A Robust, Scalable Multi-Robot Control and Coordination Framework Achieving High Throughput for Parcel Sorting Centers

Ch’ng Chee Henn

Master of Science (Computer Science)

Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman
March 2024

A ROBUST, SCALABLE MULTI-ROBOT CONTROL AND COORDINATION FRAMEWORK ACHIEVING HIGH THROUGHPUT FOR PARCEL SORTING CENTERS

By

CH'NG CHEE HENN

A dissertation submitted to the
Department of Computer Science,
Faculty of Information and Communication Technology,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Science (Computer Science)
March 2024

Copyright Statement

This dissertation is submitted in partial fulfilment of the requirements for the degree of Master of Science (Computer Science) at Universiti Tunku Abdul Rahman (UTAR). This dissertation represents the work of the author, except where due acknowledgment has been made in the text. No part of this dissertation may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

A Robust, Scalable Multi-Robot Control and Coordination Framework Achieving High Throughput for Parcel Sorting Centers

Ch'ng Chee Henn

This dissertation presents a robust, scalable multi-robot control and coordination framework for high-throughput parcel sorting centres. The research addresses the complexities of multi-robot systems in indoor settings, where homogeneous robots operate on a shared grid, focusing on enhancing pathfinding, coordination, and throughput. It begins by highlighting the need for advanced multi-robot systems in automation, identifying the limitations of existing multi-agent pathfinding (MAPF) solutions, such as static assumptions, computational overhead, and inflexibility in dynamic environments and human-robot interactions. To overcome these challenges, the research introduces a novel framework that separates the problem into path planning and resource allocation, simplifying the MAPF problem and optimizing robot coordination. The algorithmic model supports efficient path generation, dynamic resource allocation, and conflict resolution, with a focus on plan satisfiability and operational feasibility. The implementation strategy includes dynamic plan updates, operator selection, and traffic control mechanisms to enhance efficiency. Simulation experiments demonstrate the framework's adaptability and effectiveness in managing multi-robot dynamics. Unlike traditional approaches, it employs a dynamic iterative allocation method that handles uncertainties and optimizes plans in real time, significantly reducing computational demands. Due to its assumption of a nonperfect environment, unlike traditional algorithms, the framework is easier to implement. By balancing reactive and deliberative strategies, this framework bridges the gap between theoretical models and practical applications in robotics and automation. The research demonstrates significant improvements in system throughput, with pattern-matching allocators outperforming native allocators by up to 23.52%. The framework's traffic control mechanism achieves a 58.24% higher throughput compared to systems without such controls. The research's impact extends beyond parcel sorting centres, offering potential applications in warehouse management, manufacturing, and smart city logistics. Furthermore, its adaptability to dynamic environments and human-robot interactions paves the way for more seamless integration of robotic systems in human-centric workspaces, potentially revolutionizing collaborative robotics.

Keywords: Multi-Robot Systems, Multi-Agent Path Finding, Robot Coordination, Path Planning, Resource Allocation, Parcel Sorting Centers, High Throughput, Traffic Control, Pattern Detection, Simulation Experiments

Subject Areas: Robotics, Automation, Multi-Agent Systems, Pathfinding Algorithms, Logistics and Supply Chain Management, Simulation and Modeling, Artificial Intelligence

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Prof. Dr. Liew Soung Yue, for their unwavering support, invaluable guidance, and insightful feedback throughout the course of this research. Your expertise and understanding have played a pivotal role in my academic and personal growth. I am equally grateful to my co-supervisors, Dr. Ooi Boon Yaik and Mr. Wong Chee Siang, for their constructive criticism, patient guidance, and valuable suggestions, which have been instrumental in successfully completing this dissertation. Special thanks are extended to my family, particularly my father, Ch'ng Nam Teck, and my mother, Sim Joo Wek, for their love, encouragement, and sacrifices. Your constant belief in my abilities and unwavering support have strengthened and motivated me throughout this journey. I am blessed to have had the support of such a strong network of individuals who have contributed, directly and indirectly, to my academic endeavors and the completion of this dissertation. Thank you all for your part in my journey.

DECLARATION

I, Ch’ng Chee Henn, hereby declare that the dissertation is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

(CH’NG CHEE HENN)
12 March 2024

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

The advent of multi-robot systems heralds a new era in robotics, with applications proliferating across diverse domains such as food delivery, logistic sorting, and aerial surveillance, among others [1, 2]. As robots working in concert become increasingly integral to modern infrastructural and operational landscapes, they open up a rich avenue for scientific inquiry. However, the inherent complexity of multi-robot systems, coupled with the diverse requirements dictated by different application domains, poses significant challenges for researchers.

Multi-robot systems span various technical disciplines, including robot vision, mechanical design, and positioning. The scope of their applications is vast, extending to human-machine collaborative robots, disaster relief automatons, and domestic robots. These diverse requirements across different domains hinder the evaluation of the quality and effectiveness of multi-robot systems in a standardized manner. This highlights the necessity of tailoring the design and operation of multi-robot systems to meet the specific needs of the application at hand, thereby unlocking the full potential of these robotic ensembles.

This dissertation primarily delves into the operational dynamics of multi-robot systems within indoor settings. The objective is to develop a robust control framework to streamline multiple robots' pathfinding and coordination efforts. The proposed framework aims to imbibe key characteristics essential for effective multi-robot operation: robustness against imperfect robot execution, realizability, algorithmic flexibility, scalability with respect to the number of robotic agents, and enhanced system throughput, which translates to minimized task completion time.

Drawing upon a multi-robot sorting center as a contextual foundation, this dissertation seeks to understand the operational prerequisites of such a system. The setting involves multiple homogeneous robots, hereafter referred to as agents, sharing a common workspace, envisaged as a grid environment where each grid cell represents the smallest spatial unit an agent can occupy. Tasked with continuously shuttling packages between pickup and drop-off points, the operational modalities of the agents are explored in greater detail in a later section.

A series of simulation experiments will be conducted to evaluate the operational viability of the proposed framework. These simulated scenarios will facilitate an analytical exploration of the system properties, aiming to provide insights that could be broadly applicable across similar indoor domains. This dissertation aspires to contribute significantly to understanding and enhancing multi-robot systems' operational dynamics in indoor environments through this effort.

1.1.1 Motivation

The rapid growth in deploying multi-robot systems in sectors like logistics, surveillance, and delivery underscores their potential to enhance operational efficiency and tackle complex tasks beyond the capability of single robots. Yet, dynamic and unpredictable real-world environments present significant challenges, particularly in coordination and pathfinding. Current solutions often fall short in adapting to these dynamic environments, and the complexity of existing Multi-Agent Path Finding (MAPF) algorithms poses implementation challenges. Furthermore, the isolated nature of these methodologies hinders integrated, comprehensive studies and applications, highlighting the need for a unified, adaptable framework. This research is driven by the necessity to reconcile theoretical advancements with practical applicability in multi-robot systems, aiming to develop a framework that ensures seamless robot coordination and balances reactive with deliberative strategies to enhance system performance without compromising efficiency or safety.

1.1.2 Objectives

This research aims to devise a robust multi-robot control framework that resonates with real-world operational dynamics and mitigates the aforementioned challenges. The specific objectives of the proposed framework are delineated as follows:

1. **Adaptability:** The framework's ability to accommodate various settings in practical implementations ensures its versatility and applicability across diverse multi-robot operational domains. Adaptability enables the framework to handle dynamic environmental changes and unforeseen operational hurdles.

2. **Flexibility:** The framework's ability to seamlessly integrate with existing MAPF algorithms and other components facilitates easy extension and customization. Flexibility allows the framework to be used as a standalone coordination system or in conjunction with other algorithms and modules.

3. **Scalability:** The framework's ability to handle increasing agents and tasks without significantly increasing computational complexity. Scalability ensures that the framework remains efficient and effective as the size and complexity of the multi-robot system grow.

4. **Autonomy:** Endow robots with a level of autonomy to bolster robustness against dynamic environmental alterations and unforeseen operational hurdles.

5. **Efficient Management:** Enable adept multi-robot management, ensuring seamless coordination and pathfinding, pivotal for optimizing operational throughput and minimizing task completion time.

6. **Balanced Approach:** Strike a judicious balance between reactive and deliberative approaches, ensuring the framework can adeptly respond to dynamic changes while delivering high-quality solutions.

7. **Real-world Applicability:** Ensure that the framework is conducive to real-world applications, providing a tangible conduit for transitioning from theoretical models to practical implementations.

8. **AI Coordination:** Leveraging the recent advancements in AI technology, the framework aims to serve as a coordination hub for highly intelligent robots, ensuring optimized operational synergy and adaptive response to dynamic environmental changes.

By fulfilling these objectives, the proposed framework aspires to contribute significantly towards harnessing the full potential of multi-robot systems in navigating complex real-world challenges, thereby enriching the broader discourse in the realm of robotics and automation.

1.1.3 Research Questions

This study seeks to answer the following fundamental research questions:

1. How can pathfinding and coordination algorithms be enhanced to better adapt to dynamic changes in real-world indoor environments, and what is the trade-off between reactive and deliberative approaches?

2. How can a unified framework be designed to efficiently manage multi-robot systems, ensuring seamless coordination, pathfinding, and a balanced approach between reactivity and deliberation?

3. How does the proposed framework compare with existing solutions in terms of scalability, robustness, and applicability in real-world scenarios?

These questions drive the investigative narrative of this dissertation, providing a structured pathway to explore, analyze, and evaluate the proposed multi-robot control framework.

1.2 Key Contributions and Impact

The main contributions of this research are as follows:

1. The development of a novel conceptual framework separates the multi-agent pathfinding problem into two stages: path planning and resource allocation. This approach simplifies the MAPF problem into more manageable components, enabling efficient path generation and dynamic resource allocation to avoid conflicts and optimize throughput.

2. The introduction of a well-defined algorithmic model formalizes key concepts such as plans, assignments, state transitions, dependencies, solution sequences, and plan satisfiability. This model provides a solid theoretical foundation for analyzing and solving multi-agent pathfinding problems within the proposed framework.

3. The design and implementation of advanced pattern detectors, specifically for identifying direct cyclic dependencies and extended head-to-head conflicts. These pattern detectors significantly enhance the framework's ability to recognize and avoid potential deadlocks, thereby improving the robustness and reliability of the multi-robot system.

4. The development of a dynamic traffic control mechanism that optimizes agent paths by adjusting edge costs based on historical data. This mechanism effectively reduces congestion, minimizes conflicts, and improves overall system

efficiency and throughput.

5. Extensive simulation experiments that validate the effectiveness of the proposed framework in managing the operational dynamics of multi-robot systems within the context of an automated sorting center model. The experiments demonstrate the framework's ability to handle dynamic task assignments, avoid collisions, and optimize throughput under various conditions and parameters.

This framework's successful development and validation significantly advance multi-robot coordination and control strategies. By providing a structured, efficient, and scalable approach to multi-agent pathfinding, this research can enhance the performance and reliability of autonomous systems across various application domains, such as logistics, manufacturing, and warehouse automation.

The insights gained from this research lay the foundation for further investigations into advanced traffic control strategies, heterogeneous multi-robot systems, and the integration of machine learning techniques to improve the adaptability and performance of the framework. As multi-robot systems continue to gain prominence in real-world applications, the contributions of this dissertation will play a crucial role in shaping the future of autonomous systems, enabling more efficient, robust, and intelligent coordination and control of multi-robot teams.

1.3 Problem Statement

The burgeoning deployment of multi-robot systems across many sectors underscores the imperative to address the inherent challenges that come to the fore.

The pivotal issue centers on the effective coordination and pathfinding among robots, which is exacerbated by the dynamic and often unpredictable nature of real-world environments. Existing solutions frequently fail to adapt to dynamic alterations or deliver high-quality solutions, primarily owing to their standalone nature, creating a bottleneck for joint implementation or comprehensive study. Moreover, the intricacies associated with Multi-Agent Path Finding (MAPF)-based algorithms render them challenging to implement, thereby accentuating the necessity for a more integrated approach.

The primary problem this research seeks to address encapsulates the need for a robust multi-robot control framework that resonates with real-world operational dynamics. Such a framework necessitates a balanced fusion of reactive and deliberative approaches, ensuring seamless pathfinding and coordination among robots and a level of autonomy to bolster robustness against dynamic environmental alterations and unforeseen operational hurdles. Additionally, the framework should embody adaptability, ensuring its versatility and applicability across diverse multi-robot operational domains.

Another layer of complexity is introduced with the advent of highly intelligent robots powered by contemporary AI technologies. The potential of these intelligent agents is significantly amplified when coordinated effectively, a feat the proposed framework aims to achieve by serving as a smart coordination hub. This not only accentuates the real-world applicability of the proposed framework but also propels the discourse in multi-robot systems toward harnessing the full potential of AI-powered robotics in navigating complex operational landscapes.

In light of these challenges, this dissertation posits a set of research questions delineated in the Introduction, guiding the investigative narrative toward a thorough exploration, analysis, and evaluation of the proposed multi-robot control framework. Through this venture, it aspires to significantly contribute towards harnessing the full potential of multi-robot systems in navigating complex real-world challenges, thereby enriching the broader discourse in the realm of robotics and automation.

1.3.1 Scope and Limitations

The scope of this research is primarily confined to the simulation and algorithmic realm. It aims to devise a robust control framework that can potentially accommodate the imperfect execution inherent in real-world multi-robot systems. While the ultimate goal is to ameliorate the operational dynamics of multi-robot systems, the immediate focus is on creating a simulated environment that mirrors the challenges and constraints of real-world settings.

The limitations of this study emanate from its simulated nature. While efforts are made to encapsulate real-world robot limitations, the absence of physical robot testing could present a gap in validating the framework's efficacy in actual operational environments. Furthermore, the simulation assumes certain ideal conditions, which might not hold true in real-world scenarios. However, the simulated approach affords a controlled environment to meticulously examine the algorithmic intricacies and operational dynamics, laying a solid foundation for future real-world testing and refinements.

1.3.2 Technical Preliminaries

A comprehensive understanding of the proposed framework necessitates a preliminary grasp of certain technical concepts. Core among these are fundamental graph theory, graph searching algorithms, and basic discrete mathematics. These concepts form the bedrock of the algorithmic discourse presented in this dissertation.

1.4 Structure of the Dissertation

This dissertation is organized into seven chapters. The current chapter, Chapter 1, introduces the research background, motivation, objectives, research questions, and key contributions.

Chapter 2 presents a comprehensive literature review, covering the emergence of multi-agent path finding (MAPF), graph representations, preliminary path planning algorithms, MAPF variants, limitations of existing models and algorithms, and the motivation for a balanced framework incorporating both reactive and deliberative approaches.

Chapter 3 describes the conceptual model and representation employed in this research, including the classical MAPF encoding, sorting center models, the distinction between classical MAPF and the proposed model, and the introduction of the edge-to-vertex graph representation.

Chapter 4 outlines the conceptual framework, discussing the motivation, fundamentals, interpretation, and flexibility of the proposed framework. It also highlights the framework's structure and operational procedure.

Chapter 5 delves into the details of the plan generator and allocators, revisiting key assumptions, presenting a different perspective on plan generation and resource allocation, and introducing strategies for enhancing the resource allocator.

Chapter 6 defines the algorithmic model, detailing essential elements like plans, assignments, and state transitions. It explains how these components interact within the framework, identifies conditions under which plans are unsatisfiable, and scrutinizes solution approaches for directed graphs where edges do not reciprocate between nodes. This chapter aims to clarify the underlying mechanisms of the proposed multi-robot control framework and its operational logic in handling complex tasks.

Chapter 7 focuses on implementing the framework, discussing implementation strategies, pattern detectors, layout settings, traffic control mechanisms, and robustness considerations.

Chapter 8 presents the experimental setup, simulation settings, and results, comparing different allocators, evaluating the traffic control mechanism, investigating the impact of maximum reservation and timeout thresholds, and discussing the implications of the findings.

Finally, Chapter 9 concludes the dissertation by summarizing the research, highlighting key contributions and impacts, and outlining directions for future research.

This structure provides a logical progression from the research background and objectives through the conceptual and algorithmic foundations to the implementation and experimental validation of the proposed multi-robot control and coordination framework.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to Multi-Robot Systems and the Emergence of MAPF

Multi-robot systems (MRS) have become increasingly vital in addressing complex challenges across various fields such as logistics, surveillance, and disaster response [3, 4, 5]. The collaborative dynamics of MRS allow for addressing problems unattainable by individual robotic units. Nonetheless, coordinating multiple robots in shared environments introduces substantial complications, especially under dynamic and uncertain conditions.

A critical issue within MRS is the efficient management of pathfinding and coordination to prevent collisions and ensure the timely execution of tasks. Numerous strategies have been devised to counteract these difficulties, each characterized by specific presumptions, benefits, and drawbacks. In this realm, Multi-Agent Path Finding (MAPF) stands out as a notable strategy for enhancing coordination and pathfinding in MRS environments [6, 7, 8].

