Azarm; Schmidt, 1997)

Esse planejamento de caminhos funciona levando em conta prioridades dos robôs, que são determinadas a partir da situação que eles se encontram. Basicamente, robôs com prioridades maiores podem planejar seus caminhos primeiro, levando em consideração apenas as informações do mapa, e os robôs com prioridades menores devem levar em consideração as informações do mapa e o caminho planejado dos robôs de prioridades maiores. As prioridades são definidas a partir de uma heurística que decide a ordem em que os robôs vão realizar seus planejamentos de caminhos. Um exemplo de funcionamento do algoritmo pode ser visualizado na figura 3.5, onde ocorre um deadlock quando

dois robôs tentam atravessar a entrada da porta ao mesmo tempo.

Figura 3.5 – Resolução de conflito onde dois robôs tentam atravessar a entrada da porta simultaneamente.



Fonte: (Azarm; Schmidt, 1997)

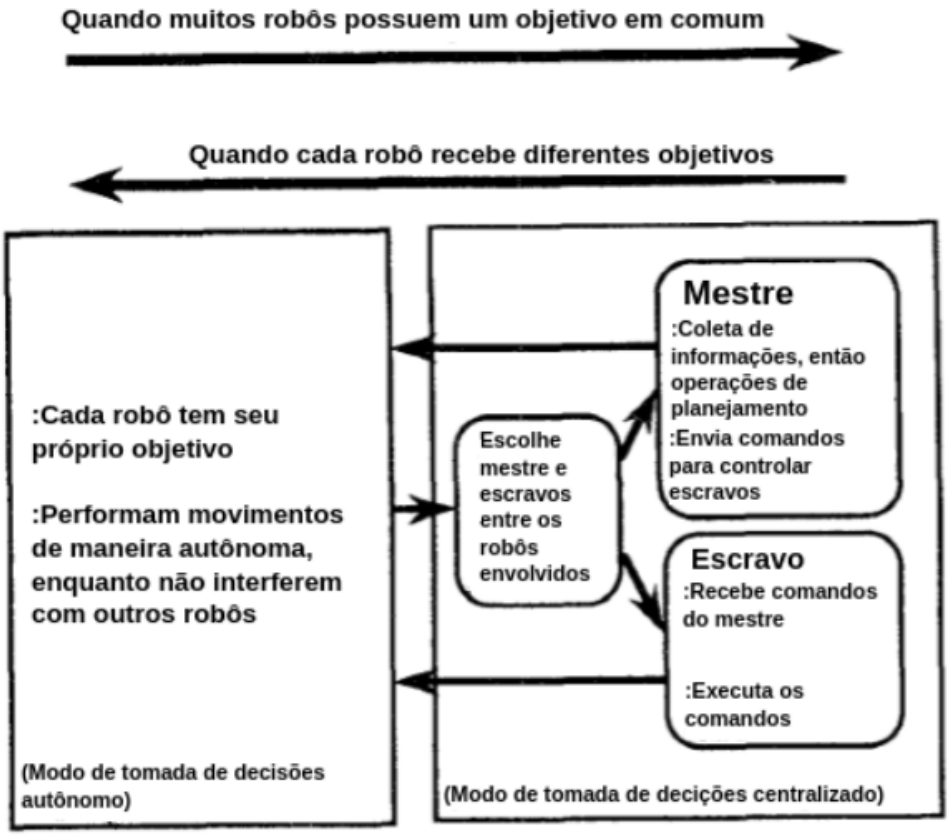
É importante perceber que o método proposto por (Azarm; Schmidt, 1997) não pretende evitar que o problema de congestionamento em corredores longos e apertados ocorra, mas tenta resolver o problema uma vez que ele acontece. E ainda é possível que esse conflito específico não seja resolvido com sucesso, uma vez que um dos robôs terá que desviar totalmente do caminho original para resolver o conflito.

(Sauer et al., 2022), em seu trabalho, divide o objetivo dos robôs em vários destinos intermediários diferentes, pré-definidos no mapa. Além disso, existem vários locais de espera espalhados pelo mapa, sendo que esse número de locais de espera deve ser maior do que o número de robôs na frota. O robô sempre precisa alocar o próximo destino intermediário em seu caminho. Caso esse destino já esteja sendo ocupado por outro robô, o robô requerente se desloca até o ponto de espera mais próximo desse destino no mapa. Uma vez que o destino é liberado, o robô volta a seguir o seu caminho. Dessa forma, o trabalho proposto por (Sauer et al., 2022) se torna útil para evitar que conflitos ocorram, porém os robôs precisam manter uma comunicação constante para a alocação dos recursos.

O trabalho realizado em (Yuta; Premvuti, 1992) comenta uma maneira de resolver deadlocks no geral. Na proposta do trabalho, os autores optaram por um mecanismo de controle que não fosse totalmente descentralizado. Sua proposta foi a de criar um sistema de múltiplos robôs distribuídos, até que os robôs entrem em um estado de deadlock. Nesse

momento em que os robôs entram em conflito, o controle dos robôs entra em um estado centralizado, porém não sendo controlado pelo sistema supervisor distribuidor de tarefas, mas sim por um robô escolhido como mestre em um sistema mestre-escravo. Nesse sistema, um dos robôs envolvidos no deadlock é escolhido para ser o mestre, sendo que o mestre fica responsável pelo planejamento de caminho e controle dos demais robôs (escravos) envolvidos no conflito. Esses modos de controle dos robôs podem ser visualizados na figura 3.6.

Figura 3.6 – Modos de controle distribuído e centralizado.



Fonte: Adaptada de (Yuta; Premvuti, 1992)

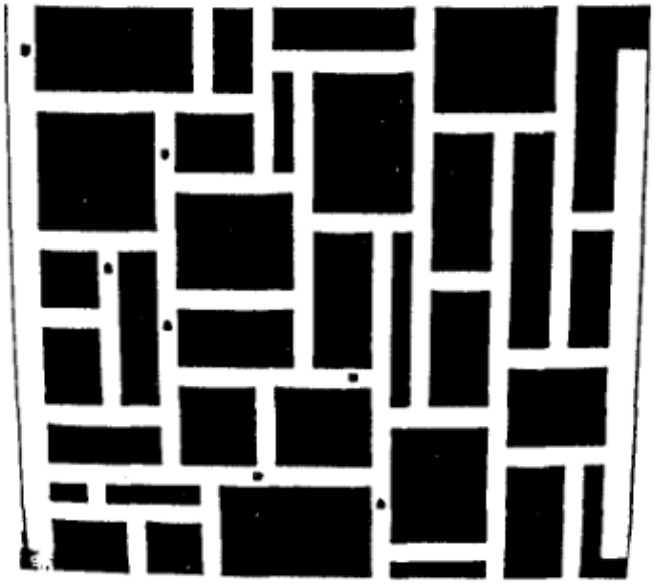
Diferente do primeiro método analisado, (Yuta; Premvuti, 1992) tenta prevenir que ocorra o problema de dois robôs conflitando em um corredor longo com pouco espaço de manobra. Em seu método, todos os robôs devem comunicar, constantemente, para os demais robôs as informações sobre seus status e recursos (como corredores e cruzamentos) utilizados no momento. Essas informações enviadas e recebidas por cada robô passam por um processo de gerenciamento de recursos, próprio de cada um, que vai auxiliar no controle do robô de forma a evitar que conflitos venham a ocorrer. Caso um robô queira entrar em um corredor longo e estreito, por exemplo, ele deve antes requisitar ao seu processo gerenciador de recursos o acesso aquele corredor. Caso o gerenciador tenha

a informação de que nenhum robô está ocupando a área desejada, uma mensagem é enviada a todos os robôs avisando que aquele espaço está reservado para o robô que deseja atravessar o caminho naquele momento, e assim ele pode seguir o seu rumo.

