

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Enhancing Mobile Robot Navigation: Integrating Time Discretisation and Action Control in a Fleet Coordination System using TEA*

Filipe António Rodrigues Cardoso



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Pedro Luís Cerqueira Gomes da Costa

Second Supervisor: Prof. José Luis Sousa de Magalhães Lima

July 29, 2025

Resumo

O Time Enhanced A* (TEA*), é uma forma de planeamento de trajetórias adotada para coordenar frotas de robôs móveis autónomos. Embora o TEA* introduza uma camada temporal para evitar colisões e otimizar o planeamento de rotas, o seu funcionamento baseia-se em alguns pressupostos simplificadores. Estes podem ser resumidas da seguinte forma: todos os tempos de execução de uma aresta do mapa de grafos são uniformes e as interações com recursos externos, tais como portas ou elevadores, são instantâneas ou têm custos fixos. Contudo, esta premissa nem sempre se verifica em contextos reais, em que variáveis como a variação da extensão dos percursos, a velocidade dos robôs e os obstáculos dinâmicos podem perturbar significativamente a execução.

Na seguinte dissertação foram propostas e desenvolvidas duas extensões modulares para ultrapassar os desafios identificados e aumentar a viabilidade real dos sistemas de coordenação multi-robô: o submódulo Graph Decomposer e o Action Handler. O Decomposer introduz a discretização temporal na estrutura de planeamento do TEA*, subdividindo as arestas no gráfico do ambiente de acordo com o tempo de travessia sincronizando globalmente com o resto do mapa. Esta melhoria aumentou a resolução e a coerência do planeamento, permitindo uma sincronização mais precisa entre agentes. O módulo Action Handler aumenta a estrutura de coordenação incorporando pré-condições e pós-condições baseadas na lógica que regem a interação entre robôs e elementos de infraestrutura partilhados, como portas e elevadores. Esta adição facilita a capacidade do sistema para antecipar atrasos e programar interações com maior inteligência, reduzindo o tempo de inatividade e evitando conflitos de percurso em pontos de engarrafamento críticos.

Ambos os módulos foram implementados num sistema de coordenação de frotas desenvolvido pelo Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESCTEC) no Laboratório de Indústria e Inovação (iiLab). O processo de validação destes módulos foi efetuado utilizando o ambiente de simulação Stage no Robot Operating System (ROS). Tal como demonstrado pelas simulações, verificou-se uma melhoria acentuada do desempenho do sistema, incluindo uma redução dos erros de planeamento, melhores tempos de conclusão das tarefas e uma maior escalabilidade com uma elevada densidade de robôs. A integração destes componentes representa um avanço substancial no desenvolvimento de sistemas de gestão de frotas inteligentes, adaptáveis e operacionalmente robustos, estabelecendo assim as bases para a implementação de robôs móveis autónomos em ambientes industriais e de serviços complexos e dinâmicos.

Palavras-chave: Coordenação multi-robô, TEA*, Gestor de Ações, Pré-condições e Pós-condições, Discretização Temporal, Decomposição de Grafos, Planeamento de Trajetórias, Robôs Móveis Autónomos (AMRs)

Abstract

The Time Enhanced A* (TEA*) algorithm, a path planning strategy adopted for coordinating fleets of autonomous mobile robots. Whilst TEA* introduces a temporal layer to avoid collisions and improve route planning, it operates under some simplifying assumptions. These can be summarised as follows: all edge traversal times are uniform, and interactions with external resources (such as doors or elevators) are instantaneous or incur fixed costs. However, this is not always valid in real-world scenarios, where factors such as varying path lengths, robot speeds, and dynamic obstacles can significantly disrupt execution.

Two modular extensions were proposed and developed to surmount the challenges identified and enhance the real-world viability of multi-robot coordination systems: the Decomposer Submodule and the Action Handler. The Decomposer introduces temporal discretisation into the TEA* planning framework by subdividing edges in the environment graph according to traversal time and a globally optimised time step. This enhancement has increased planning resolution and consistency, enabling more accurate synchronisation across agents. The Action Handler module augments the coordination framework by incorporating logic-based preconditions and postconditions that govern the interaction between robots and shared infrastructure elements such as doors and elevators. This addition facilitates the system's capacity to anticipate delays and schedule interactions with greater intelligence, reducing idle time and avoiding path conflicts in critical choke points.

Both modules were implemented within the fleet coordination system developed by the Institute for Systems and Computer Engineering, Technology and Science (INESCTEC) at the Industry and Innovation Lab (iiLab). The validation process for these modules was undertaken using the Stage simulation environment within the Robot Operating System (ROS) framework. As demonstrated by the simulations, there was a marked improvement in system performance, including a reduction in planning errors, improved task completion times, and enhanced scalability under high robot density. The integration of these components signifies a substantial advancement towards developing intelligent, adaptable, and operationally robust fleet management systems, thereby establishing the foundation for implementing autonomous mobile robots in complex, dynamic industrial and service environments.