This chapter aims to establish a unified terminology and foundation for our subject matter and delve into existing research to foster a comprehensive and technical understanding of the motivations behind our study.

2.1.1 Rationale for Adopting MAPF and Graph Representation

Typical MAPF models provide a structured framework to model and solve the pathfinding and coordination problem for multiple robots. It formalizes the problem as a set of agents navigating on a graph, where each node represents a discrete position, and edges represent the possibility of movement between positions. This abstraction is powerful yet flexible enough to represent various real-world scenarios.

Furthermore, graph representation has long been a common language in the study of path planning, dating back to the early exploration of single robot systems. Its expressive nature and algorithmic tractability have made it a favored representation within the computer science field [9]. Iconic algorithms such as Dijkstra's Algorithm and the A* algorithm have their foundations in graph theory, and their utilization in path planning for both single and multi-robot systems underscores the importance and adaptability of graph representation [10].

1. **Expressive Representation:** Graph-based representations allow for a clear, structured, and compact depiction of the environment, making it amenable to algorithmic analysis and solution. The nodes and edges of a graph can encapsulate crucial spatial and temporal information, providing a rich basis for planning and coordination.

2. **Compatibility with Discrete Event Systems:** Many practical scenarios in

MRS can be modeled as discrete event systems. The graph representation aligns well with this, where transitions between discrete states are naturally captured as movements along the edges of the graph.

3. **Ease of Integration:** Graph-based MAPF can be integrated with other systems and frameworks. It provides a common language that can be used to communicate with other components of a larger system, such as higher-level task allocators or human-robot interfaces.

4. **Benchmarking and Comparison:** The standardization provided by the MAPF model facilitates benchmarking and comparison of different algorithms and frameworks, aiding in objectively evaluating their performance [6].

By transitioning from the broad domain of multi-robot systems to the specific framework of MAPF and graph representation, this research navigates towards a structured and algorithmically tractable approach to address the inherent challenges in multi-robot coordination and pathfinding. The subsequent sections delve deeper into the technical intricacies of MAPF, exploring existing algorithms, identifying gaps, and laying the groundwork for the proposed robust multi-robot control framework.

2.1.2 Graph Representations and Choice of Manhattan-based Representation

Various graph representations exist to model the environment and agents' movements in multi-robot systems. Common representations include grid-based graphs, visibility graphs, roadmaps, and topological graphs, each with unique advantages and disadvantages depending on the application context[11].

In this research, a Manhattan-based graph representation is adopted for simulation and experimentation purposes due to the following reasons:

- **Simplicity and Structured Representation:** Manhattan-based graphs provide a simple and structured way to represent the environment, facilitating the development and analysis of pathfinding and coordination algorithms.

- **Alignment with Real-world Scenarios:** The orthogonal movement restrictions in Manhattan-based graphs often align well with real-world constraints found in indoor environments and urban landscapes.

- **Ease of Implementation:** The straightforward nature of Manhattan-based graphs often leads to easier implementation and debugging of algorithms.

- **Widely Adopted:** Grid-like environment is widely adopted in MAPF problems [11, 12, 13, 14], especially the lifelong MAPF variants.

However, it is critical to emphasize that the proposed framework and algorithms are designed to be compatible with regular directed graph representations and do not inherently rely on the directional constraints characteristic of Manhattan-based graphs.

2.2 Preliminary Path Planning Algorithms

Path planning forms the bedrock of navigation tasks in robotics and artificial intelligence. A prevalent approach involves utilizing graph representations, where the environment is abstracted into a graph comprising nodes representing discrete positions and edges indicating permissible movements between these positions. This section succinctly recapitulates fundamental graph concepts and introduces two seminal path-planning algorithms: Dijkstra's and A* algorithms. These algorithms are pivotal for comprehending the more intricate Multi-Agent Path Finding (MAPF) models elucidated in the ensuing sections of this chapter.

2.2.1 Brief Recapitulation of Graph Concepts

A graph $G = (V, E)$ is constituted by a set of vertices V and a set of edges E. Within the path planning ambit, each vertex symbolizes a discrete position in the environment, and each edge denotes a potential movement between two positions. The weight associated with an edge typically signifies the cost of the movement, which could be construed in terms of distance, time, or any other pertinent metric.

2.2.2 Dijkstra's Algorithm

Dijkstra's algorithm is a quintessential graph search algorithm devised to solve the single-source shortest path problem for a graph with non-negative edge weights, thereby engendering the shortest path tree. This algorithm is foundational in computer science and robotics for path planning in static environments.

2.2.3 A* Algorithm

The A* algorithm augments Dijkstra's algorithm with heuristics to expedite the search process. It assesses paths through the graph by minimizing the aggregate of the cost incurred to reach a node and an estimate of the cost to extend the path to the goal. Provided the heuristic is admissible and consistent, A* is optimal and complete. Its widespread utilization in various path-planning applications is attributable to its efficiency and optimality.

Algorithm 1 Dijkstra's Algorithm

```
procedure DIJKSTRA($Graph$, $source$)
    Create vertex set $Q$
    for each vertex $v$ in $Graph$ do
        $dist[v] \leftarrow \infty$
        $prev[v] \leftarrow$ UNDEFINED
        $Q$.add($v$)
    end for
    $dist[source] \leftarrow 0$
    while $Q$ is not empty do
        $u \leftarrow$ vertex in $Q$ with min $dist[u]$
        Remove $u$ from $Q$
        for each neighbor $v$ of $u$ still in $Q$ do
            $alt \leftarrow dist[u] + length(u,v)$
            if $alt < dist[v]$ then
                $dist[v] \leftarrow alt$
                $prev[v] \leftarrow u$
            end if
        end for
    end while
    return $dist[], prev[]$
end procedure
```

Understanding Dijkstra's and A* algorithms is instrumental as they serve as the building blocks for many advanced path planning algorithms, including those in the domain of Multi-Agent Path Finding (MAPF). The discourse herein is a precursor to the intricate dynamics of pathfinding and coordination in multi-robot systems, setting the stage for a more profound exploration of MAPF models and algorithms in the subsequent sections.

2.3 Multi-Agent Path Finding (MAPF) and Its Variants

Multi-Agent Path Finding (MAPF) is a well-established problem in artificial intelligence and robotics, aiming to find collision-free paths for multiple agents from their respective start positions to their respective goal positions in a shared environment. The classic formulation of MAPF and its variants address different aspects

Algorithm 2 A* Algorithm

```
procedure ASTAR($Graph$, $start$, $goal$)
    Initialize open list with $start$
    Initialize closed list to be empty
    $g[start] \leftarrow 0$
    $f[start] \leftarrow g[start] + h(start)$
    while open list is not empty do
        $current \leftarrow$ the node in open list having the lowest $f[]$ score
        if $current = goal$ then
            return reconstructed path from $start$ to $goal$
        end if
        Remove $current$ from open list
        Add $current$ to closed list
        for each neighbor $neighbor$ of $current$ do
            if $neighbor$ is in closed list then
                continue to next neighbor
            end if
            $tentative_g \leftarrow g[current] + dist(current, neighbor)$
            if $neighbor$ not in open list then
                Add $neighbor$ to open list
            else if $tentative_g \geq g[neighbor]$ then
                continue to next neighbor
            end if
            $g[neighbor] \leftarrow tentative_g$
            $f[neighbor] \leftarrow g[neighbor] + h(neighbor)$
            $prev[neighbor] \leftarrow current$
        end for
    end while
    return failure
end procedure
```

and challenges associated with multi-agent coordination and navigation.

2.3.1 Classic MAPF Formulation

The Multi-Agent Pathfinding (MAPF) problem is defined on a graph $G = (V,E)$, where V represents positions and E denotes transitions between them. Given agents $A = \{a_1,a_2,\ldots,a_n\}$, each with a start s_i and goal g_i, the objective is to find collision-free paths $P = \{p_1, p_2,\ldots, p_n\}$ connecting s_i to g_i for each agent, avoiding vertex collisions and edge swaps. A MAPF instance is considered solvable if there exists a set of collision-free paths for all agents from their start positions to their goal positions, this is also known as solvability of MAPF instance.

MAPF relates to the Pebble Motion problem [15], where pebbles move on a graph using a single free space to reach target configurations without collisions. MAPF extends this by allowing simultaneous movements on graphs with multiple free spaces and often focuses on optimizing metrics like total path cost or makespan, adding to its computational complexity.

On undirected graphs, solvability can be determined in polynomial time under certain conditions, such as sufficient free space, using algorithms like Push and Swap [16], which focus on feasibility rather than optimization. However, when optimizing objectives like minimizing total path length or makespan, MAPF becomes NP-hard, even on undirected graphs. For directed graphs, the problem is inherently more complex, as deciding solvability is NP-hard due to the restrictive nature of directed paths, which limit maneuverability and reversibility [17].

In summary, the classic MAPF formulation emphasizes finding feasible solutions, with complexity highly influenced by graph structure and optimization goals. Differences between undirected and directed graphs highlight the varying computational challenges across MAPF scenarios.

2.3.2 Conflict Types in MAPF

In the discussion of Multi-Agent Path Finding (MAPF), understanding the nature of conflicts that arise during agent navigation is central to appreciating the challenges inherent in solving MAPF problems. Conflicts in MAPF generally arise due to the shared use of space by multiple agents, leading to situations where two or more agents contend for the same spatial resource at the same time. Identifying and resolving these conflicts ensures collision-free paths and efficient navigation. The common types of conflicts in MAPF include [6]:

- **Vertex Conflicts:** Occur when two or more agents attempt to occupy the same vertex at the same time.

- **Edge Conflicts:** Arise when two or more agents attempt to traverse the same edge in opposite directions at the same time.

- **Swap Conflicts:** Occur when two agents attempt to swap their positions, requiring coordinated movement to avoid a collision.

- **Following Conflicts:** Arise when an agent follows another too closely, inhibiting the leading agent's ability to maneuver or backtrack.

- **Cycle Conflicts:** Involve a scenario where a group of agents forms a cycle, each blocking the other's path to lock them in a circular wait.

These types of conflicts, especially swap conflicts and cycle conflicts, are integral to understanding the challenges within MAPF and play a crucial role in the framework proposed in this dissertation. Identifying and efficiently resolving these conflicts is central to achieving effective multi-robot coordination and pathfinding, which is a core objective of the proposed framework. The nuanced handling of these conflicts, along with the extension and detailed discussion on conflict resolution strategies, will be further elaborated in the later sections of this dissertation.

2.3.3 MAPF Variants

Several variants of the classic MAPF formulation have been proposed to address different challenges or to extend the problem to new domains. Some of these variants include:

- **Multi-Agent Path Finding with Deadlines (MAPF-D):** In MAPF-D, each agent is assigned a deadline by which it must reach its goal. This variant addresses scenarios where timeliness is crucial [18].

- **Lifetime Multi-Agent Path Finding (L-MAPF):** In L-MAPF, the goal is to find paths for agents that maximize the time they remain operational (i.e., the time before they reach their goals and stop), which is particularly useful in scenarios where continuous operation is desired [19, 11].

- **Multi-Agent Pickup and Delivery (MAPD):** In MAPD, agents must pick up items from specified locations and deliver them to other locations, extending the problem to include handling of external objects [8].

- **Multi-Agent Path Finding with Capacity Constraints (MAPF-CC):** MAPF-CC extends the classic MAPF by introducing capacity constraints, where agents have limited capacities and must coordinate to collectively transport a set of items [20].

These variants reflect the rich landscape of problems that can be modeled as MAPF, each with unique challenges and requirements. Understanding these variants lays a foundation for developing a robust and flexible multi-robot control framework that addresses a broad spectrum of real-world multi-agent coordination and navigation challenges.

2.3.4 Algorithms for MAPF

The Multi-Agent Path Finding (MAPF) problem is known to be NP-hard, which entails that finding an optimal solution can be computationally intractable for large instances [17]. This has led to the development of various algorithms, each with its own trade-offs between optimality, completeness, and computational efficiency. These algorithms can be broadly categorized into exact and approximate algorithms and further distinguished based on centralized or decentralized approaches.

Exact Algorithms

Exact algorithms guarantee to find an optimal solution to the MAPF problem, differing in their approach and computational requirements. Notable exact algorithms include:

- **Conflict-Based Search (CBS):** CBS is a two-level search algorithm that independently plans paths for each agent and resolves conflicts by branching. The low-level search typically employs the A* algorithm to find paths avoiding conflicts, making it a leading method for solving the MAPF problem optimally [21].

- **A* Algorithm:** A* extends Dijkstra's algorithm by adding heuristics to enhance pathfinding efficiency. This heuristic component makes A* popular for pathfinding in single-agent systems. Although A* is primarily a general search method, it is applicable to MAPF due to its generality [10].

- **SAT-based approaches:** These model the MAPF problem as a satisfiability problem, framing the task in terms of logical propositions to find optimal solutions. Although computationally intensive, SAT-based methods provide a rigorous framework for exact solutions in MAPF scenarios [6].

Approximate Algorithms

Given the NP-hard nature of the MAPF problem, approximate algorithms aim to find near-optimal solutions in a more computationally efficient manner, trading off optimality for scalability. Some examples include:

- **Prioritized Planning:** Prioritized planning solves the MAPF problem by planning for one agent at a time, often employing A* for individual pathfinding, while treating earlier planned agents as moving obstacles [22, 23, 24].

- **Bounded Suboptimal Search:** Algorithms like Explicit Estimation Conflict-

Based-Search (EECBS), a bounded suboptimal variant of the Conflict-Based Search (CBS) algorithm. EECBS uses online learning to improve cost estimates and employs Explicit Estimation Search to select nodes, aiming to enhance performance while maintaining bounded suboptimality. [19]

Centralized versus Decentralized Approaches

Centralized approaches solve the MAPF problem in a single computational entity with access to all the information, while decentralized approaches distribute the computation among the agents with partial information.

- **Centralized Algorithms:** Algorithms like CBS and SAT formulations are examples of centralized approaches that provide a global solution to the MAPF problem.

- **Decentralized Algorithms:** These involve distinct planning for each robot with limited coordination. The paper [25] discusses solving path conflicts through negotiation among self-interested agents, and the paper [26] employs deep reinforcement learning for MAPF in decentralized settings.

This structured overview of MAPF algorithms highlights the range of strategies employed to tackle the inherent computational challenges of multi-robot coordination and pathfinding. The discussion sets the stage for introducing the proposed algorithms in this dissertation, emphasizing the motivation behind their development.

2.3.5 Limitations of MAPF Models and Existing Algorithms

The traditional MAPF models and most existing algorithms operate under idealized assumptions that often do not hold in real-world scenarios.

1. **Ignoring Kinematic Constraints:** Standard MAPF models typically assume that robots can instantaneously change their state, ignoring real-world robots' kinematic and dynamic constraints. This assumption can lead to infeasible solutions in practice, where robots cannot execute sharp turns or instantaneously alter their velocities [27] [28].

2. **Static Environment Assumption:** Many MAPF algorithms assume a static, known environment. However, real-world environments are dynamic and often unpredictable, especially in scenarios with human interaction or other moving entities. This necessitates algorithms capable of handling uncertainty and dynamically adapting to environmental changes [29].

3. **Single-Task Focus:** Traditional MAPF models often focus on single-task assignments for each robot. In contrast, real-world scenarios frequently involve robots executing a sequence of tasks over time (lifelong MAPF). While some algorithms, like the token-passing algorithm [8] [29] and the horizontal rolling collision resolution mechanism [13] address this issue, they still face challenges regarding computational efficiency and scalability.

4. **Computational Overhead:** Exact MAPF algorithms, while providing optimal solutions, suffer from high computational overhead due to the problem's NP-hard nature. Even approximate or suboptimal algorithms can require sig-

nificant computational resources, especially in large-scale or complex scenarios.

These limitations underscore the necessity for more robust, scalable, and realistic MAPF models and algorithms capable of handling the complexities of real-world multi-robot systems.

5. **Under-utilization of Individual Robot Capabilities:** Traditional MAPF models often operate under a centralized control paradigm, striving to orchestrate every action of the robots within the system. This centralized approach can under-utilize the capabilities of highly intelligent or autonomous robots, especially those endowed with advanced AI technologies that could enable more efficient or effective task performance based on local information available to individual robots.

6. **Inadequate Handling of Human-Robot Interaction:** In scenarios involving human-robot interaction, human behavior's unpredictability presents a significant challenge for traditional MAPF models and algorithms. These models often assume a static or predictable environment, which is seldom the case in human-centric settings. The dynamic nature of human actions and preferences necessitates more flexible and adaptive pathfinding and coordination algorithms that can respond in real time to the actions of human actors in the environment.

These additional limitations further accentuate the need for innovative frameworks and algorithms that can better harmonize with the capabilities of highly intelligent robots and the dynamic nature of human-robot interactive environments. The proposed framework in this dissertation seeks to address these and previously mentioned limitations by fostering a more adaptive and decentralized control paradigm and by incorporating mechanisms for real-time adaptation to dynamic human-robot interaction scenarios.