Os testes realizados com sucesso pelo autor foram feitos em um mapa que apresenta apenas corredores estreitos e longos, como pode ser verificados pela figura 3.7, o que comprova sua validade para a solução do problema. Porém, por mais que o método citado realmente ajude a evitar que o problema dos corredores estreitos ocorra, é importante perceber que o método exige uma comunicação constante entre todos os robôs da frota. Levando em consideração que, em um ambiente de galpões logísticos, nós estamos tratando de várias centenas de robôs, e que a tendência é que a quantidade de robôs aumente, esse método, por mais que funcione, já se torna um pouco menos atrativo devido a essa comunicação constante entre todos os robôs. Nesse tipo de ambiente de indústria, torna-se necessário um método que venha a ocorrer menos troca de mensagens entre os robôs.

(Kim; Jeon; Ryu, 2007) também utiliza um processo de reserva para evitar conflitos. Nesse artigo é usado um processo de reserva de grids do mapa, com prioridades, usando um grafo de reservas para evitar que deadlocks e colisões ocorram. Todos os robôs precisam compartilhar desse grafo para saber se o caminho a ser percorrido é válido ou não, e alocar novos caminhos.

Figura 3.7 – Mapa composto por corredores estreitos e longos.

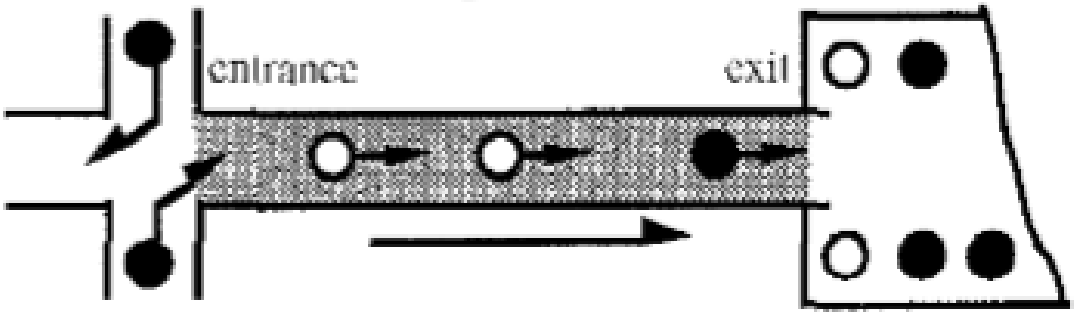


Fonte: (Yuta; Premvuti, 1992)

(Kato; Nishiyama; Takeno, 1992) e (Wang; Premvuti, 1995) realizaram trabalhos

na área das arquiteturas distribuídas de frotas de robôs, tentando resolver problemas de deadlock e conflitos através do uso de um sistema regulador de tráfego. Nesses casos, eles tentam estabelecer determinadas regras no mapa que todos os robôs devem seguir. A ideia nesses trabalhos é de tentar simular leis de trânsito dentro de um ambiente industrial. Nesse caso, a solução encontrada pelos dois artigos para resolver o problema dos corredores estreitos e longos foi a de criar uma regra que determina que essas passagens sejam unidirecionais, ou seja, com apenas uma entrada e uma saída, como se pode observar na figura 3.8.

Figura 3.8 – Exemplo de corredor unidirecional.



Fonte: (Wang; Premvuti, 1995)

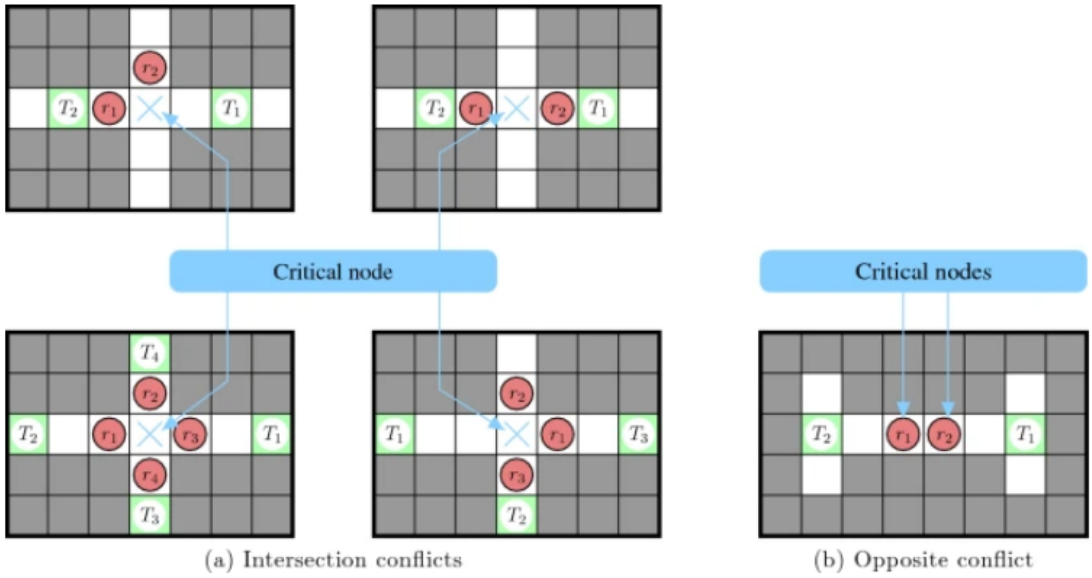
A proposta de permitir que os corredores possam ser percorridos por apenas um sentido funciona, porém acaba limitando muito a resolução de tarefas de cada robô da frota. E ainda deve-se levar em consideração que, dependendo do mapa, pode existir um único corredor que conecta dois espaços diferentes do mapa. Nesse caso, os robôs que atravessassem o corredor não poderiam mais retornar para a área original deles.

(Maoudj; Christensen, 2023) desenvolveu um trabalho que também tenta evitar que deadlocks, como os da figura 3.9, ocorram. Em seu trabalho, é utilizado um grafo para simbolizar o mapa, e antes de avançar em cada nodo no mapa, o robô deve analisar o nodo a sua frente, no caminho já planejado por ele, para detectar possíveis conflitos que venham a ocorrer com outros robôs.

Caso algum robô detecte um conflito, ele deve replanejar o seu caminho enquanto adiciona obstáculos virtuais em cada nodo vizinho ocupado. Se todos os nodos vizinhos estiverem ocupados, o robô deve solicitar um nodo livre aos robôs vizinhos, que devem disponibilizá-lo, e replanejar o seu caminho com base nos novos nodos liberados.

Cada robô recebe uma prioridade para resolver os conflitos. No caso em que os robôs conflitarem por estarem andando em sentidos opostos de uma passagem, o robô que possui a maior prioridade passa adiante enquanto o outro robô sai do caminho se dirigindo para um nodo livre. Caso o robô com menor prioridade não encontre um nodo livre, ele

Figura 3.9 – Tipos de conflitos que devem ser evitados.



Fonte: (Maoudj; Christensen, 2023)

escolhe o nodo do robô que está atrás dele, e solicita que ele libere um nodo. Isso se sucede até que não ocorra mais conflitos.

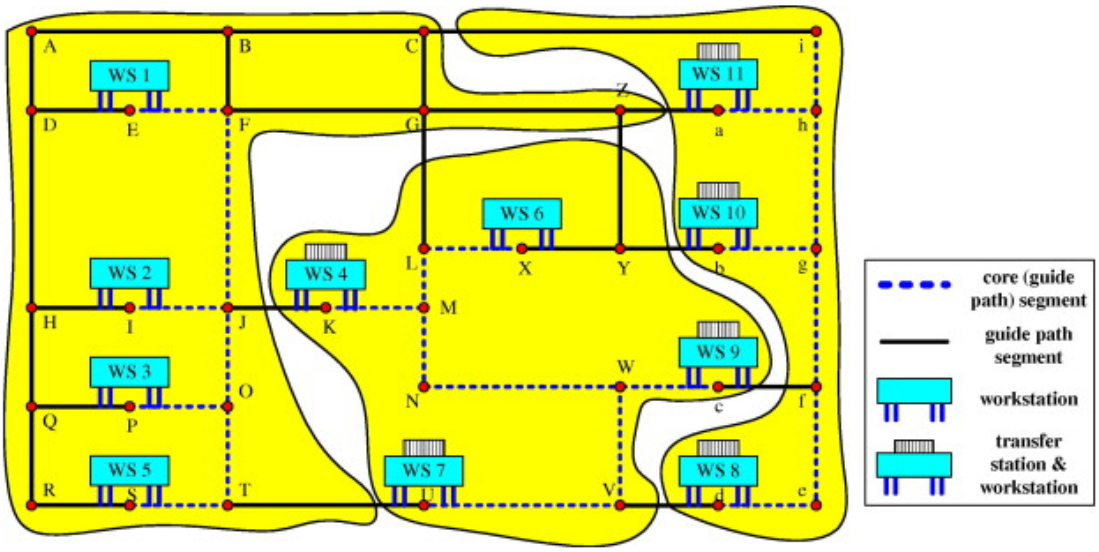
(Zhao et al., 2021) também se utiliza de uma metodologia bem semelhante, porém ele não armazena o mapa em um grafo. O mapa é armazenado em diferentes zonas de tamanhos iguais, e quando ocorre um conflito o robô de menor prioridade se desloca para a zona livre mais próxima que não faz parte do caminho do outro robô.