Keywords: Multi-robot coordination, TEA*, Action Handler, Preconditions and Postconditions, Temporal Discretisation, Graph Decomposition, Path Planning, Autonomous Mobile Robots (AMRs)

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The research presented in this dissertation contributes to several Sustainable Development Goals through its focus on efficient, scalable, and intelligent coordination of mobile robot fleets. The following SDGs have been identified as most relevant to the scope and impact of this work:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialisation and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	Developing the Decomposer and Action Handler supports process automation, boosting industrial productivity through smart, modular innovation.	Reduction in robot idle time and task completion duration in simulated environments.
9	9.4	Upgrading coordination systems with time-aware and action-aware components improves energy efficiency and reduces system waste.	Lower rate of path replanning and emergency stops; improved task allocation success rate.
	9.5	Advancing robotic research by proposing and validating novel extensions to TEA* and publishing findings in academic venues.	Peer-reviewed publication output; system reuse or adaptation in other research contexts.
12	12.2	Improving resource use in automated warehouses via better coordination of robot fleets reduces waste and promotes sustainable production.	Optimised fleet movement patterns; reduced robot overlap in shared zones; improved scalability in tests.
	12.4	Enabling safer and more predictable robot behaviour in shared environments supports sustainable and responsible industrial automation.	Fewer collisions and coordination deadlocks in high-density simulations.

Acknowledgements

I want to begin by expressing my deepest gratitude to my academic supervisor, Professor Pedro Costa and the co-supervisor of this work, Professor José Lima, for their guidance, insightful feedback, and continued effort put in regular monitoring, which were instrumental throughout this dissertation. Their expertise and availability were invaluable in navigating technical challenges and academic demands.

I also want to sincerely thank Diogo Matos, my supervisor at INESC TEC, for his continuous support, technical insights, and availability during the development of this work. His critical feedback, collaborative spirit and hands-on expertise greatly contributed to the quality and relevance of this dissertation.

I want to extend my heartfelt gratitude to my parents, António Cardoso and Fátima Cardoso, and my sister, Joana Cardoso, for their unwavering support, patience, and love. They were always there to help me when I was in doubt, to lift me up when I was feeling down, and to remind me of my purpose when things felt overwhelming. Their belief in me has been a constant source of strength, and their encouragement laid the foundation for the perseverance and values that have guided me throughout this journey, which will remain at the core of who I am and will uphold throughout my life. I am profoundly grateful; none of this would have been possible without them.

I would like to give a special thanks to my friends Marco Moreira and Mariana Gomes for the time we shared during this semester. I am genuinely grateful for the mutual support we gave each other on our projects, the spirit of camaraderie that made even the busiest days lighter, and all the laughs that helped us push through. Your friendship made this journey not only easier but also far more enjoyable.

Finally, I want to thank all my friends, those I met at university and those who have been part of my life for much longer. Their companionship, meaningful conversations, and light-hearted moments played a vital role in helping me stay balanced and motivated throughout this journey. Whether through insightful advice or a simple shared laugh during tough times, their presence made all the difference; for that, I am genuinely grateful.

Filipe António Rodrigues Cardoso

“We cannot change the cards we are dealt, just how we play the hand.”

Randy Pausch

Contents

Resumo	i
Abstract	iii
UN Sustainable Development Goals	v
Acknowledgements	vii
1 Introduction	1
1.1 Context	1
1.2 Motivation	2
1.3 Objectives	2
1.4 Contributions	3
1.5 Document Structure	3
2 State-of-the-art	5
2.1 State Machines	5
2.2 Deadlocks	6
2.3 Action Control Methods	7
2.3.1 Locks and Mutexes	8
2.3.2 Semaphores	8
2.3.3 Precondition and Postcondition	8
2.3.4 Action Control in the Literature	8
2.4 Possible Architectures of a Fleet Coordination System	10
2.4.1 Decentralised Coordination	10
2.4.2 Centralised Coordination	11
2.4.3 Hybrid Coordination	13
2.5 Grid-Based and Graph-Based Path Planning	14
2.5.1 Grid-Based Path Planning	14
2.5.2 Graph-Based Path Planning	15
2.5.3 Summary	16
3 Fleet Coordination System	17
3.1 Graph Representation	18
3.1.1 Bézier Curves	20
3.1.2 Casteljau’s algorithm	21
3.2 Graph Pre-Processor	22
3.3 Path Planner	23
3.3.1 Time Enhanced A* (TEA*)	23