2.4 Comparative Analysis of Existing Literature

This section presents a comparative analysis of the key drawbacks associated with existing approaches in Multi-Agent Path Finding (MAPF). A succinct summary is provided to highlight the challenges and limitations inherent in these methods, particularly emphasizing that many are grounded in mathematical models that are often too idealized to be practically applicable.

- **Exact Algorithms (e.g., CBS, SAT-based):** Exact algorithms guarantee optimal solutions by formulating the MAPF problem through complex mathematical models. However, they suffer from high computational overhead, especially as the problem size grows. Additionally, these models assume static environments, ignore the kinematic constraints of real robots, and are often too rigid and impractical for real-world scenarios where conditions are dynamic and unpredictable.

- **Approximate Algorithms (e.g., Prioritized Planning, Bounded Suboptimal Search):** Approximate algorithms attempt to balance computational efficiency and solution quality by reducing the search space. However, they still operate within the state-time space, which is significantly larger than the purely state space used in single-agent pathfinding. This expanded search space leads to increased complexity and can still be computationally demanding. Consequently, while these algorithms offer scalability improvements, they often remain insufficiently robust and efficient in highly dynamic and uncertain real-world settings.

- **Classic MAPF Models and Variants:** Traditional MAPF models and their variants are based on idealized mathematical formulations, assuming complete and perfect knowledge of static environments. These models often focus on specific scenarios like single-task assignments or special constraints, limiting their general applicability. They fail to consider real-world factors such as dynamic changes, continuous task allocation, and robot kinematics, making them difficult to adapt and apply in practical, complex environments.

- **Centralized Control Approaches:** Centralized approaches rely heavily on global information and attempt to manage all agent decisions from a single control point. While this can provide a comprehensive coordination strategy, it often leads to computational bottlenecks and scalability issues. Moreover, centralized control does not effectively utilize the autonomous capabilities of individual robots, leading to inefficiencies.

- **Decentralized Control Approaches:** Decentralized methods distribute the computational load among agents, improving scalability and allowing for more autonomous operations. However, these approaches face challenges in coordinating agent actions without complete information, often resulting in suboptimal paths and increased conflicts. The limited communication between agents further complicates conflict resolution, particularly in tightly coordinated tasks.

- **Handling Human-Robot Interaction:** Most MAPF approaches assume environments with predictable dynamics, which falls short in scenarios involving human interaction. The unpredictability of human behavior poses significant challenges to these models, as they lack the real-time adaptability required to safely and effectively navigate in shared spaces with humans.

The above analysis underscores that while there have been significant advances in MAPF, the prevailing approaches are fundamentally constrained by their reliance on mathematical models that oversimplify real-world complexities. This necessitates the development of more adaptable and pragmatic frameworks that can bridge the gap between theoretical models and practical, real-world applications.

2.5 Motivation for a Balanced Framework

The inherent complexity and the multi-faceted nature of MAPF problems demand a balanced approach to achieve a harmonious trade-off among conflicting objectives such as optimality, efficiency, scalability, and real-world applicability. While the quest for a one-shot solution that excels in all dimensions remains elusive, a more pragmatic approach lies in developing a flexible framework capable of adapting to varying use case requirements and operational contexts.

2.5.1 Trade-offs in MAPF Solutions

The literature reveals a spectrum of MAPF algorithms, each excelling in different dimensions. Some prioritize optimality and provide exact solutions at the expense of computational efficiency, while others focus on scalability and real-time applicability and may deliver sub-optimal solutions. The ideal balance among these dimensions often depends on the specific use case, operational environment, and tolerance for sub-optimality.

2.5.2 Need for a Tunable Framework

A desirable avenue to navigate the trade-offs in MAPF is to design a tunable framework that allows for adjustments based on the prevailing operational requisites. Such a framework can embody a set of algorithms with varying degrees of optimality, efficiency, and scalability, enabling a tailored approach to different MAPF challenges.

- **Use Case Adaptability:** A tunable framework allows for the customization of algorithmic strategies based on a use case's specific needs. For instance, in mission-critical scenarios, the emphasis may be on optimality and reliability, while in large-scale, dynamic environments, efficiency and scalability may take precedence.

- **Operational Tuning:** By providing mechanisms for operational tuning, the framework can adapt to the evolving demands of the operational context, be it changes in robot density, environmental dynamics, or task complexity.

- **Benchmarking and Evaluation:** A flexible framework facilitates structured evaluation and benchmarking across various scenarios and algorithmic strategies, aiding in the objective assessment and continuous improvement of the MAPF solutions.

2.6 Reactive Approaches and Balancing Deliberation

Multi-agent pathfinding (MAPF) primarily embodies deliberative or planning-based approaches, where path planning is performed based on a complete or near-complete understanding of the environment and agent goals. However, reactive approaches often provide a viable alternative in dynamic and uncertain environments, enabling agents to respond to changes in real time.

2.6.1 Reactive Approaches in MAPF

Unlike their deliberative counterparts, reactive approaches do not require a priori complete knowledge of the environment or other agents' plans. They are characterized by real-time decision-making and responsiveness to environmental changes and agent interactions. Examples of reactive approaches include potential fields, swarm intelligence, and decentralized control strategies.

- **Advantages:** Reactive approaches often excel in dynamic, uncertain, or highly interactive environments due to their real-time decision-making capabilities. Due to their decentralized nature, they can also scale well to a large number of agents.

- **Disadvantages:** The lack of global planning and coordination can lead to sub-optimal solutions, local minima traps, and, in some cases, the inability to fulfill complex tasks or goals.

2.6.2 Balancing Deliberative and Reactive Approaches

The dichotomy between deliberative and reactive approaches presents a compelling motivation to explore a balanced framework that harnesses the strengths of both paradigms. Such a balanced framework could offer:

- **Real-Time Responsiveness:** Borrowing from reactive approaches to provide real-time adaptability to dynamic changes and agent interactions.

- **Structured Planning:** Leveraging the structured planning and optimality characteristics of deliberative approaches to ensure goal fulfillment and efficient resource utilization.

- **Scalability:** Ensuring the framework's ability to handle a growing number of agents and tasks without a prohibitive increase in computational complexity.

The endeavor to balance deliberative and reactive approaches forms a central theme of this dissertation. The proposed framework aims to meld real-time responsiveness with structured planning to tackle the diverse and nuanced challenges inherent in multi-agent pathfinding and coordination. The subsequent sections delve into this balanced framework's architectural and algorithmic design, along with an extensive evaluation of its performance across a spectrum of scenarios and operational contexts.

2.6.3 Aiming for a Balanced Approach

The subsequent sections of this dissertation introduce a novel framework designed with the above considerations in mind. The proposed framework strives for a balanced approach to MAPF, offering a suite of algorithms and tuning mechanisms to navigate the trade-offs inherent in multi-robot coordination and pathfinding. Through extensive evaluation across diverse scenarios, this work aims to demonstrate the adaptability and effectiveness of the proposed framework in addressing the nuanced challenges of real-world MAPF.

CHAPTER 3

CONCEPTUAL MODEL AND REPRESENTATION

This chapter provides essential information regarding the employed model and discusses the classical MAPF encoding.

3.1 Classical MAPF Representation

Before delving into the specific model for our problem, it is crucial to comprehensively understand the classical Multi-Agent Path Finding (MAPF) model. This section aims to offer a succinct yet formal description of the classical MAPF model, facilitating a shared terminology for further discussion.

3.1.1 Classical MAPF Problem Encoding

The classical MAPF model is defined on a directed graph

$$\mathscr{G} = (\mathscr{V}, \mathscr{E}), \tag{3.1}$$

where $\mathscr{V}$ represents a set of vertices corresponding to discrete locations within the environment, and $\mathscr{E}$ represents a set of edges denoting permissible transitions between these locations. In this model, each robot r is depicted as an agent with a unique starting vertex s_r and a goal vertex g_r on the graph $\mathscr{G}$.

The primary objective is to identify a set of non-colliding paths

$$\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\} \tag{3.2}$$

for n robots, where each path π_i is a sequence of vertices from s_{r_i} to g_{r_i}, ensuring that every robot reaches its designated goal without any collisions. Collision avoidance is guaranteed by ensuring that no two robots occupy the same vertex at any given timestep and no two robots simultaneously traverse the same edge in opposite directions.

3.1.2 Classical MAPF Problem Solution Encoding

A solution to the MAPF problem is encoded as a collection of paths, one for each robot, that satisfies the specified non-collision conditions. Although different variants may introduce subtle differences, the basic idea remains analogous to the classical formulation. Formally, a solution Σ is defined as a mapping:

$$\Sigma : \mathscr{R} \times \mathbb{T} \to \mathscr{V}, \tag{3.3}$$

where $\mathscr{R} = \{r_1, r_2, \ldots, r_n\}$ denotes the set of robots, $\mathbb{T} = \{0, 1, 2, \ldots, T\}$ denotes the set of discrete time steps with T being the makespan of the solution, and $\mathscr{V}$ denotes the set of vertices in graph $\mathscr{G}$. The makespan T is the total number of discrete time steps required to complete all robot tasks.

Each robot r is assigned a path $\pi_r : \mathbb{T} \to \mathscr{V}$ by the mapping Σ, such that $\pi_r(t)$

denotes the position of robot r at time t for all $t \in \mathbb{T}$ and $r \in \mathscr{R}$. The path π_r is subject to the initial and goal conditions:

$$\pi_r(0) = s_r, \quad \pi_r(T) = g_r. \tag{3.4}$$

Furthermore, the solution Σ adheres to the non-collision condition:

1. Vertex conflicts are avoided by ensuring that for any $t \in \mathbb{T}$, and any distinct robots $r_i, r_j \in \mathscr{R}$:

$$\pi_{r_i}(t) \neq \pi_{r_j}(t). \tag{3.5}$$

2. Edge conflicts are avoided by ensuring that for any $t \in \mathbb{T}$, and any distinct robots $r_i, r_j \in \mathscr{R}$, the condition

$$(e_{r_i}(t), e_{r_j}(t)) \neq (e_{r_j}(t), e_{r_i}(t)) \tag{3.6}$$

holds, where $e_{r_i}(t)$ denotes the edge being traversed by robot r_i at time t.

3.2 Sorting Center and Models

A sorting center exemplifies a common use case for multi-robot systems. Its hallmark characteristics encompass a structured environment, numerous mobile robots, and a substantial requirement for system throughput, thereby serving as an ideal reference for multi-robot control systems.

3.2.1 Environment Modeling

Typically, a multi-robot system sorting center or warehouse environment is modeled as a domain with designated pickup and drop-off points. The primary task of the robots is to execute pickup and delivery operations, as illustrated by the multi-robot pickup and delivery model. Moreover, these settings often exhibit the property that a robot is promptly assigned a new one upon completing a task.

3.2.2 Task Cycle and Performance Measurement

A complete pickup and delivery operation constitutes one cycle. A robot's objective is to complete as many cycles as possible; upon reaching a destination, a robot immediately receives a new task, initiating the pickup process anew. Like many MAPF variants, the environment is discretized into nodes or vertices, each representing a unique location, which can only be occupied by at most one stationary agent at any given time. If an agent is moving, it occupies two nodes, i.e., its currently occupying node and the heading node.

The primary performance metric is the maximum achievable throughput (Θ), defined as:

$$\Theta = \frac{\text{Number of completed cycles}}{\text{Unit of time}}, \tag{3.7}$$

irrespective of the number of robots employed; that is, any number of robots can be utilized as long as they contribute to higher throughput.

3.2.3 Illustrative Example

To elucidate the operational model, consider a simplified scenario with a single robot within a sorting center environment, as illustrated in Figure 3.1.

1. Initially, the robot is assigned a pickup task.

2. Upon reaching the designated pickup point, the robot immediately receives information about the corresponding drop-off point.

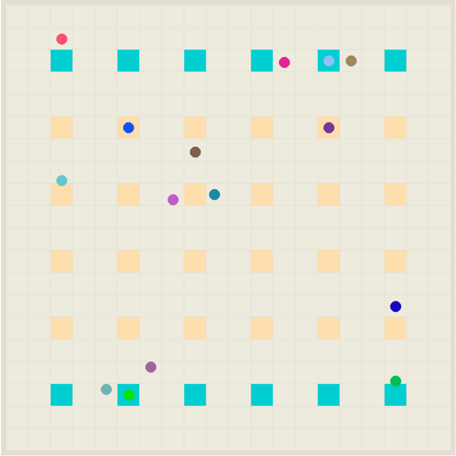


Figure 3.1: Blue square, yellow square, and colored circle represent pickup points, drop-off points, and robots, respectively.

3. The robot then travels to the drop-off point, adhering to the rule that it must not occupy the same node as another robot at any given time to prevent collisions.

4. Upon reaching the drop-off point, a cycle is completed. The robot is immediately assigned a new pickup task, marking the beginning of the next cycle.

This continuous cycle of tasks underscores the model's lifelong nature. Robots seamlessly transition between pickup and delivery tasks to achieve high system throughput.

3.3 Distinction Between Classical MAPF and Proposed Model

The proposed model departs from the classical Multi-Agent Path Finding (MAPF) framework in several notable dimensions tailored to address the unique operational dynamics and performance metrics inherent to a sorting center environment. This subsection delineates these distinctions and underscores the rationale behind the modifications employed in our model.

3.3.1 Dynamic Task Assignment

In the classical MAPF model, the goal destinations for all agents are predetermined and static throughout the problem instance. Conversely, our model espouses a dynamic task assignment paradigm, where robots are continually assigned new pickup and delivery tasks upon completing the preceding task. This dynamicity mirrors the perpetual task influx characteristic of sorting center operations and engenders a more realistic representation of the problem domain.

3.3.2 Throughput Optimization

The primary objective in classical MAPF centers around devising collision-free paths to ensure all robots reach their respective goals. In contrast, our model pivots towards maximizing system throughput (Θ), gauged by the number of completed pickup and delivery cycles per unit time. This shift in objective necessitates reevaluating solution methodologies to prioritize throughput enhancement, thus aligning with the operational exigencies of a sorting center.

3.3.3 Lifelong Path Finding

Classical MAPF largely operates within a finite-horizon framework, where the problem instance concludes upon all robots reaching their goal destinations. Our model, however, adopts a lifelong path-finding approach, where robots perpetually transition between tasks in a continuous cycle to bolster system throughput. This lifelong perspective cultivates a more nuanced understanding of performance metrics over an extended operational timeline, fostering a more robust and scalable solution framework.

3.3.4 Environment Structure

The classical MAPF model generally assumes a generic graph-based representation of the environment. Our model, on the other hand, encapsulates the structured yet dynamic nature of a sorting center, with designated pickup and drop-off points alongside transient congestion due to ongoing robot activities. This tailored environment model facilitates a more accurate reflection of real-world conditions and challenges prevalent in sorting center scenarios.

3.4 Introduction to Edge-to-Vertex Graph Representation

As outlined in the accompanying paper, the Edge-to-Vertex graph representation enhances the traditional graph model by integrating the rotational costs incurred by robots during transitions between vertices, thereby refining the fidelity of simulation environments. This section briefly introduces this representation and underlines its relevance to our proposed multi-robot system model within a sorting center setting.

3.4.1 Motivation for Edge-to-Vertex Transformation

Traditional graph representations may fail to capture rotational costs accurately, a shortfall more pronounced in complex scenarios where rotations markedly impact path efficiency and overall system throughput. The Edge-to-Vertex transformation addresses this limitation by revamping the graph structure to seamlessly encapsulate rotational costs.

3.4.2 Transformation Mechanics

This transformation transmutes the edges of the original graph into vertices in a new graph, and new edges are formed between these vertices based on the adjacency of the original edges. For example, edges (a,b) and (b,c) in the original graph lead to a new edge $((a,b),(b,c))$ in the transformed graph, facilitating a nuanced incorporation of rotational costs into the graph representation.

3.4.3 Cost Formulation

The cost associated with an edge in the transformed graph is depicted as:

$$\alpha \cdot c_1 + r, \tag{3.8}$$

where:

- c_1 denotes the node cost of the end node,
- r represents the rotation degree,
- α is a coefficient modulating the balance between the ordinary transition cost and the rotation penalty.

3.4.4 Cost Balancing and Optimization

The coefficient α provides a mechanism for balancing between minimizing rotation and other transitional costs. Adjusting the value of α enables tailored optimization based on operational priorities and environment specifics, thereby potentially enhancing the system's performance and throughput within the sorting center scenario.

3.4.5 Scope of Edge-to-Vertex Representation

The introduction of the Edge-to-Vertex graph representation here is solely for completeness. While there might be other methods to achieve similar outcomes and various ways to select the parameters, this paper will not delve into these as they fall outside the scope of the current discussion. The Edge-to-Vertex representation preserves the generalizability of the proposed framework, ensuring its broad applicability across a myriad of multi-robot system domains remains intact.