(De Ryck; Versteyhe; Debrouwere, 2020) comenta que outra tecnica muito utilizada para evitar que deadlocks e colisões ocorram é o controle de zonas. Nesse tipo de método é necessário dividir o mapa em diferentes zonas e atribuir, para cada uma dessas zonas, um número informando a quantidade de robôs que podem passar por essas regiões. Muitas pesquisas na área utilizam esse método de zonas para evitar deadlocks. (Fanti, 2002) cria um método de validação de caminhos e zonas para confirmar se o caminho do robô respeita as regras de tráfego de cada zona no mapa. E (Ho; Liao, 2009) desenvolveu um código que divide o mapa em diferentes zonas, e decide qual a quantidade máxima adequada de robôs dentro de determinada zona para se evitar ao máximo que deadlocks ocorram. Para o caso dos corredores estreitos, por exemplo, é permitido que apenas um robô passe por vez. Na figura 3.10 é possível visualizar um exemplo de mapa separado em diferentes zonas criadas por seu algoritmo.

(Li et al., 2016) também fez um estudo sobre o controle de zonas, onde ele se utiliza de zonas menores no mapa, dessa maneira permitindo um robô por vez em cada zona do mapa. De acordo com o autor, tudo que o robô faz durante a execução de sua

tarefa é passar de uma zona pra outra. Para isso é necessário analisar sempre se a zona seguinte do mapa se encontra livre ou ocupada por outro robô. Dessa maneira é possível evitar deadlocks.

Figura 3.10 – Resultado final do mapa dividido em zonas.



Fonte: (Ho; Liao, 2009)

3.3 Comparação dos Métodos

A seguir, na tabela 3.1, podemos observar um resumo de algumas características dos tipos de metodologias estudadas na seção 3.2 para se resolver o problema das passagens estreitas.

A proposta apresentada neste trabalho tenta evitar ao máximo que possíveis conflitos e deadlocks ocorram durante a execução das atividades da frota de robôs. Porém não foi apresentado um método de resolver os deadlocks quando eles ocorrem, ficando assim a cargo de um sistema externo, ou um algoritmo presente nos próprios robôs, resolver os problemas quando eles aparecerem.

Levando as informações presentes na tabela 3.1 em consideração, é possível concluir que, dentre as propostas que tentam evitar os conflitos de acontecerem, a proposta apresentada neste trabalho, em comparação às outras estudadas na seção 3.2, apresenta o melhor desempenho de uma arquitetura totalmente distribuída em termos de números de mensagens trocadas com outros robôs ou com um sistema externo. A proposta presente em (Kato; Nishiyama; Takeno, 1992) e (Wang; Premvuti, 1995) é a única, além da proposta apresentada, que evita conflitos e possui baixa quantidade de troca de mensa-

Tabela 3.1 – Comparação dos métodos estudados na seção 3.2.

Métodos	*Evita Conflitos*	*Resolve Conflitos*	*Quantidade de troca de mensagens*	*Vantagens*	*Desvantagens*
(Azarm; Schmidt, 1997)	Não	Sim	Baixo	Troca de mensagens apenas com os robôs envolvidos no conflito	Não tenta evitar que conflitos ocorram
(Yuta; Premvuti, 1992) / (Kim; Jeon; Ryu, 2007) t	Sim	Sim	Alto	Consegue evitar que conflitos ocorram	Troca constante de mensagens entre todos os robôs da frota
(Kato; Nishiyama; Takeno, 1992) / Wang; Premvuti, 1995)	Sim	Não	Baixo	Consegue evitar que conflitos ocorram; baixa quantidade de troca de mensagens	Utiliza corredores unidirecionais, limitando a resolução de tarefas
(Maoudj; Christensen, 2023) / (Zhao et al., 2021)	Não	Sim	Baixo	Troca de mensagens apenas com os robôs envolvidos no conflito	Não tenta evitar que conflitos ocorram
(Fanti, 2002) / (Ho; Liao, 2009)	Sim	Não	Alto	Consegue evitar que conflitos ocorram	Monitoramento constante das diferentes zonas; robôs devem informar suas posições constantemente
Proposta do atual trabalho	Sim	Não	Baixo	Consegue evitar que conflitos ocorram; baixa quantidade de troca de mensagens	Atualmente permite a passagem de um robô por vez; não procura outro caminho quando identifica um possível conflito

Fonte: De autoria própria.

gens, porém ela limita os corredores a serem unidirecionais, o que pode dificultar muito a realização das tarefas dos robôs a depender do mapa sendo utilizado.

4 PROPOSTA

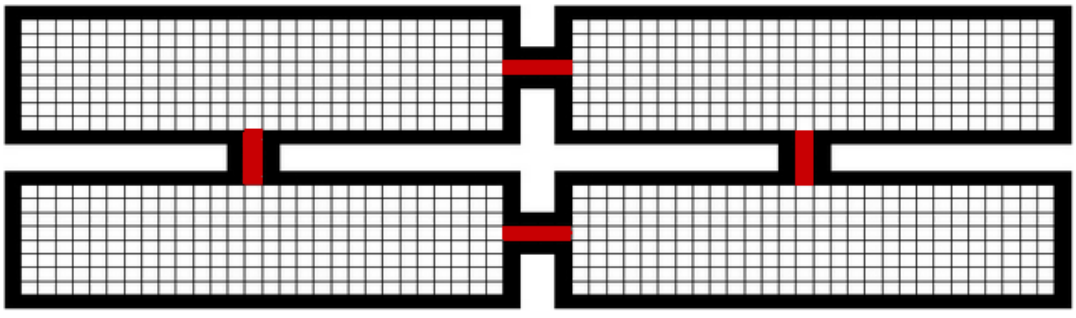
Muitos dos artigos recentes, presentes nas referências bibliográficas deste trabalho, ainda citam o problema dos corredores estreitos e longos como um dos principais problemas das arquiteturas distribuídas de frotas de robôs.

Neste capítulo, será apresentada a proposta de um método com o objetivo de evitar que conflitos, como o dos corredores estreitos, ocorram durante a execução da frota de robôs em um sistema distribuído. Além disso, será comentado como foi realizado o processo de simulação e implementação dos testes desta proposta.

4.1 Método

Este trabalho apresentará uma solução para o problema dos corresdores estreitos utilizando uma arquitetura totalmente distribuída. Para isso, é necessário que o sistema possua alguns pré-requisitos. Esses requisitos são que os robôs devem ter conhecimento total do mapa de antemão, e os pontos críticos do mapa também devem ser de conhecimento comum de todos, inclusive o sistema supervisor. Esses pontos críticos devem ser definidos pelo usuário que for fazer uso da frota, e não precisam ser exclusivamente sobre os corredores apertados, mas qualquer ponto do mapa onde possa ocorrer algum conflito ou deadlock. Um exemplo de mapa assim pode ser visualizado na figura 4.1.

Figura 4.1 – Mapa contendo pontos críticos (marcados em vermelho)



Fonte: (González et al., 2022)

Como já comentado anteriormente, em um gestor de frotas de robôs com arquitetura distribuída, existe um sistema externo supervisor que apenas monitora o status dos robôs de tempos em tempos e distribui as tarefas entre os membros da frota. Todo o planejamento de caminhos e controle de cada robô fica a cargo dos próprios robôs. A ideia por trás dessa proposta é manter essas características do sistema supervisor na arquitetura

distribuída.

Durante a execução da frota, cada um dos robôs deve possuir as informações da área que cada ponto crítico ocupa dentro do mapa. Além disso, cada um dos robôs deve, constantemente, calcular a sua respectiva distância para esses pontos críticos e, caso esteja a uma determinada distância pré-estabelecida de algum ponto crítico, deve notificar o servidor caso precise passar pelo ponto.

Enquanto os robôs não precisarem passar por nenhum ponto crítico, eles devem seguir seus caminhos planejados até os seus objetivos normalmente, até se aproximarem de uma região crítica. Quando o robô se aproximar da região, ele deve enviar uma mensagem para o supervisor avisando ele de que gostaria de entrar na área crítica. O supervisor, por sua vez, vai ter uma estrutura de dados do tipo fila para cada ponto crítico (PCn) do mapa, onde ele terá armazenado o id de todos os robôs que requisitarem entrar naquele ponto. Caso a fila do ponto crítico específico que o robô deseja entrar esteja vazia no momento da requisição, aquele robô poderá continuar seguindo seu caminho sem problema algum, sendo posto no início da fila. Porém, caso já exista algum robô atravessando a região, esse novo robô requerendo a passagem também deverá ser colocado no final da fila e receberá um objetivo temporário novo de ir até um local seguro para esperar a sua vez de passar na região. Quando o robô que estiver atravessando a área crítica deixar o local, ele deverá enviar uma mensagem notificando o supervisor. Nesse momento, o supervisor irá liberar o próximo da fila para seguir com o seu objetivo original. O algoritmo 1 mostra o processo realizado no lado do supervisor.

Algorithm 1 Critical Point (*PC*) control

```
if robot wants to enter PCn then
    if PCn queue is empty then
        insert robot in PCn queue and let it pass
    else
        insert robot in PCn queue
        send robot to waiting position
    end if
else if robot is leaving PCn then
    remove robot from PCn queue
    send original goal to next robot in queue
end if
```

É importante que exista alguma região segura onde os robôs possam esperar por sua vez de avançar, impedindo assim que vários robôs que estejam esperando tranquem a saída do ponto crítico para o robô que estiver atravessando. A sugestão é que esse ponto

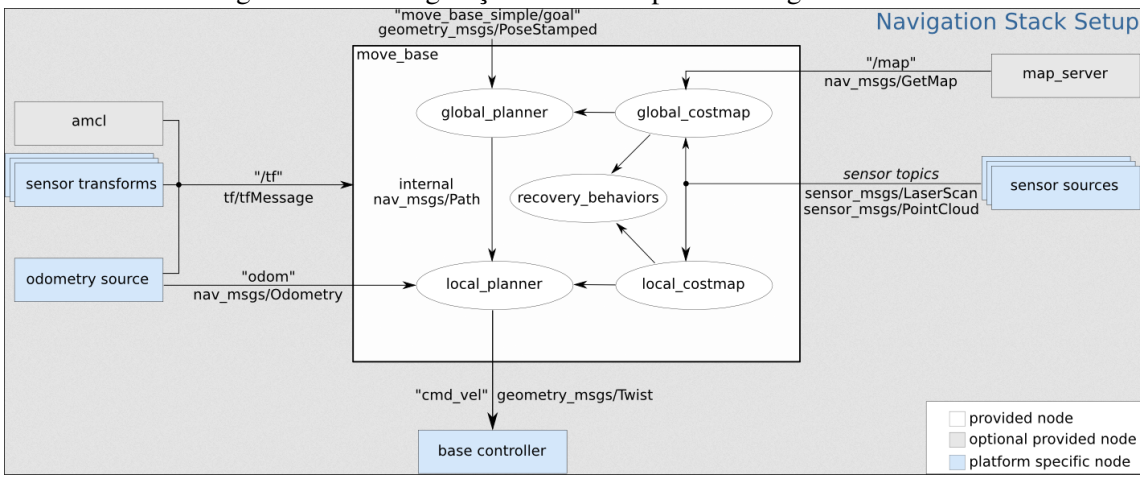
de espera seja o mais perto possível da região por onde se deseja passar, talvez na própria lateral das passagens, se assim for possível. Dependendo do mapa, pode-se apenas pedir para que o robô pare no local, sem ir para um ponto específico, aguardando sua vez.

Com esse algoritmo, apesar de haver um sistema externo supervisor que atua de forma centralizadora, a arquitetura continua sendo distribuída em termos de planejamento de caminhos, visto que o supervisor não controla nem planeja o caminho dos robôs. Ele apenas monitora o status atual das zonas críticas no momento em que algum robô requisitar a passagem por um ponto crítico e, caso necessário, envia um novo objetivo para o robô ir para um local seguro de espera. É importante salientar que essa proposta tenta impedir ao máximo que ocorram quaisquer tipos de conflitos ou deadlocks durante a execução da frota, porém estados de deadlocks ainda podem acontecer. Nos casos em que acontecerem os deadlocks, é necessário que exista algum resolvedor de conflitos entre robôs que trabalhe na resolução do problema. A presente proposta diminui consideravelmente as chances de ocorrerem conflitos, mas não os resolve caso ocorram.

4.2 Framework de simulação

O trabalho foi desenvolvido usando a ferramenta rospy, que é uma biblioteca escrita usando a linguagem de programação Python para facilitar a interface com o ROS. O planejamento de caminhos do robô se dá a partir do Navigation Stack do ROS, que funciona usando o algoritmo de planejamento Dijkstra. Na figura 4.2, podemos ver a configuração dos robôs necessária para o Navigation Stack funcionar.

Figura 4.2 – Configuração necessária para o Navigation Stack

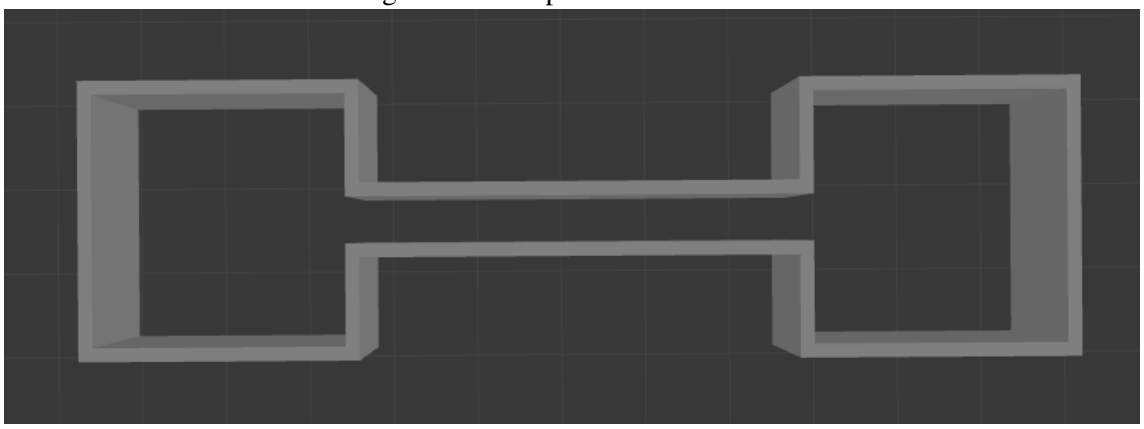


Fonte: (ROS.org, 2018)

Foram utilizados os ambientes de simulação Gazebo e Rviz. Extensivos testes

foram executados utilizando o mapa criado no Gazebo, mapa que pode ser visualizado na figura 4.3.