CHAPTER 4

CONCEPTUAL FRAMEWORK

This chapter presents a conceptual overview of the proposed framework. It eschews technical details to maintain clarity and simplicity and effectively elucidate the core idea.

4.1 Motivation

The preceding section outlined the Multi-Agent Path Finding (MAPF) model and the conventional encoding of its solutions. This section explores the limitations inherent to traditional MAPF models (and their variants), emphasizing the necessity for a more adaptable framework in light of advanced robotic capabilities.

4.1.1 Limitations of the Traditional MAPF Solution

Recalling, the standard solution encoding in MAPF involves paths of agents navigating through a location-state space enriched with a time dimension, which will henceforth be referred to as the *time-dependent solution*. This representation offers several advantages:

- Facilitating straightforward optimization measures, such as minimizing the total sum of time or the makespan.

- Providing a clear mathematical formulation, aiding algorithmic approaches.

Nevertheless, the model focuses on either the sum of time or the makespan, rendering finding an optimal and complete solution intractable due to the inherent complexity of these problems. An efficient solution to surmount these challenges remains elusive.

4.1.2 Misalignment with Real-World Scenarios

The conventional encoding approach misaligns with real-world scenarios. The primary discord emanates from the assumption of discrete time, which is unrealistic in real-world settings where robots operate under physical constraints like acceleration and deceleration, all within the continuum of time. Moreover, real-world environments are rarely perfect; robots might not execute tasks with exact precision, arriving at a position earlier or later than initially planned, especially in dynamic environments with unforeseen variables.

4.1.3 The Imperative for a Flexible Framework

Technological advancements have significantly enhanced the capabilities of individual robots, enabling them to execute tasks and manage unforeseen events reactively and autonomously. However, integrating such autonomous robots within the conventional MAPF model poses a significant challenge. Harnessing the synergy inherent in multi-robot systems becomes a complex endeavor without a coordinated framework. This research advocates for a more flexible framework to adeptly coordinate these robots and maximize their capabilities, especially as we transition towards more complex use cases in the near future.

4.2 Fundamentals of the Proposed Framework

The proposed framework's foundational concept stems from a central constraint intrinsic to multi-robot systems: no two robots can occupy the same position simultaneously. A simplistic strategy to mitigate this constraint is to reserve the nodes currently occupied by robots, thus mirroring the physical environment. This methodology epitomizes a reactive approach wherein robots navigate through the environment and address conflicts as they emerge.

Leveraging this concept, each robot is only required to plan individually, obviating the need for a time dimension in the planning phase. This simplification creates a naturally scalable, uncomplicated, and highly efficient algorithm. However, it becomes evident that conflict resolution could quickly burgeon into a critical issue without a robust coordination mechanism, posing significant challenges to enacting this elementary idea.

Consequently, an alternative proposition is to devise a time-independent plan—a foresighted strategy comprising a collection of paths for each agent. This time-independent plan will henceforth be referred to as the *plan* for brevity. The core essence of this proposition will be elucidated in Chapter 6. Subsequently, a time-dependent solution could be dynamically reconstructed based on the evolving state of the environment, aligning the theoretical model more closely with real-world dynamics.

4.2.1 Framework Structure Overview

We adopt the classical Multi-Agent Path Finding (MAPF) problem as our foundational model to foster clarity. In this model, a cadre of robots is assigned specific objectives, which they strive to accomplish without interference. This model will be extended to align with our research goals in subsequent discussions.

The proposed framework operates as follows:

1. **Plan Generation:**

   - A coherent plan, constituting a series of paths for the robots, is meticulously devised by a component termed the *plan generator*.

2. **Plan Reception by Resource Allocator:**

   - The devised plan is dispatched to another component termed the *resource allocator*, which is responsible for distributing the resources (nodes) among the robots based on the plan.

3. **Node Allocation:**

   - The *resource allocator* scrutinizes the plan to ensure:

     (a) Nodes are sequenced to robots aligning with each robot's designated path.

    (b) No node is concurrently allocated to multiple robots.

    (c) A vacated node can be reallocated as necessitated.

4. **Node Occupation and Release:**

    - A robot is permitted to occupy a node only if allocated by the *resource allocator* and is obliged to relinquish nodes to the *resource allocator* post utilization.

4.2.2 Framework Operational Procedure

To streamline operations, two ideal assumptions are posited, to be elaborated in subsequent sections:

1. A *plan generator* is postulated, adept at crafting a *satisfiable* plan, contingent on its feasibility.

    - Herein, *satisfiable* denotes the availability of at least one allocation sequence enabling all robots to fulfill their objectives.

2. An efficient *resource allocator* is presupposed, proficient in consistently identifying these allocation sequences, ensuring an unimpeded progression of the robots towards their objectives.

The procedure to address the MAPF challenge is delineated as follows:

1. The *plan generator* formulates a plan.

2. This plan is relayed to the *resource allocator*.

3. The *resource allocator* discerns an optimal allocation sequence:

    - Resources are allocated to robots to facilitate their one-step progression toward their objectives.

    - Upon a robot's advancement, the previously occupied resource is vacated.

    - The *resource allocator* may reassign this resource to another robot if deemed necessary.

    - This iterative process continues until every robot reaches its designated destination.

This procedure epitomizes a systematic strategy for the MAPF challenge, encompassing plan generation, resource allocation, and dynamic resource management. It ensures goal realization for all robots.

4.3 Framework Interpretation

The proposed framework can be envisaged as a sophisticated coordination mechanism or, alternatively, a robust algorithm. This dual characterization accentuates the framework's myriad advantages, enabling seamless operation as a system while opening many avenues for algorithmic exploration, analysis, and enhancement.

As a system, the framework orchestrates multi-robot operations through structured components like the *plan generator* and the *resource allocator*. It adeptly accommodates real-world dynamics by enabling real-time resource allocation, minimizing conflicts, and ensuring the successful realization of goals for each robot.

On the other hand, the algorithmic interpretation manifests as a two-tier MAPF algorithm. The initial tier is responsible for constructing the path for each robot, while the subsequent tier determines the timing for each robot to reach its designated positions within the path sequence.

4.4 Framework Flexibility

While designed as a standalone coordination system, the proposed framework exhibits seamless interoperability with existing MAPF algorithms, facilitating easy integration or extension with other components. This framework is not only straightforward to implement but also underscores flexibility and ensures safety for the robots, attributable to its strict allocation constraints. Specifically, a resource can only be reallocated to another robot once it has been released, a constraint that can be relaxed if necessary, although such relaxation falls outside the purview of this research.

This framework can function effectively as a coordination layer for robots with reactive capabilities. For instance, robots can individually plan their paths and control their speeds autonomously, provided they adhere to the common rule of only accessing nodes or positions allocated to them. The dynamic allocation strategy inherent in this framework seamlessly accommodates changes in individual paths or the addition of new tasks.

Moreover, the framework's compatibility with existing MAPF algorithms extends its flexibility in handling unexpected events. This compatibility can be achieved by reducing a time-dependent solution to a plan accepted by the *resource allocator*. The time-dependent solution can also be valuable for more efficient resource allocation.

4.4.1 Future Work

The proposed framework opens doors for various future explorations. While this research lays the foundation, subsequent studies could delve into relaxing allocation constraints, integrating real-time communication protocols among robots, or exploring the framework's application in heterogeneous robotic systems with varying capabilities.

CHAPTER 5

PLAN GENERATOR AND ALLOCATORS

In the preceding chapters, we laid the groundwork for our framework to address the Multi-Agent Path Finding (MAPF) problem, highlighting the critical roles of the *Plan Generator* and the *Resource Allocator*. This chapter delves deeper into these components, elucidating their functionalities and importance.

5.1 Revisiting Our Assumptions

Previously, we introduced assumptions for conceptual clarity. Here, we revisit these assumptions in a more concise manner:

1. **Plan Generator:**

    - Assumes the existence of a *Plan Generator* capable of creating a satisfiable plan, where satisfiable implies feasibility in terms of robots' initial positions and goals.

2. **Resource Allocator:**

    - Presumes an efficient *Resource Allocator* for optimal node allocation, ensuring conflict-free progression of robots towards their objectives.

These assumptions, while idealized, simplify the framework's conceptual understanding. They represent optimal conditions but may be challenging to achieve in practical scenarios, especially considering the complexity and scalability demands of MAPF.

5.2 A Different Perspective on Plan Generation and Resource Allocation

Before we proceed, let's consider the Plan Generator and the Resource Allocator from a different perspective. This approach allows us to appreciate the nuances of our two-stage solution in the context of the Multi-Agent Path Finding (MAPF) framework.

1. **Plan Generator - Pathfinding Without Time:** The Plan Generator's role is to create collision-free paths for each robot. This stage mirrors standard MAPF algorithms by omitting the time dimension and focusing solely on spatial pathfinding. This simplifies the task, making it more manageable.

2. **Resource Allocator - Adding Time Back:** The Resource Allocator reintroduces the time dimension. It effectively schedules the movements of robots based on the paths generated in the first stage, ensuring conflict-free movement and adherence to the schedule.

This method divides the MAPF problem into more manageable parts, with the first stage focused on pathfinding and the second on temporal coordination. This division is particularly suitable for scenarios where immediate time constraints are not a primary concern.

5.3 Independent Path Planning for Each Robot

As part of our approach to solving the MAPF problem, we implement a strategy where each robot's path is planned independently using standard shortest-path algorithms. This method treats the pathfinding for each robot as an individual task. Important aspects of this approach include:

- **Low Robot Density and Marginal Benefit:** Lower robot density in environments decreases the likelihood of generating unsatisfiable paths. The marginal benefit of adding more robots also diminishes, supporting the viability of this method.

- **Environment Layout and Traffic Control:** Strategic design of the environment layout and effective traffic control measures reduce the risk of unsatisfiable path creation.

- **Backup Resolution Mechanisms:** Including backup mechanisms like a timeout process helps resolve potential conflicts or unsatisfiable paths.

This approach, combining lower robot density with strategic planning and backup mechanisms, offers a practical solution for our MAPF framework. It balances path generation simplicity with the need to manage multi-robot system complexities.

5.4 Enhancing the Strategy for the Resource Allocator

In our Multi-Agent Path Finding (MAPF) framework, the strategy employed by the Resource Allocator is crucial for efficiently managing robot interactions and ensuring effective pathfinding. While the initial strategy is focused on preventing collisions, we must consider and address more complex scenarios to enhance the allocator's effectiveness.

5.4.1 Basic Resource Allocation Strategy

Initially, the Resource Allocator operates on a fundamental principle: it allows robots to access resources (nodes) unless such access would result in two robots occupying the same position at the same time. While effective in preventing direct conflicts, this basic collision avoidance strategy does not guarantee that all robots can successfully reach their destinations.

5.4.2 Advanced Allocation Strategy

In complex scenarios, our Resource Allocator not only anticipates indirect conflicts but also inevitable direct conflicts. Consider the following example:

- Robot 1 is scheduled to move from point A to B and then to C.
- Robot 2 is set to move from point D to C and then to B.

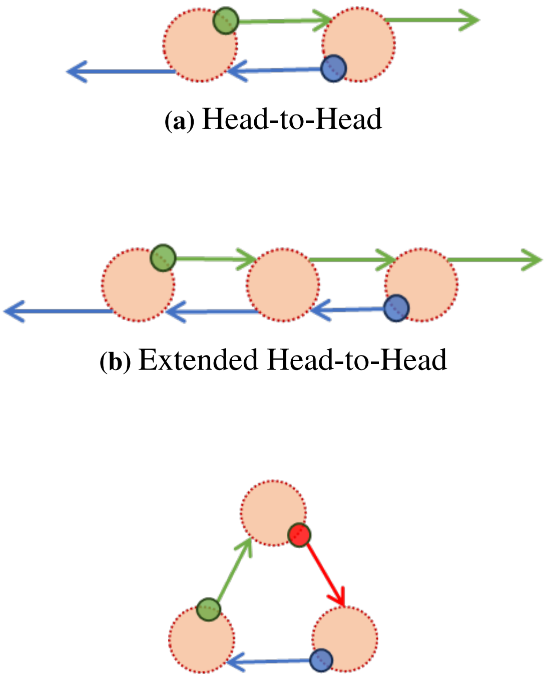
At first glance, this scenario seems free of direct conflicts. However, a closer analysis reveals that a direct conflict is unavoidable. The path of Robot 1 from B to C will eventually intersect with Robot 2's path from D to C. Subsequently, both robots are scheduled to move to point B, leading to a direct path conflict. In such cases, the Resource Allocator must identify these inevitable direct conflicts arising from path interdependencies and strategize to mitigate them, ensuring efficient and collision-free navigation for all robots.

5.4.3 Pattern Matching Resource Allocator

Our multi-agent pathfinding framework introduces the *Pattern Matching Resource Allocator* to address complex scenarios efficiently. This innovative allocator extends beyond conflict resolution by identifying a spectrum of potential deadlock scenarios through predefined patterns indicative of unsolvable path overlaps. These patterns are designed to be abstract yet adaptable, meeting varied scenarios' specific dynamics and requirements.

One illustrative example involves the indirect interference between two agents' paths, showcasing a typical deadlock pattern. The Pattern Matching Resource Allocator is engineered to recognize and mitigate this pattern and a variety of others potentially leading to deadlocks. For instance, it can detect cyclic dependency chains involving multiple agents, which is crucial in scenarios where simultaneous movement resolutions are infeasible.

Figures 5.1a, 5.1b, and 5.1c showcase the primary patterns addressed in this dissertation, with a detailed examination to follow in subsequent sections. In these illustrations, small circles represent agents, the large circles with the dotted border represent nodes, and arrows of corresponding colors depict their paths.



(a) Head-to-Head

(b) Extended Head-to-Head

(c) Direct Cyclic Dependency

Figure 5.1: Primary patterns addressed in the dissertation.

5.4.4 Sliding Windows and Iterative Resource Allocator

In our Multi-Agent Path Finding (MAPF) framework, integrating a proficient resource allocator and a satisfiable plan enables the dynamic reconstruction of MAPF solutions. Implementing a sliding windows approach, which enhances the allocator's efficiency and adaptability, is key to achieving this.

Sliding Windows Concept: The sliding windows approach involves setting a window size, denoted as k, for each robot. This window size dictates the maximum number of resources a robot can reserve at any given time, such as grid cells. As a resource is released from a reservation, it becomes available for allocation to other robots. This method is particularly effective in environments where continuous paths are beneficial for maintaining higher robot speeds.

Customizable Window Sizes: One of the notable advantages of this approach is its flexibility. The window size *k* can vary for different robots, accommodating diverse operational characteristics like varying acceleration and movement speeds. This adaptability allows the allocator to cater to each robot's specific requirements, enhancing the system's overall efficiency and responsiveness.

Iterative Allocation Process: The resource allocator operates iteratively, reassessing and adjusting resource allocations as the robots progress and release resources. This iterative process ensures optimal resource utilization and conflict-free navigation, even in dynamic and complex scenarios.

5.5 Algorithmic Models

The discussion thus far has provided a high-level overview of the plan generator and resource allocators. While this offers a broad understanding of the concepts, it is imperative to delve into the specifics to avoid ambiguity in further analysis and discussion.

The next chapter will establish a theoretical and algorithmic foundation for the concepts discussed so far, e.g., the definition of *allocator* and *satisfiability*.

This approach will ensure that the theoretical underpinnings are solidly established, providing a robust platform for practical application and further enhancing the multi-robot control framework.

CHAPTER 6

ALGORITHMIC MODEL

6.1 Preliminaries

Consider a group of m agents navigating a directed graph $G = (V, E)$. Each agent is denoted by a_i, where i is an index ranging from 1 to m. Each agent a_i is assigned a unique path γ_i in the set of all possible paths Γ, represented as

$$\gamma_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,|\gamma_i|}\}, \tag{6.1}$$

where $v_{i,1}$ and $v_{i,|\gamma_i|}$ are the starting and goal nodes for agent a_i, respectively.

The path γ_i consists of a sequence of vertices in G, and its length is denoted by $|\gamma_i|$. In accordance with common multi-agent path-finding formulations, for any two distinct agents a_i and a_j (where $i \neq j$), their starting nodes must be distinct, i.e., $v_{i,1} \neq v_{j,1}$.

6.1.1 Definition of a Plan

Let us denote a ***Plan*** as Π. A Plan Π within this context refers to a set of paths $\{\gamma_1, \gamma_2, \ldots, \gamma_m\}$ for all agents $\{a_1, a_2, \ldots, a_m\}$ that satisfies the following criteria:

- Each path γ_i adheres to the graph structure of G and follows the sequence of vertices from the starting node $v_{i,1}$ to the goal node $v_{i,|\gamma_i|}$.

- Each path γ_i in the Plan must contain at least one vertex, ensuring non-emptiness and consistency. Formally, for each path, $|\gamma_i| \geq 1$.

- The starting nodes for all agents are distinct, i.e., $v_{i,1} \neq v_{j,1}$ for all $i \neq j$.