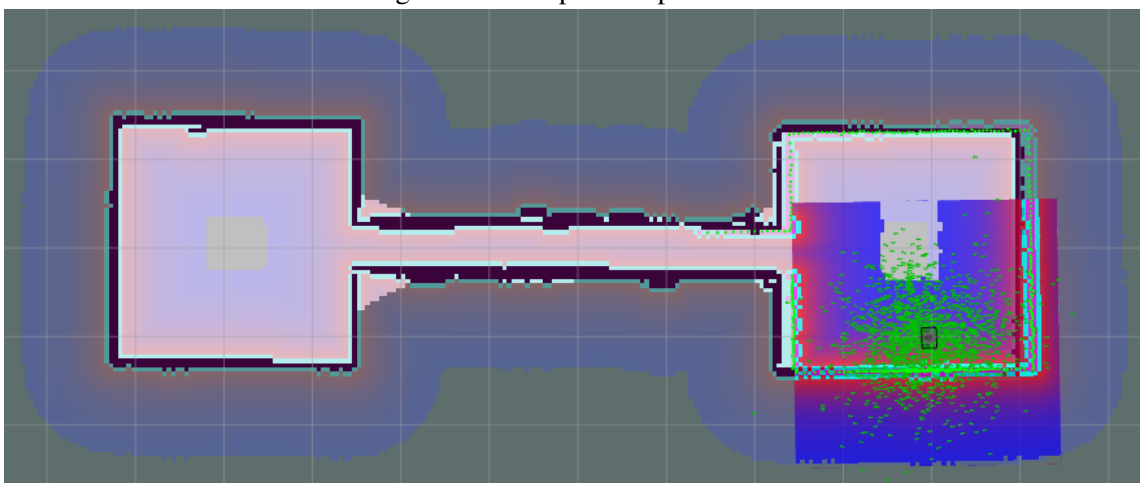
Figura 4.3 – Mapa criado no Gazebo



Fonte: De autoria própria.

A partir do mapa criado no Gazebo, foi feito um processo de Localização e Mapeamento Simultâneos (SLAM) no Rviz para ter acesso ao mesmo mapa com informações de custos globais de obstáculos para melhor guiar o robô pelo Rviz. O mapa extraído pelo processo de SLAM foi utilizado nos testes pelo Rviz, que simula as informações de sensores utilizados por cada robô no ambiente virtual. A imagem de um robô no ambiente virtual Rviz pode ser vista na figura 4.4.

Figura 4.4 – Mapa visto pelo RViz



Fonte: De autoria própria.

Os robôs se localizam através do algoritmo de localização Monte Carlo Adaptativo (AMCL). E utilizam um planejador global para planejar o seu caminho até o objetivo. Além disso, utilizam um planejador local útil para identificar e desviar de possíveis obstáculos que apareçam no caminho. Importante salientar que, nesses testes, todos os robôs

são considerados obstáculos. Não há comunicação direta entre robôs. Eles apenas enviam notificação de status para o sistema externo supervisor e recebem objetivos dele.

4.3 Implementação

Para o trabalho, foram criados dois nodos ROS. Um nodo representando o lado do sistema supervisor, e o outro representando a frota de robôs. O nodo do sistema externo, quando iniciado, envia um objetivo diferente para cada um dos robôs da frota e fica em repouso até receber alguma notificação dos robôs solicitando um novo objetivo ou requisitando a passagem por uma área crítica. No caso de um robô notificar que chegou ao seu objetivo, um novo objetivo é designado a ele. Mas caso o robô notifique que pretende entrar em uma área crítica, o supervisor então vai executar o algoritmo 1.

O nodo do lado da frota de robôs fica em repouso até receber um objetivo vindo do nodo do sistema externo supervisor. Nesse caso, o respectivo robô que recebeu a tarefa deve se deslocar até o seu local de destino utilizando seu algoritmo de planejamento global e local. Caso um robô chegue a uma distância de 1.3 metros (distância suficiente para não atrapalhar a passagem dos outros robôs) de um ponto crítico já definido no mapa, ele deve notificar o sistema externo supervisor desse fato. Após notificar o supervisor, ele deve aguardar uma resposta dele, que avisará se o robô pode seguir para o seu objetivo, ou se receberá um novo objetivo para aguardar sua vez de avançar. Na figura 4.5, é possível ver o ponto crítico que foi marcado no mapa de cada um dos robôs.

Figura 4.5 – Ponto crítico contido no mapa de cada robô.

Fonte: De autoria própria.

Neste trabalho, como não foram realizados testes com mais de quatro robôs, não houve necessidade de enviar os robôs para um ponto específico do mapa para aguardar

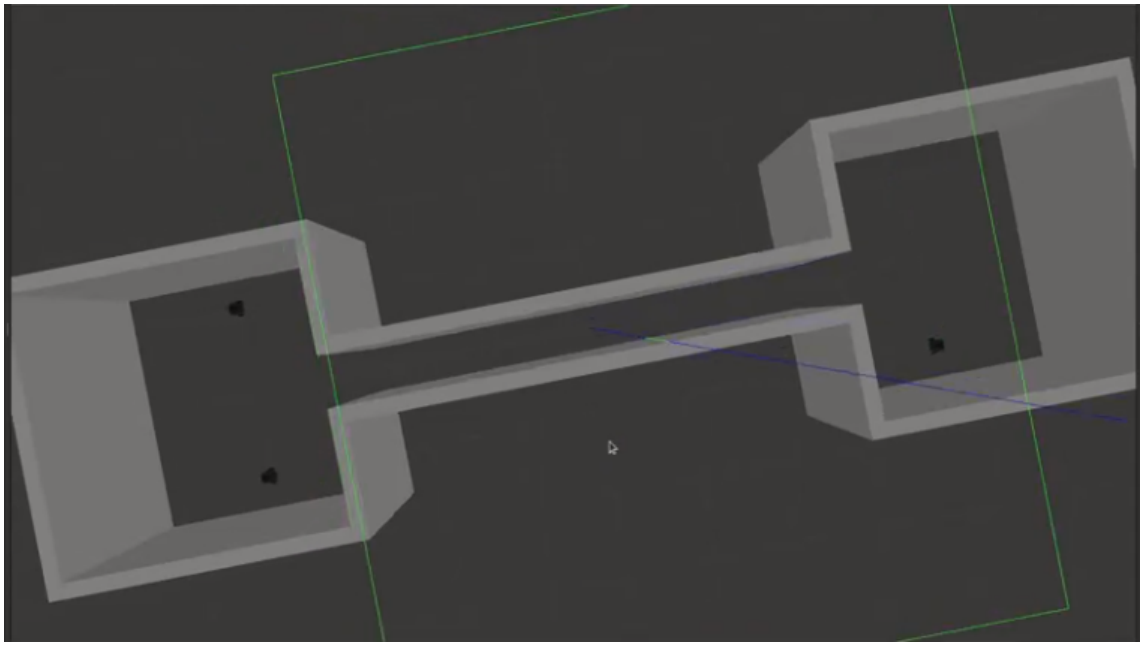
suas vezes. Quando é solicitado que os robôs aguardem, eles recebem um objetivo de ficar em suas posições atuais, ou seja, eles recebem a missão de ficarem parados esperando suas vezes.

5 RESULTADOS

Os testes da solução proposta no presente trabalho foram desenvolvidos e executados em um computador com intel CORE i5 da 11º geração, com 250 GB de SSD e 8 GB de memória. Foi utilizado nele o sistema operacional Linux Ubuntu 20.04.6 LTS com a versão Noetic do Ros.

Foram realizados diversos testes utilizando o mapa visualizado na figura 4.3. Os testes foram executados com 2, 3 e 4 robôs turtlebots recebendo suas tarefas simultaneamente para áreas opostas àquelas onde os robôs se encontravam, forçando assim que todos os robôs tentem cruzar o ponto crítico ao mesmo tempo. A posição inicial dos robôs, no caso dos testes feitos com três robôs, pode ser vislumbrada na figura 5.1, e os objetivos recebidos por cada robô, assim como o resultado de seus planejamentos de caminhos globais, podem ser visualizados na figura 5.2.

Figura 5.1 – configuração inicial dos robôs no mapa.

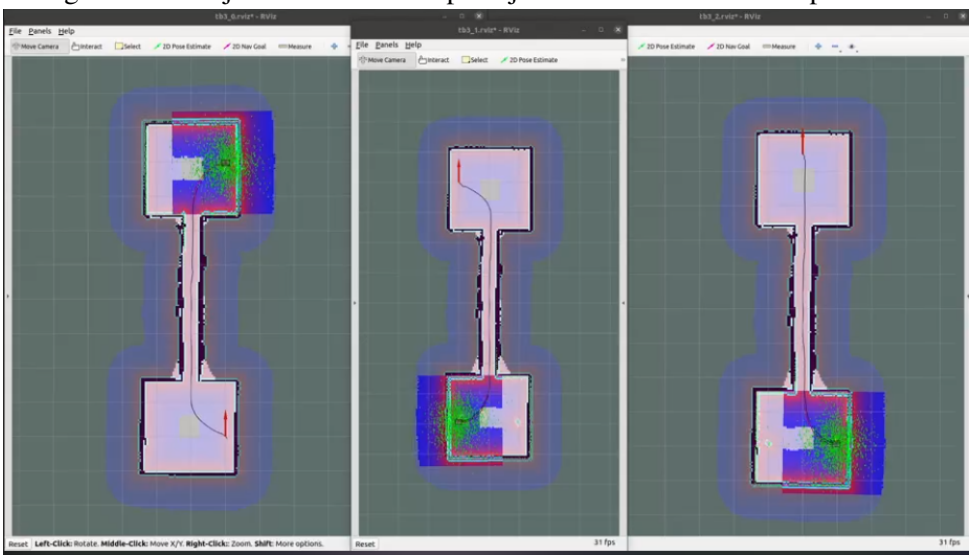


Fonte: De autoria própria.

Os testes possuem o propósito de analisar como a frota de robôs atua quando eles precisam passar por um ponto crítico com a grande possibilidade de um conflito ocorrer. O sucesso desses testes se dá quando todos os robôs da frota conseguem chegar aos seus objetivos sem que ocorra a ocorrência de conflitos entre os robôs.

Quando os robôs se aproximam do ponto crítico marcado (via código) no mapa, eles enviam uma requisição para o sistema supervisor pedindo para avançar no mapa seguindo com os seus caminhos. O primeiro robô a enviar o pedido pode seguir avançando

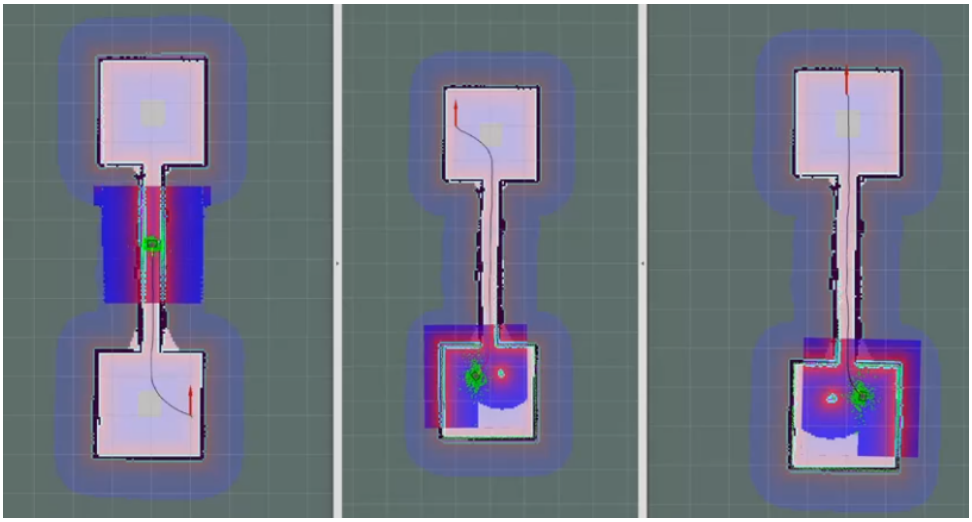
Figura 5.2 – Objetivos e caminhos planejados de cada robô vistos pelo RViz.



Fonte: De autoria própria.

pelo ponto crítico. Os demais recebem um novo objetivo temporário pedindo para parar em suas posições atuais. Na figura 5.3, podemos ver um robô que conseguiu avançar no mapa, enquanto os outros dois ficaram esperando em seus respectivos lugares.

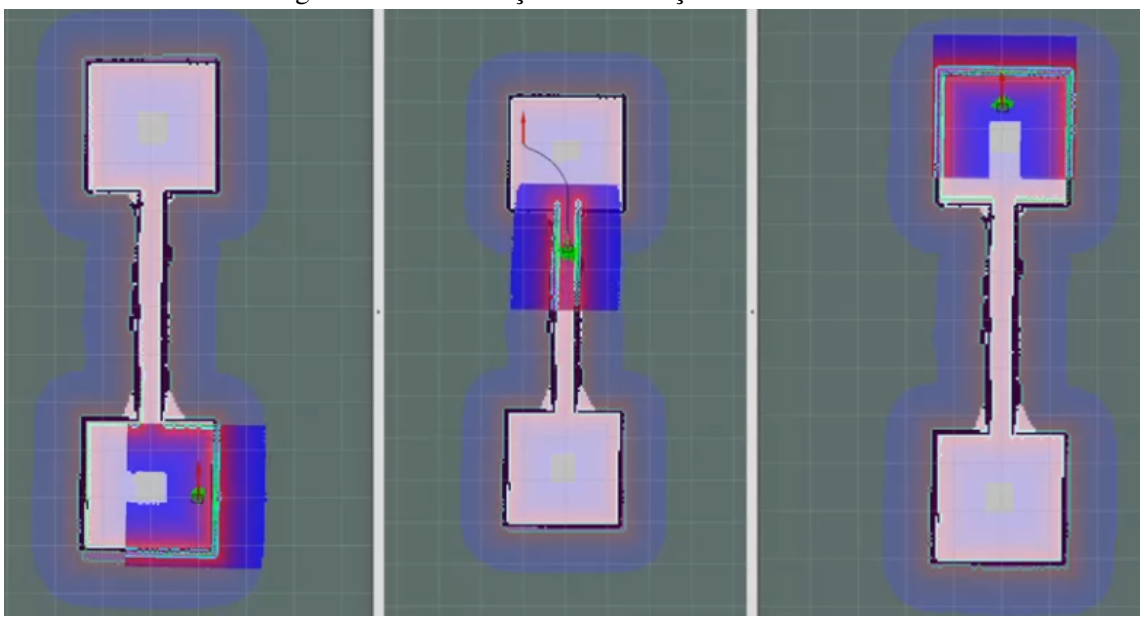
Figura 5.3 – Um robô avança pelo ponto crítico enquanto os outros esperam suas vezes.



Fonte: De autoria própria.

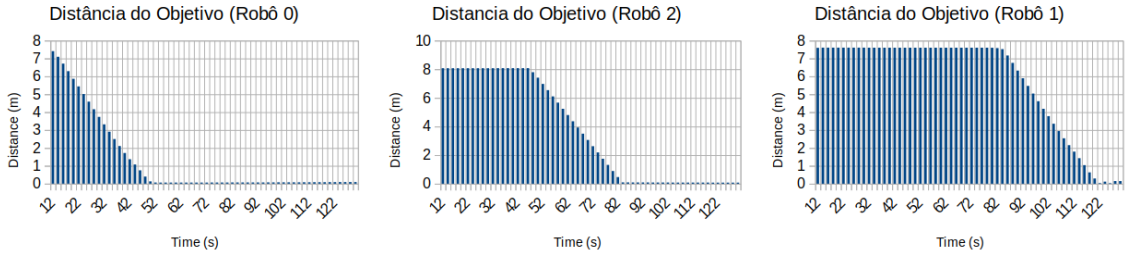
Após o robô se afastar por uma determinada distância do ponto crítico, o próximo robô na fila do lado do supervisor recebe o seu objetivo original e, assim, também recebe o aval para poder avançar pelo ponto crítico. A distância estabelecida de referência para todas as notificações enviadas pelos robôs, neste trabalho, foi de 1.3 metros. Na figura 5.4, podemos ver a finalização da simulação dos três robôs avançando simultaneamente. E na figura 5.5 é possível analisar os gráficos mostrando a distância euclidiana de cada robô até seus objetivos durante o período da execução da simulação.