- For any path γ_i, every pair of consecutive vertices $v_{i,j}$, and $v_{i,j+1}$ within the path must be distinct. Formally, for every i and for every $j < |\gamma_i|$, $v_{i,j} \neq v_{i,j+1}$.

This definition ensures that each Plan meaningfully represents the agents' positions and objectives within graph G while adhering to the system's specific constraints. The last constraint ensures that each step in a path moves to a new vertex, prohibiting the agent from remaining stationary or revisiting the same vertex in consecutive steps.

6.1.2 Definition of a Target Plan

A **Target Plan** Π_T represents a set of plans where, for any given plan within Π_T, every agent's path consists solely of its goal position, implying no movement is necessary for the agent to achieve its objective. This concept reflects situations

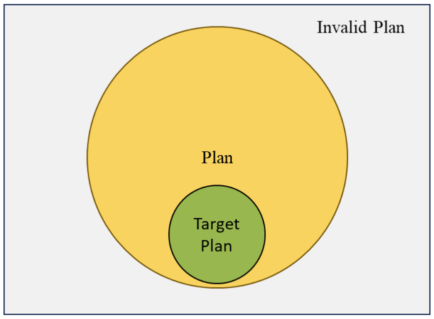


Figure 6.1: Target Plans within Valid Plan Space, Excluding Invalids

where agents are already positioned at their intended destinations at the start of the plan. Formally, for any plan within Π_T and for every agent a_i in that plan, the path γ_i is defined as:

$$\gamma_i = \{v_{i,1}\} \quad \text{where} \quad |\gamma_i| = 1, \tag{6.2}$$

indicating that $v_{i,1}$ is both the starting and goal node for agent a_i. Consequently, this definition underscores that each path within any plan in Π_T comprises exactly one vertex, underscoring the agents' stationary status at their respective goal positions.

We define the Target Plan as a subset of the Plan to create a unified framework that provides flexibility in defining *operators*, which will soon be discussed. This approach allows more flexible state transformations; for instance, an action that changes a robot's destination can be easily captured without breaking our model.

6.2 Assignment and State Transition

The concept of an ***assignment*** is fundamental to understanding the dynamic evolution of system states. It represents a process through which a given state, or a *Plan* in our context, is transformed into another state. This transformation reflects changes in the system, such as the movement of an agent.

6.2.1 Definition of Assignment

An *assignment* is a function or operation that maps one state to another, representing the transition from one snapshot of the environment to the next. In the context of our analysis, where states are represented as Plans, an assignment is a process that, for example, can modify a Plan based on the movement of an agent within graph G.

$$\text{Assignment: } \Pi \to \Pi' \tag{6.3}$$

where Π is the original Plan, and Π' is the new Plan resulting from the assignment.

6.2.2 Assignment Operators Set

Let us denote the set of all possible assignment operators as $\mathscr{A}$. Each element in $\mathscr{A}$ is an operator that can transform a Plan Π into a different Plan Π'.

$$\mathscr{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \tag{6.4}$$

where each α_i is a distinct assignment operator within the set.

6.2.3 Definition of Assignment Operator

The ***assignment operator*** is a specific form of assignment that defines the rules or mechanisms through which a state is transformed. This operator takes into account the factors that influence the state transition, such as the agents' actions, environmental changes, or system rules.

$$\text{Assignment Operator: } \alpha(\Pi) = \Pi' \tag{6.5}$$

where α represents a specific assignment operator in $\mathscr{A}$, transforming an existing Plan Π into a new Plan Π'.

This framework of a set of assignment operators allows for a more comprehensive analysis and modeling of system dynamics. It provides the flexibility to apply different transformation rules under various scenarios.

6.2.4 Basic Operations

Let $\mathscr{B} = \{\beta_1, \beta_2, \ldots, \beta_k\}$ be a subset of $\mathscr{A}$, called basic operations, and k is the number of agents. Each basic operation β_i in $\mathscr{B}$ is a function that transforms a Plan Π into a new Plan Π' by moving the path of a specific agent a_i forward by one step:

$$\Pi' = \beta_i(\Pi) = \begin{cases} \gamma_i' = \{v_{i,2}, v_{i,3}, \ldots, v_{i,|\gamma_i|}\}, & \text{if move is valid for agent } a_i \\ \gamma_i' = \gamma_i, & \text{otherwise} \end{cases} \tag{6.6}$$

where γ_i is the current path for agent a_i in Plan Π, and γ_i' is the updated path in the new Plan Π'. The operation removes the current starting node $v_{i,1}$ and treats $v_{i,2}$ as the new starting node for agent a_i, effectively moving the path of agent a_i by one step, given that the resulting Plan Π' is still a valid plan.

If the path γ_i contains only one node (i.e., the agent has reached its goal) or if the transformation would result in an invalid Plan (not a Plan), then the path remains unchanged.

6.3 Dependency and Rotation Operations

This section introduces and formalizes the notions of direct and cyclic dependency and a specialized operation known as the rotation operation, which is essential for resolving potential deadlock situations arising from these dependencies.

6.3.1 Direct Dependency

In our framework, direct dependency refers to a situation where the immediate next movement of one agent is contingent upon the movement of another agent. This occurs when the next intended position of one agent is currently occupied by another agent. We denote a direct dependency of agent a_i on agent a_j as $a_i \rightarrow a_j$, indicating that a_i's next move depends on a_j's movement.

$$a_i \rightarrow a_j \quad \text{if and only if } v_{i,2} = v_{j,1} \text{ for agents } a_i \text{ and } a_j \tag{6.7}$$

6.3.2 Direct Cyclic Dependency

Direct cyclic dependency represents a condition where a sequence of agents depend on each other in a closed loop, creating a deadlock scenario. Each agent in the loop is poised to move to a node currently occupied by the next agent in the cycle. This scenario necessitates an intervention, such as the rotation operation ρ, which will be discussed shortly, to resolve the deadlock and allow the agents to progress toward their goals.

$$a_i \rightarrow a_j \rightarrow \ldots \rightarrow a_i \quad \text{forms a direct cyclic dependency} \tag{6.8}$$

6.3.3 Rotation Operation

The rotation operation, denoted as ρ, is crucial for resolving deadlocks involving more than two agents in direct cyclic dependencies. Alternative strategies must be employed for cycles with two agents, as ρ ensures synchronous movement of all agents to their next positions, resolving the deadlock and enabling progression toward their goals, which is not applicable to cycles that involve only two agents.

$$\text{Rotation Operation: } \rho(\Pi) = \Pi' \tag{6.9}$$

Under ρ, the paths of agents involved in a cyclic dependency with more than two agents are updated as follows:

$$\Pi' = \rho(\Pi) = \begin{cases} \gamma_i' = \{v_{i,2}, \ldots, v_{i,|\gamma_i|}\}, & \text{if } a_i \text{ in cyclic dependency} \\ \gamma_i' = \gamma_i, & \text{otherwise} \end{cases} \tag{6.10}$$

6.4 Solution Sequence and Plan Satisfiability

This section introduces the concepts of a *solution sequence* and *plan satisfiability*. These notions are pivotal in determining the feasibility and execution of plans within our framework. We explore how specific operations sequences can transform an initial plan into a target configuration and define criteria to evaluate whether such transformations are achievable under given constraints.

6.4.1 Definition of a Solution Sequence

A ***solution sequence*** is a sequence of operations applied to an initial Plan Π, with the aim of transforming it into a plan that meets the criteria of a Target Plan. Given a set of assignment operators $\mathscr{A}$, a solution sequence details the required steps to navigate agents from their starting positions to their goal positions within the system's constraints.

The solution sequence is expressed as:

$$\text{Solution Sequence: } \sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \tag{6.11}$$

such that applying this sequence to Π results in a Plan that belongs to the set of all possible Target Plans Γ_T, defined by the system's constraints. Formally, this is represented as:

$$\alpha_n(\ldots(\alpha_2(\alpha_1(\Pi)))\ldots) \in \Gamma_T \tag{6.12}$$

6.4.2 Plan Satisfiability

Plan satisfiability evaluates whether a given Plan Π can be transformed into a Target Plan under the operations in $\mathscr{A}$. This binary classification assesses the attainability of end states in the framework.

Satisfiable Plan A plan Π is considered **satisfiable** if there exists a sequence of operations σ that transforms Π into a plan belonging to the set of all possible Target Plans Γ_T. In other words, a satisfiable plan allows the agents to reach their respective goals without any conflicts, making it possible to find a solution sequence that meets all system constraints.

Unsatisfiable Plan Conversely, a plan Π is deemed **unsatisfiable** if no sequence of operations σ can transform Π into a plan within the set Γ_T. An unsatisfiable plan implies that, due to conflicts such as cyclic dependencies or blockages, it is impossible to achieve the target configuration, regardless of the operations attempted.

Formally:

$$\text{Plan } \Pi \text{ is satisfiable } \leftrightarrow \exists\sigma \text{ such that } \sigma(\Pi) \in \Gamma_T \tag{6.13}$$

$$\text{Plan } \Pi \text{ is unsatisfiable } \leftrightarrow \nexists\sigma \text{ such that } \sigma(\Pi) \in \Gamma_T \tag{6.14}$$

These definitions of a solution sequence and plan satisfiability provide a clear framework for determining the achievability of goals within the multi-agent control framework in this research and form the basis for further algorithmic development and analysis.

6.4.3 Characterization of Unsatisfiable Plans

Every unsatisfiable plan must eventually lead to one of the following scenarios, irrespective of the sequence of assignments applied:

1. A path is obstructed by an agent already reaching its destination.

2. The formation of a direct cyclic dependency involving two or more agents, where no alternation or rotation movements can resolve the situation.

Consider an unsatisfiable plan Π in a multi-agent navigation context. By definition, no sequence of assignment operations σ can transform Π into a target plan Π_T under the system's constraints.

As we progress the agents through a sequence of assignments from the initial state defined by Π, and since Π is unsatisfiable, at least one agent, denote this agent as a_i, cannot reach its goal due to blockages. These impediments emerge in two main forms:

Destination Blockage: Agent a_i is hindered by another agent a_j, where a_j has reached its destination. This scenario matches the first condition, where a_i's path is directly obstructed by a_j, who occupies the required position and has no further actions to execute.

Cyclic Dependencies: If agent a_i is blocked, and not because another agent has reached its goal, we examine the sequence of blockages. This examination outlines a sequence $a_i \to a_j \to a_k \to \ldots$. This sequence leads to:

a. A cyclic dependency, where the sequence eventually loops back (e.g., $a_j \to a_k \to \ldots \to a_j$). This cycle may directly involve a_i or be independent of a_i, yet still result in a_i's blockade, fulfilling the second condition.

b. A sequence that doesn't form a loop but concludes with an agent obstructed by another that has attained its designated endpoint, thus returning to the first scenario.

If any agent within a blocking sequence could move, it would negate our assumption of unsatisfiability, as this action would relieve at least one blockage. Therefore, under the presumption of an unsatisfiable plan Π, the system must invariably lead to a state where no agent can advance, encompassed by the outlined conditions.

6.5 Satisfiability in Uni-directional Graphs

Consider a directed graph $G = (V,E)$ with a special property: for any pair of vertices $v_1, v_2 \in V$, if there exists an edge from v_1 to v_2 (denoted as $(v_1, v_2) \in E$), then there must not exist an edge from v_2 to v_1 (i.e., $(v_2, v_1) \notin E$). This property prevents the formation of bidirectional edges, inherently eliminating two-agent cycles. We named it uni-directional graphs.

Under these conditions, combined with the allowed rotation operator ρ and arbitrary movement post-destination, we can argue that any plan Π generated within this graph structure is satisfiable.

From our earlier discussions, we know that an unsatisfiable plan necessitates a direct cyclic dependency leading to deadlock or a blockage by agents reaching their destinations. In the context of G, the absence of bidirectional edges eliminates 2-agent cycles. Although cycles involving three or more agents could theoretically still occur, these multi-agent cycles can be resolved by applying the rotation operator ρ.

Consider an unsatisfiable plan Π in G, assuming it leads to a deadlock scenario due to a cyclic dependency among agents $a_i, a_j, ..., a_n$. Since G allows for no reciprocal edges, any such cycle cannot result from back-and-forth paths but must instead form a larger loop involving three or more nodes.

By the definition of our rotation operator ρ, in any cycle involving three or more agents, applying ρ permits a simultaneous shift in positions along the cycle, effectively breaking the deadlock condition without violating the directed nature of

G. Moreover, since agents can move arbitrarily post-reaching their destinations, any potential blockages caused by agents at their goal nodes can be resolved, ensuring these agents don't contribute to unsatisfiable conditions.

Therefore, given the structural constraints of G and the operational rules (including the rotation operator ρ and post-destination movements), any plan Π within this context is inherently satisfiable, as mechanisms exist to resolve all potential deadlock scenarios.

In conclusion, under the stipulated conditions, all plans generated in the unidirectional graph G are satisfiable, reinforcing the critical impact of structural graph properties and operational freedoms on resolving multi-agent pathfinding challenges.

6.6 Plan Simplification

Agents often have lengthy paths when deployed in large and complex environments, leading to substantial plans. Managing these extensive plans can impose significant computational demands, depending on the architecture of the implementing framework. Importantly, when the ratio of agents to unique nodes is low, indicating a sparse plan scenario, it's possible to simplify these plans. To facilitate our discussion, paths will be represented using string notation. For instance, $\gamma_i : ABC$ signifies that agent i traverses from node A to node C via node B.

Let's outline the simplification process:

1. Track the occurrence count of each node across all paths.

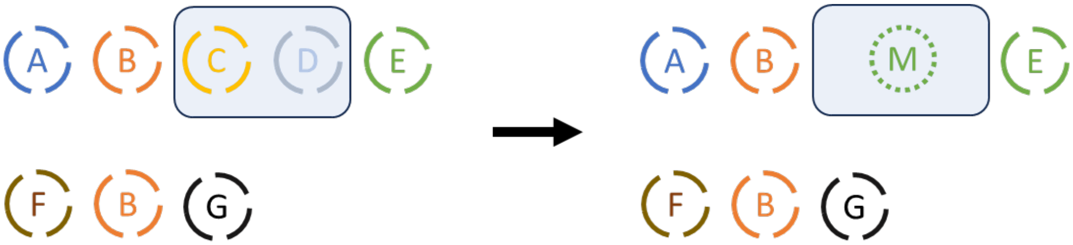


Figure 6.2: Example of Plan Simplification

2. Identify the first and last nodes for every path.

3. For each path, if it contains a sequence of more than two consecutive nodes that only occurs once (indicating these nodes are exclusively used by this agent), replace this sequence with a virtual node.

For example, consider a plan consisting of $\gamma_1 : ABCDE$ and $\gamma_2 : FBG$, as shown in Figure 6.2. This can be simplified to $\gamma_1 : ABME$ and $\gamma_2 : FBG$, where M is a virtual node representing the sequence CD. This approach is based on the rationale that if an agent can reach the first node in a sequence of exclusively used nodes, it can subsequently reach the rest. Thus, if a simplified plan is satisfiable, the original plan is also satisfiable, and vice versa. This property can significantly reduce the number of assignments that need to be verified, enhancing computational efficiency.

6.7 Conceptual Framework and Algorithmic Model

Now that we have a clear understanding of both the conceptual framework and the algorithmic model, we can seamlessly integrate these components.

A conceptual framework comprises two essential parts:

1. An algorithm for generating or updating the *Plan*.

2. An allocator that determines the order for applying operators.

Ideally, the selected allocator should maintain an initially satisfiable plan throughout every transformation. Furthermore, we may need to refine the model and select suitable operators to better align it with our specific use cases.

6.8 Conclusion

In this chapter, we meticulously outlined the foundational algorithmic model that underpins our multi-agent navigation framework within directed graph environments. By establishing precise definitions for *Plans*, *Target Plans*, and the operational mechanics behind *Assignment Operations*, we have laid down a comprehensive groundwork that not only delineates agent paths and objectives but also integrates dynamic elements such as agent dependencies and state transitions.

By introducing *Solution Sequences* and *Plan Satisfiability*, we provided a structured approach to assess the feasibility and execution of agent plans, reinforcing the framework's capability to address complex navigational challenges. The integration of the *rotation operation* concept, particularly in the context of uni-directional graphs, exemplifies our approach to resolving potential deadlock scenarios, thereby enhancing the robustness of agent coordination mechanisms.

Furthermore, our exploration into the characteristics of unsatisfiable plans underpins the critical importance of algorithmic adaptability and strategic operational planning in avoiding deadlock conditions and ensuring the successful execution of agent objectives.

This model is vital for our research's subsequent development and analysis. It offers a scalable, adaptable framework that addresses current multi-agent system challenges and lays the foundation for future innovations and strategies in complex system navigation and coordination.