Figura 5.4 – Finalização da simulação com 3 robôs.



Fonte: De autoria própria.

Figura 5.5 – Gráficos demonstrando a execução da simulação.



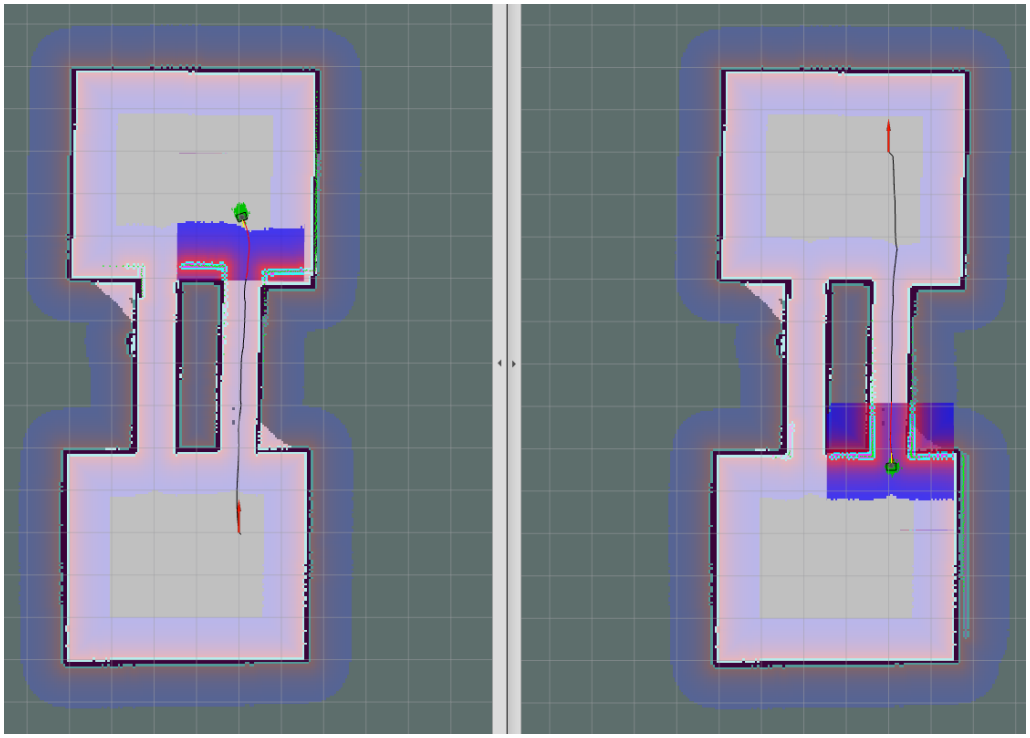
Fonte: De autoria própria.

Foram realizados mais testes nesse mapa com 2 e 3 robôs também. Todos os testes foram bem-sucedidos, sem ocorrer problemas em nenhuma das simulações. Além desse mapa, também foi executado testes em um mapa mais simples, onde dois robôs tentam atravessar a passagem de uma porta. Esses testes tiveram suas simulações igualmente bem-sucedidas, demonstrando a eficiência do método nesses ambientes com áreas contendo pontos críticos.

A presente proposta apresentada possui o objetivo de evitar que conflitos entre robôs ocorram no mapa utilizando o menor número de troca de mensagens possível. Porém um menor número de mensagens não torna essa, necessariamente, a opção mais rápida e eficaz disponível. Um exemplo onde podemos perceber que a proposta não é a mais rápida está presente no mapa da figura 5.6. Neste mapa, podemos ver que mesmo possuindo uma maneira de evitar o conflito indo por outro caminho, o robô decide esperar o outro passar para poder avançar. Isso se deve ao fato de que a presente proposta, atualmente, não apresenta uma maneira de fazer o robô procurar outro caminho possível. Ele

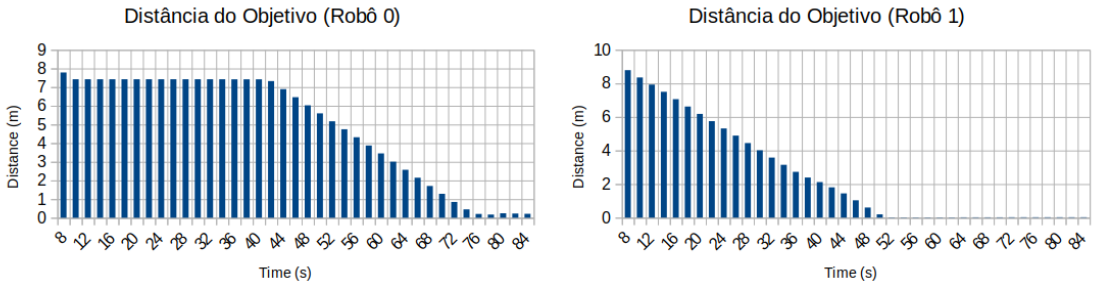
segue em seu caminho original nessas situações. Isso foi feito com o intuito de diminuir ao máximo a comunicação entre todas as partes envolvidas, permitindo, assim, uma escala maior no uso dos robôs. Na figura 5.7 podemos ver um gráfico que mostra a distância euclidiana dos robôs até os seus objetivos durante a execução do algoritmo no mapa da figura 5.6. O tempo total de execução poderia ter diminuindo mais de 30 segundos se o robô 0 tivesse seguido pelo outro caminho livre.

Figura 5.6 – Exemplo de mapa com dois caminhos possíveis.



Fonte: De autoria própria.

Figura 5.7 – Gráficos exibindo a distância dos robôs de seus objetivos durante a execução da simulação.



Fonte: De autoria própria.

Importante salientar que, neste trabalho, não houve preocupação com a resolução de conflitos. O objetivo da proposta é tentar evitar ao máximo que conflitos ocorram em pontos críticos, mas é importante que exista algum resolvedor de conflitos para o caso de

vir a ocorrer algum conflito ou deadlock durante a execução da frota de robôs.

6 CONCLUSÃO

Nos encontramos atualmente no meio da quarta revolução industrial, também conhecida como Indústria 4.0. Nesta indústria, o principal ambiente utilizado para armazenamento de produtos são os galpões logísticos, que por serem muito grandes, para suportar grandes quantidades de produtos, beneficiam-se do uso de robôs móveis autônomos para realizar o transporte de carga de um ponto a outro no galpão.

Existem atualmente dois tipos principais de arquiteturas de frotas de robôs: arquiteturas centralizadas e arquiteturas distribuídas. As arquiteturas centralizadas são as mais utilizadas atualmente, uma vez que elas oferecem maior segurança e previsibilidade durante a execução da frota, pois todos os robôs são controlados por um único sistema externo supervisor que monitora, planeja o caminho dos robôs e também cuida da movimentação deles. Porém, essa arquitetura também possui alguns problemas que tornam seu uso complicado dentro do ambiente de galpões logísticos, sendo o principal problema a dificuldade de escala da frota, já que o sistema supervisor controla todos os aspectos dos robôs. Ao considerar esse desafio contido no uso das arquiteturas centralizadas, é necessário voltarmos os olhos para as arquiteturas distribuídas.

Nas arquiteturas distribuídas, o sistema supervisor fica encarregado apenas da distribuição de tarefas para cada robô. Os robôs realizam o seu próprio planejamento e controle de movimentos para concluir o objetivo recebido do supervisor. Essa característica torna seu sistema mais escalável quanto a quantidade de robôs utilizada em ambientes de galpões logísticos. Porém, esta arquitetura possui um problema aparente, citado em diversos artigos recentes da área. Podem ocorrer conflitos entre dois robôs vindo de sentidos opostos em um corredor longo e estreito, sem espaço para manobras entre os robôs.