Moving forward, the principles and methodologies delineated in this chapter will guide the implementation of advanced algorithmic strategies, propelling our framework towards solving increasingly sophisticated problems in multi-agent systems. This endeavor not only broadens the horizon of algorithmic research but also contributes significantly to the practical applications and theoretical advancements in the field of multi-agent pathfinding.

CHAPTER 7

FRAMEWORK REALIZATION

As previously described, this chapter is dedicated to implementing our conceptual framework within a simulated multi-robot system. We revisit the scenario of a homogeneous fleet of robots navigating a grid environment, assigned to dynamically move between pickup and drop-off points, as detailed in Section 3.2. The grid's layout facilitates dynamic and unrestricted route planning.

The primary objective is maximizing throughput, which is defined as the total number of tasks completed within a predetermined timeframe. The model operates under the assumption of an endless task supply and supports the integration of as many robots as necessary as long as they contribute to an increase in overall throughput.

To adapt our framework to this scenario, we focus on customizing the algorithmic model to manage dynamic task assignments and optimize robot coordination efficiently. This is key to effectively utilizing grid space while avoiding collisions and congestion. The following sections outline strategies for integrating these components to enhance system efficiency.

7.1 Framework Implementation Strategies

This section discusses core strategies for implementing our framework, focusing on dynamic plan updates, operator selection, and pattern monitoring.

7.1.1 Dynamic Plan Updates Upon Task Completion

Our use case, designed for continuous operation, is apt for environments with frequent changes. When a robot completes a task and is assigned a new one, it necessitates a plan update. This involves generating a new path for the robot, reflecting its route to the next destination. While this presents challenges, we will showcase its effectiveness in dynamic environments and discuss strategies to mitigate these challenges.

7.1.2 Operator Selection

Choosing suitable operators is crucial for the effective realization of our model. Given the complexities involved, we primarily focus on basic operators, excluding rotation operators, to impose stricter constraints and better understand the framework's limits. This approach aids in understanding and preparing for worst-case scenarios.

7.1.3 Pattern Selection for Resource Allocator

In implementing the Pattern Matching Allocator, refer to Section 5.4.3, a crucial component of our framework, our allocator's focus is on two primary pattern types:

- *Direct Cyclic Dependency*, as elaborated in Section 6.3.2.

- *Extended Head-to-Head Conflicts*.

Direct Cyclic Dependency *Direct Cyclic Dependency* in this context focuses on cycles involving more than two robots, as shown in Figure 5.1c, while cycles with only two robots fall under the *Extended Head-to-Head Pattern*, i.e., Figure 5.1b. This pattern is particularly relevant when rotation operators are disallowed, representing an extreme scenario. In practical applications, allowing simultaneous robot rotations can substantially simplify the system.

Extended Head-to-Head The extended head-to-head pattern identifies an impending conflict where two robots require the same resource that is currently being occupied by the other. This scenario indicates an unsatisfiable plan. The pattern is recognized when the conflict is immediate and from certain precursor situations. Recognizing these extended patterns allows for proactive conflict resolution, addressing potential unsatisfiability before it manifests.

These pattern types are chosen based on the understanding that under our model's constraints — specifically the use of only basic operators — any unsatisfiable plan inevitably leads to one of three scenarios, as discussed in Section 6.4.3:

1. Blockage by a stationary robot at its destination.

2. A cyclic dependency involving more than two robots.

3. A head-to-head conflict between two robots.

In the context of our target application, which involves a continuous cycle of tasks where robots are consistently reassigned upon completing their tasks, we assume that new paths will invariably lead to satisfiable plans. This assumption will be examined and elaborated upon in subsequent sections. Consequently, our primary focus is determining whether there is a direct cyclic dependency involving more than two robots and whether a pattern of inevitable head-to-head conflict between two robots exists, *i.e., Extended Head-to-Head Conflicts*.

7.2 Implementation of Pattern Detectors

In implementing our pattern detectors, we address two primary concerns: detecting cyclic dependencies involving more than two robots and identifying extended direct conflicts between two robots, termed Extended Head-to-Head Conflicts. The methodology for each is outlined below.

7.2.1 Cyclic Dependency Detection

We construct a directed graph using the first two nodes of each robot's path from the current plan to detect cyclic dependencies involving more than two robots. This graph construction excludes robots that have reached their destination. Afterward, we employ various algorithms to identify cycles. For targeted detection, we utilize a depth-first search (DFS) algorithm, which effectively traverses the graph to identify cycles, similar to cycle detection in a linked list.

This DFS approach efficiently identifies cycles within the graph, highlighting direct cyclic dependencies among multiple robots. Since assignments are performed iteratively, this check is selectively applied to the currently active robot—the one potentially being assigned a new path segment in the current iteration—thereby

Algorithm 3 DFS Algorithm for Cycle Detection

```
procedure DFS_CYCLE_DETECT(graph)
    Initialize visited ← {}, stack ← {}
    for each node in graph do
        if node not in visited then
            if DFS_VISIT(node, visited, stack) then
                return true                              ▷ Cycle detected
            end if
        end if
    end for
    return false                                         ▷ No cycle detected
end procedure
procedure DFS_VISIT(node, visited, stack)
    Add node to visited and stack
    for each neighbor of node do
        if neighbor not in visited then
            if DFS_VISIT(neighbor, visited, stack) then
                return true
            end if
        else if neighbor in stack then
            return true                                  ▷ Cycle found
        end if
    end for
    Remove node from stack
    return false
end procedure
```

reducing the algorithm's complexity.

For comprehensive global detection, we can use Tarjan's algorithm [30, 31] or Kosaraju's algorithm [32] to identify all strongly connected components within the directed graph. This method is highly effective for quickly pinpointing all cycles, thereby facilitating a global overview of cyclic dependencies. Tarjan's algorithm enhances our capacity to detect and address potential deadlocks system-wide, ensuring robustness in our planning and coordination efforts.

7.2.2 Extended Head-to-Head Conflict Detection

The detection of Extended Head-to-Head Conflicts aims to identify scenarios where the paths of two agents directly oppose each other, potentially leading to collisions. This is characterized by two agents having path segments that, if extended toward each other, would result in a conflict due to occupying the same space from opposite directions.

Formally, let P_1 and P_2 represent the paths of two distinct agents, expressed as strings. An Extended Head-to-Head Conflict occurs if there exist substrings s_1 from P_1 and s_2 from P_2, each starting from the beginning of their respective strings, such that s_1 is the reverse of s_2. This indicates a potential for collision or deadlock, as it implies the agents are moving towards each other along the same trajectory but from opposite directions.

To detect this conflict, we examine the paths of the currently active agent and compare them with those of all other agents within the system. The detection process operates in linear time relative to the maximum path length of the agents, ensuring efficiency. If any pair of agents exhibit this pattern, we consider the Extended Head-to-Head Conflict to exist.

Algorithm 4 Pattern Detector for Extended Head-to-Head Conflicts

```
procedure HasExtendedHeadToHead($arr1$, $arr2$)
    $minLength \leftarrow \min(\text{length}(arr1), \text{length}(arr2))$
    for $i \leftarrow 0$ to $minLength - 1$ do
        if $arr1[i] \neq arr2[minLength - i - 1]$ then
            return false
        end if
    end for
    return true
end procedure
```

7.3 Incompleteness of Pattern Detectors

The pattern detectors, as described, focus on identifying direct cyclic dependencies involving more than two agents and extended head-to-head conflicts between two agents. However, it is important to note that these detectors are not complete in identifying all possible deadlock scenarios that may arise within the system.

While the detectors can effectively identify the aforementioned patterns, there exist scenarios where deadlocks can occur indirectly that the current implementation does not capture. Specifically, situations can arise where a combination of agent positions and path assignments results in an indirect dependency that does not directly manifest as a cycle or head-to-head conflict yet still leads to a deadlock condition.

For example, consider a plan with four agents, where the agent paths are represented as strings: $\gamma_1 = AB$, $\gamma_2 = BCD$, $\gamma_3 = DE$, and $\gamma_4 = ECA$, as in Figure 7.1 In this configuration, while initially no direct cyclic dependency or extended head-

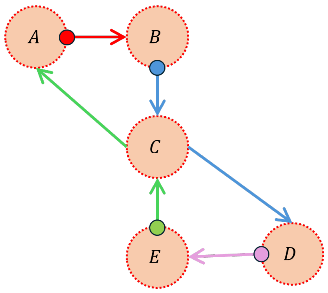


Figure 7.1: Non-Trivial Deadlock

to-head conflict may be detected, the system indeed faces an unsatisfiable state: there exists a condition where, due to indirect dependencies, at least one agent cannot proceed to its destination, signifying an underlying deadlock condition that bypasses the direct detection mechanisms.

It is crucial to acknowledge the potential existence of such indirect dependencies and non-trivial deadlock situations that may not be captured by the current pattern detectors. While the detectors effectively identify and mitigate direct conflicts and cyclic dependencies, they do not guarantee the complete elimination of all deadlock scenarios within the system.

7.4 Layout Settings and Traffic Control

While the pattern detector mechanism is imperfect, strategic layout design and effective traffic control measures can significantly reduce the likelihood of deadlocks in real-world applications.

Careful layout planning, such as incorporating unidirectional paths where possible, can inherently minimize the potential for indirect deadlocks. When combined with allowing simultaneous rotations of agents, this approach can effectively eliminate deadlocks for any plan within the system.

In situations where unidirectional paths are not feasible, such as in corridors, the risk of indirect deadlocks can still be mitigated through the use of carefully designed allocators and traffic control mechanisms. These measures can help generate high-quality plans with a higher likelihood of being satisfiable and a lower potential for conflicts.

For instance, in a general graph, a well-designed traffic control mechanism can prioritize creating plans with unidirectional properties by tracking agents' historical movements. This approach reduces the likelihood of opposing agent movements, minimizing potential conflicts and, consequently, the risk of deadlocks.

7.4.1 Dynamic Traffic Control Mechanism

Within the domain of Multi-Agent Pathfinding (MAPF) algorithms, the deliberative approach typically focuses on optimizing the time spent completing tasks. However, this methodology may not be ideally suited to our framework, which places significant emphasis on traffic control within congested environments, essential for ensuring smooth and efficient navigation. A sophisticated allocation system could underperform if based on a plan of suboptimal quality. For instance, significant delays and blockages can arise if too many robots attempt to navigate through a critical junction simultaneously, adversely affecting overall system flow.

Ideally, robots should reach their destinations in the minimal possible time. Creating such optimal plans from a computational standpoint poses challenges, often hindered by incomplete information and potential execution imperfections in robots. To address these complexities, we have devised a dynamic traffic control mechanism to assist robots in identifying high-quality paths. These paths aim to minimize the total completion time rather than merely reducing travel distance.

The cornerstone of our approach is minimizing the number of junctions that robots need to traverse. We define a junction as a critical point through which multiple robots coming from different directions must pass. Path optimization is achieved by monitoring and utilizing the average travel time across each edge to calculate edge costs.

Specifically, the cost of traversing an edge (a,b) is computed by dividing the occupied time by the total time, representing the probability of that edge being occupied. This derived cost, denoted as c', undergoes a transformation defined by the function $\frac{1}{(1-c')^{\gamma}}$, where γ serves as an adjustment factor. This transformation amplifies the impact of congestion, discouraging the use of highly occupied paths.

The transformation function $\frac{1}{(1-c')^{\gamma}}$ is specifically chosen due to its ability to exponentially increase the traversal cost as the probability of occupancy c' approaches 1. This behavior effectively discourages the use of highly congested paths, aligning with the objective of minimizing overall traffic bottlenecks. The adjustment factor γ provides flexibility, allowing the sensitivity of the cost to be tuned based on specific operational requirements.

The exponential growth reflects the real-world compounding effects of congestion, where increased occupancy disproportionately affects travel times, thereby guiding robots toward less congested routes. This choice ensures that even marginal increases in congestion can significantly alter path selection, promoting a balanced distribution of traffic and enhancing overall system efficiency.

The variation of this cost function is depicted in Figure 7.2, illustrating its exponential growth as edge saturation increases. This function has shown empirical adequacy for our purposes. While future work could explore alternative cost functions or traffic control strategies, this model effectively demonstrates the core concept of our approach.

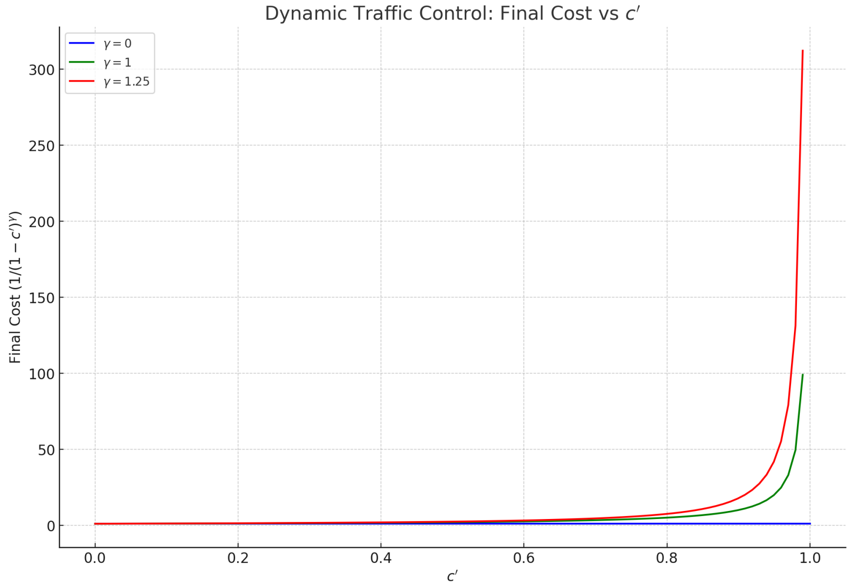
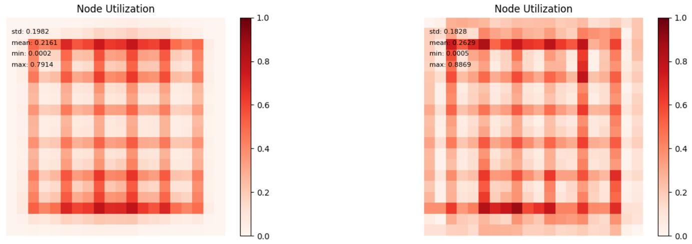


Figure 7.2: Variation of the final cost with respect to c' for different control strengths (γ).

Implementing this strategy encourages the avoidance of highly utilized edges, thus mitigating congestion by directing traffic towards less crowded, peripheral paths, as observed from Figure 7.3.

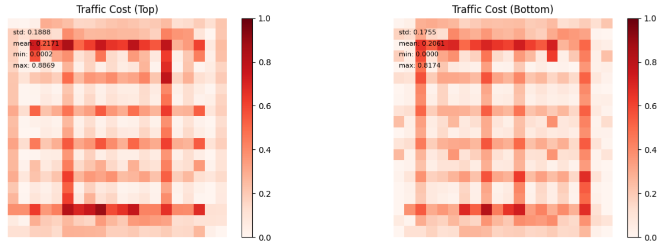
Furthermore, this method reduces the likelihood of head-on collisions and cross-junction traffic jams, inadvertently establishing a rudimentary form of road regulation. An example of such regulation could be the formation of unidirectional lanes by the robots to facilitate smoother travel. While these guidelines are not enforced rigidly, deviations are permitted should the robots assess them as beneficial. This phenomenon can be observed from Figure 7.4, as there is clear evidence that paths, in straight lines vertically or horizontally, are assigned with higher costs, in-



(a) Heat Map without Traffic Control

(b) Heat Map with Traffic Control

Figure 7.3: Heat Maps for Node Utilization Rate



(a) Costs for Moving Up

(b) Costs for Moving Down

Figure 7.4: Heat Maps for Edge Costs.

directly resulting in the preference for unidirectional regulation.

7.5 Robustness and Timeout Mechanism

While the proposed pattern detectors and traffic control mechanisms significantly reduce the likelihood of deadlocks and enhance system efficiency, it is crucial to incorporate additional measures to ensure system robustness and adaptability in the face of unexpected scenarios or imperfect executions.

7.5.1 Timeout Mechanism

In optimal scenarios, the allocator should ensure that a satisfiable initial plan remains feasible after task assignment. However, this ideal state may not always be achievable in practice due to unforeseen disruptions or execution errors.

To mitigate such discrepancies, we introduce a timeout mechanism. This mechanism comes into play if a robot fails to secure necessary resources within a designated timeframe. Consequently, the robot deems its current path unfeasible and initiates the creation of an alternative route, possibly engaging the traffic control strategies outlined earlier. This approach prevents robots from being trapped indefinitely in deadlock situations due to unavailable resources.

Specifically, when a robot's path is determined to be unsatisfiable, the unattainable resource is added to a tabu set, rendering it ineligible for selection in the subsequent path calculation. This tabu set remains active until the end of the current cycle or all alternative paths are blocked by the tabu constraints. Although this may result in a robot's inability to find an immediate alternate route, it typically resolves congestion, as all robots are subjected to similar constraints.

While this mechanism might erroneously classify a merely congested path as unreachable, it indirectly aids traffic management by deterring robots from entering high-density areas. The results section will evaluate and discuss this method's practicality and efficiency.

7.5.2 Robustness and System Resilience

The timeout mechanism addresses potential deadlocks and contributes to the multi-robot system's overall robustness and resilience. In real-world scenarios, robots may encounter unexpected challenges, such as hardware failures or human interventions, that can hinder their progress and affect the entire system.

By incorporating the timeout mechanism, other robots can adapt to these situations by modifying their paths and avoiding areas where a robot may be stuck or operating at a significantly reduced speed. This adaptability prevents system paralysis and allows the multi-robot system to function effectively despite unexpected failures.

CHAPTER 8

EXPERIMENTS AND DISCUSSIONS

This chapter implements the proposed automated sorting center model framework, which was previously mentioned in Section 3.2. We summarize the model as follows:

- The sorting center environment is modeled as a domain with designated pickup and drop-off points.

- Multiple homogeneous robots share this workspace, tasked with continuously shuttling packages between these points.

- The environment is discretized into nodes or vertices, each representing a unique location that can only be occupied by one agent at a time.

- A complete pickup and delivery operation constitutes one cycle. Upon completing a cycle, a robot immediately receives a new task, initiating the next pickup process.

- The primary performance metric is the maximum achievable throughput (Θ), defined as the number of completed cycles per unit of time, irrespective of the number of robots employed.

Additionally, all the robots have a maximum rotation speed of $k\ rad/s$, maximum acceleration $a_{max}\ ms^{-2}$, maximum deceleration $d_{max}\ ms^{-2}$, and maximum speed $v_{max}\ ms^{-1}$. Figure 3.1 shows an automated sorting center layout, and Table 8.1 shows the exact values of the above parameters. This model assumes that the robots try to reach their destination as quickly as possible.

Table 8.1: Value of described parameters.

Parameter Name	**Value**	**Unit**
$N \times M$	20×20	-
P_{num}	12	-
D_{num}	24	-
ℓ	0.6	m
v_{max}	3	ms^{-1}
a_{max}	3	ms^{-2}
d_{max}	3	ms^{-2}
k	π	rad/s

8.1 Framework Implementation Details

We implemented sliding window and iterative resource allocators, along with a pattern-matching allocator equipped with advanced head-to-head and direct cyclic dependency detection, as detailed in Section 5.4.3. In our iterative resource allocator, each agent is limited to reserving at most K continuous resources. Additionally, we maintain a reservation queue for each agent, ensuring resources are reassigned only after being removed from a queue. While specific to our implementation, this detail does not impact the algorithmic model's overall analysis; the reserved resources are allocated and eventually released within a short timeframe.

Moreover, we integrated the timeout mechanism described in Section 6.4.3. This mechanism marks an agent's path as unreachable if resources are not allocated within a predefined period, prompting the generation of a new path. We also adopted the traffic control strategy from Section 3.2, which enhances path efficiency by adjusting edge costs based on historical data.

Additionally, we employed the A* algorithm 2.2.3 with a minimal Manhattan heuristic to guarantee optimality, thereby aligning the algorithm's optimality with that of Dijkstra's algorithm. This strategy helps to avoid potential suboptimality issues in our experiments, enhancing the reliability of the results.

These elements form the foundation of our proposed framework, crafting a thorough system for multi-agent pathfinding management. By amalgamating the allocators, pattern detectors, timeout mechanisms, and traffic control, we forge a potent and efficient framework capable of addressing the complexities involved in multi-agent coordination within shared spaces.

8.2 Simulation Settings

We developed the simulator using discrete event simulation techniques in C++, with outcomes recorded as JSON files for subsequent visualization and analysis in Python. The experiments were conducted in a smaller-scale environment, depicted in Figure 3.1, to ensure meaningful comparisons and manageable simulation durations. It is important to note that while scaling up to larger environments might necessitate minor parameter adjustments or implementation tweaks, the principal findings and conclusions presented here are broadly applicable.

The experiments primarily focused on evaluating the framework's key components: allocation strategies, the timeout mechanism, and traffic control. These elements represent the main themes and contributions of our study. By scrutinizing these fundamental aspects, our aim was to highlight their significance and effectiveness within the overall system.

Each experiment was run for a simulation duration of five hours to capture steady-state system behavior and provide reliable performance metrics. Multiple runs were conducted for each experimental configuration to account for stochastic variations and ensure the statistical significance of the results.

To facilitate the comparison of different experimental configurations, we introduce the concept of the best achievable throughput . For a specific set of framework components and parameters, the best achievable throughput represents the maximum throughput attained by the system, regardless of other factors, such as the number of agents. This metric allows us to evaluate the performance potential of each configuration and identify the most effective combinations of allocation strategies, timeout thresholds, traffic control parameters, and so on.

In the following sections, we will elaborate on the experimental design, present the results, and discuss the insights gained from analyzing the performance and characteristics of our proposed framework for multi-robot control and coordination. The findings validate the effectiveness of the implemented mechanisms and illustrate the framework's versatility and potential for broader applications in real-world multi-agent pathfinding scenarios.

8.3 Comparison of Different Allocators

As mentioned previously, we can design various allocators, from native (i.e., assign if there is no direct conflict on occupying the same node) to pattern-matching allocators with extended head-to-head detectors and/or cyclic dependency detectors. This section compares the performance of the native allocator, the pattern-matching allocator with an extended head-to-head detector, and the pattern-matching allocator with both an extended head-to-head detector and a cyclic dependency detector. We use the best settings for each allocator to ensure a fair comparison. The results are shown in Figure 8.1.

The orange bar represents the throughput achieved by the pattern-matching allocator with both an extended head-to-head detector and a cyclic dependency detector, the blue bar represents the throughput achieved by the pattern-matching allocator with only an extended head-to-head detector, and the purple bar represents the throughput achieved by the native allocator.

From the results, we can easily observe that the pattern-matching allocators

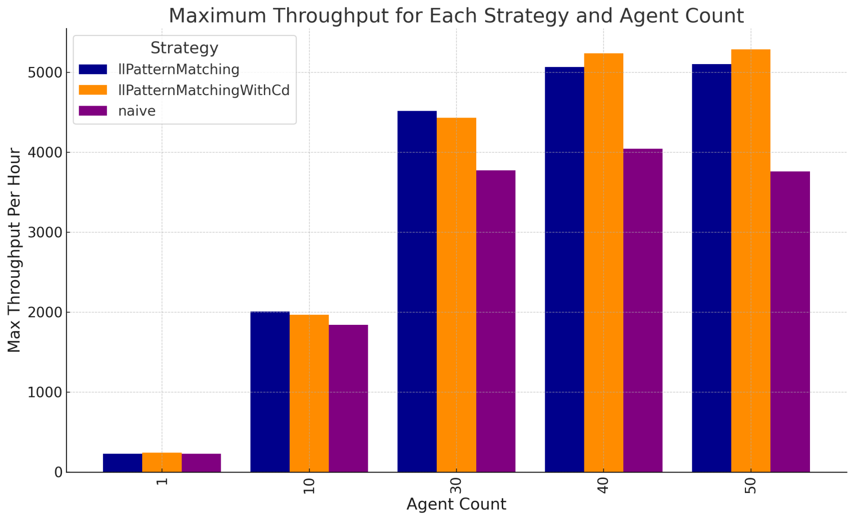


Figure 8.1: Best Throughput for Each Allocator Strategy and Agent Count

have a significant advantage over the native allocator. The addition of the cyclic dependency detector provides a relatively small benefit compared to the pattern-matching allocator with only an extended head-to-head detector, but it still produces better results in most cases. This suggests that the chance of a cycle occurring is quite low, and the cyclic dependency detector could be omitted if it burdens the computational resources.

Comparing the best achievable throughput (8.2) for each strategy, the pattern-matching allocator without a cyclic dependency detector achieves a throughput that is 3.47% lower than that with a cyclic dependency detector. The native allocator achieves a throughput that is 23.52% lower than the highest throughput achieved among the strategies.

8.4 Traffic Control Mechanism

Chapter 7.4.1 discussed the traffic control mechanism, which employs a transformation function with an adjustable factor γ indicating the strength of traffic control. When γ is zero, traffic control is turned off. As in the previous section, we will show the best results for different γ values (0.0, 1.0, 2.5, 5.0, and 7.5), grouped by the agent count, regardless of other settings.

Figure 8.2 demonstrates that the traffic control mechanism performs best when $\gamma = 2.5$, achieving an approximately 58.24% higher throughput compared to the case where $\gamma = 0.0$ (i.e., when the traffic control mechanism is disabled), considering the maximum achievable throughput regardless of the number of agents. These results clearly showcase the effectiveness of the traffic control mechanism and underscore the importance of reducing conflicts in multi-agent pathfinding scenarios.

The traffic control mechanism's success can be attributed to its ability to dynamically adjust edge costs based on historical data, effectively guiding agents towards less congested paths. Encouraging agents to avoid highly utilized edges mitigates congestion and reduces the likelihood of head-on collisions and cross-junction traffic jams. This adaptive approach improves overall system efficiency and contributes to the formation of rudimentary road regulations, such as unidirectional lanes, which further enhance the smooth flow of agent traffic.

The experimental results also highlight the significance of tuning the γ parameter to balance traffic control and individual agent objectives. While a higher γ value can lead to more stringent traffic control, an excessively high value may cause agents to deviate too far from their optimal paths, potentially impacting overall system performance. The optimal γ value of 2.5 found in these experiments suggests that a moderate level of traffic control is most beneficial, allowing agents to adapt to congestion while prioritizing their individual goals.

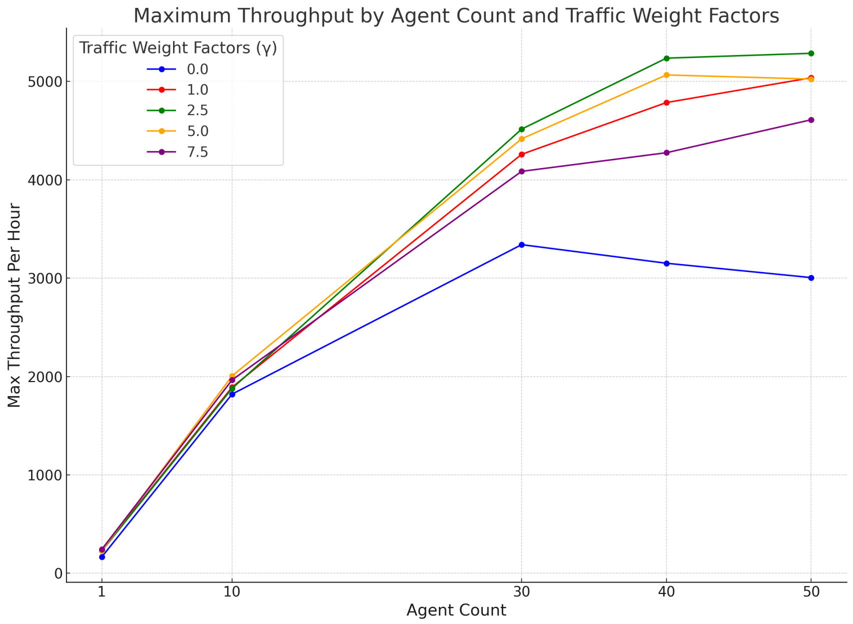


Figure 8.2: Best Throughput for Each Factor γ and Agent Count

In conclusion, the traffic control mechanism is a crucial component of the proposed multi-agent pathfinding framework, significantly enhancing system performance and throughput. Its ability to dynamically adjust agent paths based on real-time traffic conditions and historical data makes it a powerful tool for managing the complexities of multi-robot coordination in real-world scenarios. Future research could explore more advanced traffic control strategies and adaptive techniques to optimize the mechanism's performance across various domains and applications.

8.5 Maximum Reservation

The maximum reservation is an important parameter for the sliding-window approach. Limits the maximum number of resources that can be reserved per robot. The minimum reservation should be two; otherwise, the robot cannot move. A lower maximum reservation value reduces resource utilization waste. However, it also provides less buffer for the robot to accelerate, reduces the maximum achievable speed, and less tolerance for network delays and other factors.

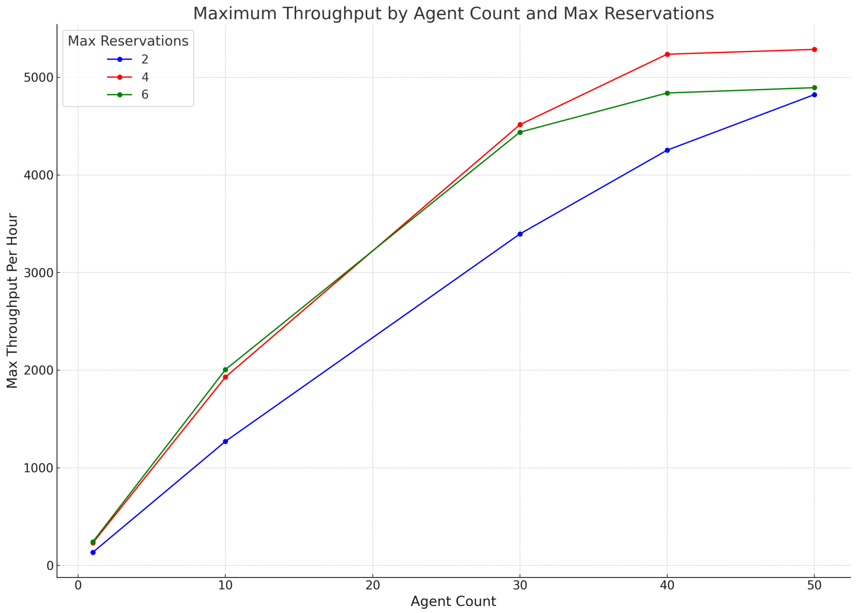


Figure 8.3: Best Throughput for Maximum Reservation and Agent Count

Figure 8.3 shows the performance impact of different maximum reservation values. As expected, a maximum reservation of two results in the lowest throughput. Interestingly, when the number of agents reaches 50, the throughput for a maximum reservation of six is close to that of two, although a reservation of six performs far better in most cases. This is because resource utilization waste reduces the system's saturation point. The saturation point is the maximum achievable throughput for a given environment, framework, and setting, regardless of the agent count.

On the other hand, a maximum reservation of four generally performs the best. It achieves the best trade-off between space utilization and providing a suitable buffer size for the robots to accelerate. It's worth noting that higher robot speeds actually increase the benefit gained per resource utilized.

8.6 Timeout Mechanism

The timeout mechanism serves as a backup mechanism for handling unsatisfiable plans. When implementing it, we need to determine a threshold for the timeout value. A higher threshold makes it more difficult for the current plan to be marked as unsatisfiable, resulting in a lower chance of path switching.

Figure 8.4 shows that the best achievable throughput is attained when the timeout threshold is six. This is because a lower threshold makes it easier for a robot to decide to switch its path, resulting in a longer path length. However, if the timeout threshold is too high, the robot waits for a resource instead of switching to a new path. This causes congestion to be more challenging to resolve, and in cases where the plan is truly unsatisfiable, the robot wastes time waiting.

The timeout mechanism experiment highlights the importance of balancing when setting the timeout threshold. A threshold that is too low can lead to unnecessary path switches and increased path lengths, while a threshold that is too high can result in robots waiting excessively for resources, even in unsatisfiable scenarios. As determined by the experiment, the optimal timeout threshold of six represents a sweet spot where robots can effectively determine when to switch paths without prematurely abandoning viable plans or waiting too long in deadlock situations.

Incorporating the timeout mechanism with a well-tuned threshold enhances the multi-robot system's overall robustness and adaptability. By allowing robots to switch paths when necessary dynamically, the mechanism helps mitigate congestion and resolve potential deadlocks, improving system performance and throughput.

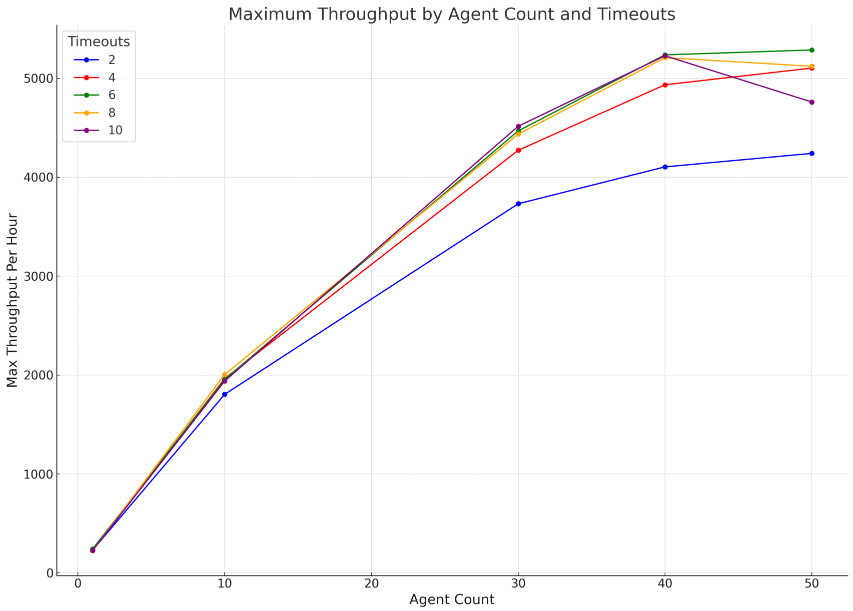


Figure 8.4: Best Throughput for Timeout Threshold (Seconds) and Agent Count

Future research could explore more advanced techniques for dynamically adjusting the timeout threshold based on real-time system conditions or historical data. Such adaptive approaches could optimize the timeout mechanism's effectiveness across different scenarios and robot densities.