O objetivo do presente trabalho foi analisar este tipo de problema e desenvolver uma estratégia para tentar evitar que esse problema apresentado ocorra nas arquiteturas de frotas de robôs descentralizadas. Para isso, foi realizado um estudo aprofundado sobre diferentes estratégias que já existem na área sobre o assunto, e após o estudo uma estratégia nova foi sugerida.

O método novo apresentado mantém a característica de ser distribuído no que tange o planejamento de caminhos, com o sistema supervisor apenas enviando objetivos para os robôs, e os robôs cuidando do seu próprio controle e planejamento de caminhos. Nesta nova estratégia, os robôs que se aproximarem de um ponto crítico qualquer, já predefinido no mapa, devem enviar uma notificação para o sistema supervisor que, com base

em uma estrutura de dados do tipo fila, deve definir se o robô segue com o seu objetivo atual, ou se lhe é dado um novo objetivo temporário para que o robô aguarde sua vez de avançar pelo ponto crítico desejado.

Com base nos testes feitos neste trabalho, é possível comprovar que esse método funciona, não apenas para evitar o problema dos corredores estreitos, mas o de qualquer área crítica presente no mapa, diminuindo assim a quantidade de deadlocks e conflitos que ocorrem no ambiente entre robôs. Com a diminuição de conflitos nos ambientes podemos tornar o uso da arquitetura distribuída mais confiável para a indústria.

Ainda podemos, em um trabalho futuro, permitir que mais de um robô atravesse uma determinada área crítica ao mesmo tempo. Nos casos, por exemplo, em que mais de um robô pretende seguir no mesmo sentido do corredor, é possível permitir que esses robôs atravessem ao mesmo tempo, mantendo uma determinada distância entre eles. Nesse caso, é importante gerar uma estratégia para que vários robôs não atravessem a passagem seguidamente em um único sentido, impedindo que os robôs vindo do outro sentido possam ter suas vezes de atravessar também. Teria que se criar uma maneira de equilibrar a quantidade de robôs que atravessam uma região de uma só vez, não causando assim um desequilíbrio onde apenas os robôs de um lado do ponto crítico consigam completar suas tarefas enquanto os do outro lado ficam aguardando por muito tempo. Esse ponto pode ser melhor trabalhado no futuro.

REFERÊNCIAS

AZARM, K.; SCHMIDT, G. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In: **Proceedings of International Conference on Robotics and Automation**. [S.l.: s.n.], 1997. v. 4, p. 3526–3533 vol.4.

BERNDT, M. et al. Centralized robotic fleet coordination and control. In: **Mobile Communication - Technologies and Applications; 25th ITG-Symposium**. [S.l.: s.n.], 2021. p. 1–8.

BERTILSSON, F. et al. Centralized versus distributed nonlinear model predictive control for online robot fleet trajectory planning. In: **2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)**. [S.l.: s.n.], 2022. p. 701–706.

BICAKU, A. et al. Monitoring industry 4.0 applications for security and safety standard compliance. In: **2018 IEEE Industrial Cyber-Physical Systems (ICPS)**. [S.l.: s.n.], 2018. p. 749–754.

BRUNO, S.; KHATIB, O. **Springer Handbook of Robotics**. 2nd. ed. [S.l.]: Springer, 2016.

COLOMBO, A. W.; BANGEMANN, T.; KARNOUSKOS, S. Imc-aesop outcomes: Paving the way to collaborative manufacturing systems. In: **2014 12th IEEE International Conference on Industrial Informatics (INDIN)**. [S.l.: s.n.], 2014. p. 255–260.

De Ryck, M.; VERSTEYHE, M.; DEBROUWERE, F. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. **Journal of Manufacturing Systems**, v. 54, p. 152–173, 2020. ISSN 0278-6125. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0278612519301177>.

FANTI, M. P. Event-based controller to avoid deadlock and collisions in zone-control agvs. **International Journal of Production Research**, Taylor Francis, v. 40, n. 6, p. 1453–1478, 2002. Available from Internet: <https://doi.org/10.1080/00207540110118073>.

GIRIJA PATIL; MAREENA, J. F. J. S. K.; KAEWKHIAOLUEANG, K. Amazon robotic service (ars). In: **Engineering and Technology Management Student Projects. 2309.** [S.l.: s.n.], 2021.

GONZáLEZ, E. et al. An efficient multi-robot path planning solution using a* and coevolutionary algorithms. **Integrated Computer-Aided Engineering**, v. 30, p. 41–52, 11 2022.

HAZIK, J. et al. Fleet management system for an industry environment. **Journal of Robotics and Control (JRC)**, v. 3, n. 6, p. 779–789, 2022. ISSN 2715-5072. Available from Internet: <https://journal.umy.ac.id/index.php/jrc/article/view/16298>.

HO, Y.-C.; LIAO, T.-W. Zone design and control for vehicle collision prevention and load balancing in a zone control agv system. **Computers Industrial Engineering**, v. 56, n. 1, p. 417–432, 2009. ISSN 0360-8352. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S036083520800137X>.

KATO, S.; NISHIYAMA, S.; TAKENO, J. Coordinating mobile robots by applying traffic rules. In: **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 1992. v. 3, p. 1535–1541.

KIM, K. H.; JEON, S. M.; RYU, K. R. Deadlock prevention for automated guided vehicles in automated container terminals. In: ____. **Container Terminals and Cargo Systems: Design, Operations Management, and Logistics Control Issues**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 243–263. ISBN 978-3-540-49550-5. Available from Internet: <https://doi.org/10.1007/978-3-540-49550-5_12>.

LI, Q. et al. A control of collision and deadlock avoidance for automated guided vehicles with a fault-tolerance capability. **International Journal of Advanced Robotic Systems**, v. 13, n. 2, p. 64, 2016. Available from Internet: <https://doi.org/10.5772/62685>.

MAKARENKO, A. et al. An experiment in integrated exploration. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2002. v. 1, p. 534–539 vol.1.

MAOUDJ, A.; CHRISTENSEN, A. L. Improved decentralized cooperative multi-agent path finding for robots with limited communication. **Swarm Intelligence**, 2023. ISSN 1935-3820. Available from Internet: <https://doi.org/10.1007/s11721-023-00230-7>.

PALLESCHI, A. et al. Toward distributed solutions for heterogeneous fleet coordination. In: . [S.l.: s.n.], 2020.

PECORA, F. et al. A loosely-coupled approach for multi-robot coordination, motion planning and control. **Proceedings of the International Conference on Automated Planning and Scheduling**, v. 28, n. 1, p. 485–493, Jun. 2018. Available from Internet: <https://ojs.aaai.org/index.php/ICAPS/article/view/13923>.

RODRIGUES, G. et al. An architecture for mission coordination of heterogeneous robots. **Journal of Systems and Software**, v. 191, p. 111363, 2022. ISSN 0164-1212. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0164121222000929>.

ROS.ORG. **Setup and Configuration of the Navigation Stack on a Robot**. 2018. Available from Internet: <http://wiki.ros.org/navigation/Tutorials/RobotSetup>.

SAUER, M. et al. Decentralized deadlock prevention for self-organizing industrial mobile robot fleets. In: **2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)**. [S.l.: s.n.], 2022. p. 1–6.

WANG, J.; PREMVUTI, S. Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space. In: **Proceedings of 1995 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1995. v. 2, p. 1619–1624 vol.2.

YUTA, S.; PREMVUTI, S. Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In: **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 1992. v. 3, p. 1566–1574.

ZHAO, Y. et al. Spare zone based hierarchical motion coordination for multi-agv systems. **Simulation Modelling Practice and Theory**, v. 109, p. 102294, 2021. ISSN 1569-190X. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1569190X21000228>.

ČáP, M.; GREGOIRE, J.; FRAZZOLI, E. Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances. In: **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2016. p. 5113–5118.