8.7 Analysis of Framework and System Characteristics

We have studied several parameters within the proposed framework and investigated their effects on the system. Generally, we observe that the marginal benefits decrease as the number of robots increases, leading to a saturation point that significantly depends on the framework's efficiency. A high-performance framework allows for satisfactory results with fewer robots and achieves higher performance.

Moreover, algorithmic complexity and iterative assignment processes influence the system's scalability. Allocating the entire plan before completing the planning phase is unnecessary. Additionally, our system implementation naturally supports parallelization, efficiently utilizing multiple processors or agents.

To provide a preliminary perspective on scalability, we conducted an experiment generating a feasible plan from a complete graph with 10,000 nodes involving 1,000 agents, each with a path length of 50 for every agent. We implemented a straightforward version of the algorithmic model discussed without plan simplification, assuming agents disappear after reaching their destinations. The pseudocode for the plan generation function used in this experiment can be found in Pseudocode 1.

Algorithm 5 Generate Satisfiable Plan from a Complete Graph

```
procedure GENERATEPLAN($agentCount$, $pathLength$, $maxResources$)
    Initialize $resourceSet$ with available resources
    Initialize empty $plan$ for each agent
    Initialize empty $solution$ sequence
    for each agent do
        for $pathLength$ times do
            Append agent to $solution$ sequence
        end for
    end for
    Shuffle $solution$ sequence randomly
    for each assignment in $solution$ sequence do
        Randomly select a resource from $resourceSet$
        if agent's path is not empty then
            Release the last resource assigned to the agent
        end if
        Remove selected resource from $resourceSet$
        Append selected resource to agent's path
    end for
    return $plan$ and $solution$
end procedure
```

This implementation, crafted in Python 3.9 without multiprocessing capabilities and executed on an i9-13900KF processor, completed the suitable assignment sequence of 49,950 assignments in approximately 25 seconds. Moreover, we anticipate that we can significantly improve its performance with proper implementation and optimization, employing appropriate programming languages and parallelization techniques.

8.8 Conclusion

The experiments conducted in this chapter validate the effectiveness and robustness of the proposed multi-robot control and coordination framework in managing the complexities of multi-agent pathfinding. The most significant finding is the framework's ability to maintain functionality and avoid system paralysis, even under extreme conditions such as high robot densities, without relying on simultaneous rotations or assumptions about road regulations. This demonstrates the framework's resilience and adaptability in challenging scenarios, making it suitable for a wide range of real-world applications.

The comparison of allocator strategies, traffic control mechanisms, maximum reservation settings, and timeout thresholds provided valuable insights into optimizing system performance and throughput. The pattern-matching allocator with advanced conflict detection achieved the highest throughput, while the traffic control mechanism significantly enhanced system performance by dynamically adjusting edge costs. The optimal maximum reservation setting balanced space utilization and robot acceleration, and the timeout mechanism proved crucial in mitigating congestion and resolving deadlocks.

In conclusion, the experiments support the framework's design choices and demonstrate its potential to revolutionize multi-robot coordination and control. The framework's robustness, versatility, and demonstrated improvements in system performance and throughput highlight its suitability for real-world applications and provide valuable guidance for future research in the field of multi-robot systems.

CHAPTER 9

CONCLUSION

9.1 Summary of the Research

This dissertation presented a robust, scalable multi-robot control and coordination framework designed to optimize operational dynamics and maximize throughput in parcel sorting centers and similar multi-agent pathfinding scenarios. The research was motivated by the growing deployment of multi-robot systems across various sectors and the inherent challenges in efficiently coordinating and navigating these systems, particularly in dynamic, real-world environments.

The core of the research was the development of an innovative two-stage framework that decouples the multi-agent pathfinding problem into path planning and resource allocation components. This separation allowed for the generation of collision-free paths for each robot, followed by the dynamic scheduling of robot movements to ensure conflict-free navigation. The framework employed a well-defined algorithmic model to formalize the concepts of plans, assignments, and operations, providing a solid foundation for agent coordination.

A comprehensive implementation strategy was provided, focusing on key aspects such as dynamic plan updates, operator selection, and pattern recognition for resource allocation. This included the development of advanced pattern detectors for identifying potential conflicts, like direct cyclic dependencies and extended head-to-head conflicts, as well as innovative traffic control mechanisms to optimize agent paths for improved efficiency.

The performance of the proposed framework was rigorously evaluated through a series of simulation experiments within the context of an automated sorting center model. The experiments demonstrated the framework's effectiveness in managing the operational dynamics of the multi-robot system, showcasing its ability to handle dynamic task assignments, avoid collisions, and optimize overall throughput, even under challenging conditions.

9.2 Key Contributions and Impact

The main contributions of this research are as follows:

1. An innovative two-stage framework for multi-agent pathfinding that decouples path planning and resource allocation, enabling efficient coordination and navigation of multi-robot systems.

2. A well-defined algorithmic model that formalizes the concepts of plans, assignments, and operations, providing a robust theoretical foundation for the proposed framework.

3. The development of advanced pattern detectors for identifying potential con-

flicts and deadlocks significantly enhances the robustness and reliability of the multi-robot system.

4. Implementing a dynamic traffic control mechanism that optimizes agent paths based on historical data, leading to notable improvements in overall system efficiency and throughput.

5. Extensive simulation experiments that validate the effectiveness of the proposed framework in managing the operational dynamics of multi-robot systems in realistic sorting center scenarios.

This framework's successful development and validation have far-reaching implications for designing and deploying multi-robot systems across various domains. By providing a structured, efficient, and scalable approach to multi-agent pathfinding, this research contributes significantly to advancing autonomous systems and their real-world applications. The framework's ability to handle dynamic task assignments and optimize agent coordination in real time makes it particularly well-suited for scenarios demanding high adaptability and responsiveness, such as logistics, manufacturing, and emergency response.

9.2.1 Limitations

While the proposed multi-robot control and coordination framework demonstrates significant potential for enhancing the operational dynamics of parcel sorting centers and similar scenarios, there are certain limitations to the current research that should be acknowledged:

1. **Simulation-based validation:** Although the simulation experiments provide strong evidence for the framework's effectiveness, the research would benefit from real-world validation through physical robot experiments or case studies in actual parcel sorting centers. This would further demonstrate the framework's robustness and practicality in handling real-world challenges and uncertainties.

2. **Homogeneous robot assumption:** The current framework assumes a homogeneous fleet of robots with identical capabilities and constraints. Extending the framework to accommodate heterogeneous robot teams with varying abilities and limitations would increase its applicability to a wider range of real-world scenarios.

3. **Centralized control architecture in the current implementation:** While the proposed framework is designed to be scalable and adaptable to different control architectures, the current implementation relies on a centralized control scheme. Although this approach simplifies the coordination process, further research is needed to explore and validate the framework's performance in partially distributed or decentralized control settings. Investigating these alternative architectures could provide valuable insights into the framework's scalability and robustness in large-scale deployments.

4. **Integration with external systems:** The current research focuses on the core aspects of multi-robot coordination and control within the proposed framework. However, integrating the framework with external systems, such as task allocation modules, higher-level decision-making systems, or human-robot interaction interfaces, would be essential for real-world deployments.

Future work should explore the challenges and opportunities associated with integrating the framework into a broader ecosystem of autonomous systems.

9.2.2 Future Research Directions

The limitations discussed above serve as motivation for several exciting avenues of future exploration:

1. Conducting real-world experiments and case studies to validate the framework's performance in various application domains, demonstrating its practical value and potential for industry adoption.

2. Investigating alternative cost functions and advanced traffic control strategies to optimize agent path planning and system throughput further.

3. Extending the framework to heterogeneous multi-robot systems with diverse capabilities and constraints, broadening its applicability to more complex real-world scenarios.

4. Incorporating machine learning techniques enables the framework to learn from historical data and adapt to improve performance over time.

5. Exploring the integration of the framework with other critical components of autonomous systems, such as task allocation and high-level decision-making, to develop more comprehensive and intelligent multi-robot systems.

6. Investigating the framework's performance in partially distributed or decen-

tralized control settings to gain valuable insights into its scalability and robustness in large-scale deployments.

9.2.3 Potential Real-World Impact and Broader Implications

The proposed multi-robot control and coordination framework has significant potential to revolutionize various industries and domains. In the warehousing and logistics sector, the framework could optimize operations in large-scale warehouses, improving inventory management and order fulfillment. Manufacturing plants could benefit from enhanced efficiency and flexibility in assembly lines and other processes. Furthermore, the framework could be applied to healthcare settings, such as hospitals and laboratories, facilitating the coordination of autonomous mobile robots for material handling and delivery.

Implementing the framework in these industries could lead to substantial economic benefits. The efficient coordination of multi-robot systems can increase productivity, reduce labor costs, and improve resource utilization, resulting in cost savings and environmental benefits. Adopting the framework can also enhance competitiveness and drive innovation in these industries, ultimately contributing to economic growth and job creation.

The framework's contributions extend beyond specific industries, as it advances the field of multi-robot systems as a whole. It provides a foundation for developing more sophisticated and adaptable multi-robot control and coordination strategies, encouraging further research on topics such as heterogeneous robot teams, machine learning integration, and human-robot collaboration. The framework demonstrates the feasibility and benefits of implementing advanced multi-robot systems in real-world settings by bridging the gap between theoretical research and practical applications.

Moreover, the framework has the potential to generate a positive societal impact. By enabling robots to perform hazardous or repetitive tasks, it can improve working conditions and safety in various industries. The efficient coordination of multi-robot systems can also facilitate the development of new products and services, benefiting consumers. Overall, the framework contributes to advancing technology and automation, improving quality of life, and increasing opportunities for human workers to focus on higher-level tasks.

9.2.4 Conclusion

In conclusion, this dissertation presents significant advancements in the field of multi-robot systems and autonomous navigation. The proposed framework, with its innovative two-stage approach to multi-agent pathfinding, offers a robust and scalable solution to the complex challenges of coordinating multiple robots in dynamic environments. By decoupling path planning from resource allocation, the framework achieves improved computational efficiency and adaptability, addressing key limitations in existing MAPF algorithms.

The research's impact extends beyond theoretical contributions, offering practical implementations that bridge the gap between abstract models and real-world applications. The development of advanced pattern detectors and dynamic traffic control mechanisms significantly enhances the framework's ability to handle complex scenarios, potentially revolutionizing operations in parcel sorting centers and similar multi-robot environments.

Furthermore, the comprehensive experimental validation provides valuable insights into the framework's performance under various conditions, establishing a solid foundation for future research in this domain. The framework's demonstrated ability to maintain functionality even under high robot densities, without relying on simultaneous rotations or predefined road regulations, underscores its robustness and potential for broad applicability across different sectors.

As autonomous systems continue to evolve, the methodologies and insights developed in this research will play a crucial role in shaping the future of multi-robot coordination and control. By addressing current limitations in MAPF algorithms and providing a flexible, efficient solution, this work paves the way for more intelligent, reliable, and scalable autonomous systems, contributing significantly to the advancement of robotics and automation technologies.

BIBLIOGRAPHY

[1] “Geek+ introduces industry’s tallest mobile robot for up to 12-meter-high warehouse automation - marketwatch.” [Online]. Available: http://surl.li/rlnab

[2] “Geodis signs expanded agreement with locus robotics to deploy 1,000 locusbots at global warehouse sites | geodis united states.” [Online]. Available: http://surl.li/rlnae

[3] O. Salzman and R. Stern, “Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems,” in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS ’20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 1711–1715.

[4] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” in *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ser. IAAI’07. AAAI Press, 2007, p. 1752–1759.

[5] D. D. Paola, A. Milella, G. Cicirelli, and A. Distante, “An autonomous mobile

robotic system for surveillance of indoor environments," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 8, 2010. [Online]. Available: https://doi.org/10.5772/7254

[6] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. Kumar, R. Barták, and E. Boyarski, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," *SOCS*, vol. 10, no. 1, pp. 151–158, Sep. 2021. [Online]. Available: https://ojs.aaai.org/index.php/SOCS/article/view/18510

[7] Z. Yan, N. Jouandeau, and A. A. Cherif, "A Survey and Analysis of Multi-Robot Coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, Dec. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.5772/57313

[8] H. Ma, J. Li, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, p. 837–845.

[9] F. ., W. Rahiman, and S. Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, pp. 1–25, 01 2019.

[10] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cyber.*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available:

http://ieeexplore.ieee.org/document/4082128/

[11] S. Song, K. I. Na, and W. Yu, "Anytime lifelong multi-agent pathfinding in topological maps," *IEEE Access*, vol. 11, pp. 20 365–20 380, 2023.

[12] F. Grenouilleau, W.-J. Hoeve, and J. Hooker, "A multi-label a* algorithm for multi-agent pathfinding," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, pp. 181–185, 05 2021.

[13] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong Multi-Agent Path Finding in Large-Scale Warehouses," *AAAI*, vol. 35, no. 13, pp. 11 272–11 281, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17344

[14] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 1152–1160.

[15] J. Yu and D. Rus, *Pebble Motion on Graphs with Rotations: Efficient Feasibility Tests and Planning Algorithms*. Cham: Springer International Publishing, 2015, pp. 729–746. [Online]. Available: https://doi.org/10.1007/978-3-319-16595-0_42

[16] M. Kulich, T. Novák, and L. Přeucil, "Push, stop, and replan: An application of pebble motion on graphs to planning in automated warehouses," in *2019 IEEE*

Intelligent Transportation Systems Conference (ITSC), 2019, pp. 4456–4463.

[17] B. Nebel, "The computational complexity of multi-agent pathfinding on directed graphs," *Artificial Intelligence*, vol. 328, p. 104063, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370223002096

[18] H. Ma, G. Wagner, A. Felner, J. Li, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with deadlines," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, p. 417–423.

[19] J. Li, W. R. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. [Online]. Available: https://par.nsf.gov/biblio/10296523

[20] P. Surynek, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with capacity constraints," in *AI*IA 2019 – Advances in Artificial Intelligence*, M. Alviano, G. Greco, and F. Scarcello, Eds. Cham: Springer International Publishing, 2019, pp. 235–249.

[21] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001386

[22] M. Cap, P. Novak, A. Kleiner, and M. Selecky, "Prioritized Planning

Algorithms for Trajectory Coordination of Multiple Mobile Robots," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7138650/

[23] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4603–4609.

[24] S. Zhang, J. Li, T. Huang, S. Koenig, and B. Dilkina, "Learning a priority ordering for prioritized planning in multi-agent path finding," *Proceedings of the International Symposium on Combinatorial Search*, vol. 15, pp. 208–216, 07 2022.

[25] M. O. Keskin, F. Cantürk, C. Eran, and R. Aydoğan, "Decentralized multi-agent path finding framework and strategies based on automated negotiation," *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 1, mar 2024. [Online]. Available: https://doi.org/10.1007/s10458-024-09639-8

[26] H. Lu, "Decentralized multi-agent path finding based on deep reinforcement learning," in *Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2023)*. Atlantis Press, 2023, pp. 185–192. [Online]. Available: https://doi.org/10.2991/978-94-6463-300-9_19

[27] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370222000029

[28] W. Hoenig, T. K. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, “Multi-Agent Path Finding with Kinematic Constraints,” *ICAPS*, vol. 26, pp. 477–485, Mar. 2016. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/13796

[29] H. Ma, W. Hönig, T. K. S. Kumar, N. Ayanian, and S. Koenig, “Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery,” *AAAI*, vol. 33, no. 01, pp. 7651–7658, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4759

[30] R. Tarjan, “Depth-first search and linear graph algorithms,” in *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, 1971, pp. 114–121.

[31] R. Chen, C. Cohen, J.-J. Lévy, S. Merz, and L. Théry, “Formal Proofs of Tarjan’s Strongly Connected Components Algorithm in Why3, Coq and Isabelle,” in *10th International Conference on Interactive Theorem Proving (ITP 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), J. Harrison, J. O’Leary, and A. Tolmach, Eds., vol. 141. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, pp. 13:1–13:19. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2019.13

[32] L. Théry, “Formally-proven kosaraju’s algorithm,” 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:3515122