3.4	Agent Supervisor	24
3.5	Graph Server	25
3.5.1	Summary	26
4	Decomposer Submodule	27
4.1	The Importance of Discretising the Time Dimension	27
4.1.1	Reducing the Impact of Security Features	28
4.2	Step Calculation	29
4.3	Edge Splitting	30
4.4	Decomposer Submodule Tests	31
4.4.1	Time to Generate a Path	32
4.4.2	Impact on Task Execution	35
4.4.3	Summary	38
5	Action Handler	39
5.1	System Architecture	39
5.1.1	Connection With The Fleet Coordination System	41
5.2	Implementing Precondition and Postcondition	41
5.3	Controlling and Managing Doors	42
5.3.1	State Machine	42
5.3.2	Door Simulation	43
5.4	Action Handler Tests	45
5.4.1	Basic Functionality Test	46
5.4.2	Performance Benchmark	48
5.4.3	Scalability Stress Test	50
5.4.4	Summary	52
6	Conclusion and Future Work	53
6.1	Goals Accomplishments	53
6.2	Future Work	54
	References	55
A	Video Links	59
A.1	Action Handler in a Simple Map	59
A.2	Action Handler in the Laboratory Map	59

List of Figures

2.1	Simple State Machine Example	6
2.2	Deadlock Situation	7
2.3	Decentralised System Exemple	10
2.4	Centralised System Exemple	12
2.5	Hybrid System Exemple	13
2.6	A Graph-Based Map and a Grid-Based Map	14
2.7	TEA* Algorithm Applied to a Grid-Based Map [28]	15
3.1	System Overview [25].	18
3.2	Simple Graph Map	19
3.3	Bézier Curve Example	20
3.4	Division of a Bézier Curve Using Casteljau's Algorithm	22
3.5	Graph Layering for TEA* [32]	24
4.1	Impact of the Parallel Edge Feature: (a) Original map; (b) Decomposed map; . .	28
4.2	Impact of a Decomposed Map on Stopping Distance.	29
4.3	Factory Floor: (a) Original; (b) Decomposed;	33
4.4	Laboratory Map: (a) Original; (b) Decomposed;	37
5.1	Updated System Architecture	40
5.2	Top-Level State Machine of the Action Handler	40
5.3	Precondition Diagram	43
5.4	Example of a Door Opening on RViz	44
5.5	Door Simulation to Open a Door	44
5.6	Simple Map Created to Evaluate the Action Handler	46
5.7	Simple Scenario with and without Precondition and Postcondition: step-by-step progression (a-j).	47
5.8	Robots' Positions in Performance Benchmark: (a) Initial Position; (b) Target Po- sition;	48
5.9	Comparing Overall Execution Time	50
5.10	Robots' Positions in Stress Test: (a) Initial Position; (b) Target Position;	51
5.11	Increased Number of Robots in a Room	52

List of Tables

4.1	Comparison of The Plan Time and The Number of Edges in The Large Factory Floor Map	34
4.2	Comparison of The Plan Time and The Number of Edges in The Laboratory Map	35
4.3	Comparing Task Execution	36
5.1	Execution Times With and Without the Action Handler (in seconds)	49
5.2	Number of robots in the system and their impact	50

List of Acronyms

AGV	Autonomous Guided Vehicles
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
FIFO	First-in First-out
FSM	Finite-state Machine
ID	Identification
iiLab	Industry and innovation lab
INESCTEC	Institute for Systems and Computer Engineering, Technology and Science
KPI	Key Performance Indicator
PDDL	Planning Domain Definition Language
ROS	Robot Operating System
STRIPS	Stanford Research Institute Problem Solver
TEA*	Time Enhanced A*
UML	Unified Model Language
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

1.1 Context

The rapid advancements in robotics and automation technologies have precipitated the necessity for sophisticated fleet coordination systems, particularly within industrial and service environments. These systems become a key component [1] in facilitating the operation of automated guided vehicles (AGVs), ensuring efficiency, safety, and scalability. These systems' applications encompass various domains, including manufacturing and logistics. In these domains, the seamless coordination of multiple agents is imperative to optimise workflows, optimise resource maximisation, and minimise utilisation. For these reasons, AGVs are a viable option to reduce the cost of pick-and-place operations in warehouses, which account for 55% of the daily cost [2].

As the scale and complexity of these deployments increase, the coordination of the behaviour of multiple robots within a shared space becomes a significant challenge [3]. Robots must demonstrate the capacity for efficient navigation and the capability to interact safely with both their immediate environment and each other. In addition, they must be able to execute tasks while accommodating varying levels of urgency and resource requirements [4]. This level of coordination necessitates more than simply reliable navigation, it demands intelligent, adaptive, and context-aware management systems.

Fleet coordination systems are intricate software architectures that oversee many operations, including global route planning, task allocation, collision avoidance, and real-time oversight of agent status [5]. These systems are commonly structured as centralised, decentralised, or hybrid structures, each introducing distinct trade-offs in communication overhead, fault tolerance, and scalability. It is imperative that they also manage shared resources, a task often addressed through the use of concurrency control techniques such as locks, mutexes and semaphores [6]. The mechanisms mentioned above guarantee mutual exclusion in critical regions, thereby mitigating the risk of deadlocks and ensuring the maintenance of orderly robot interactions.

1.2 Motivation

In this context, the TEA* algorithm [7] [8], an extension of the traditional A* algorithm, has emerged as a valuable approach for temporal coordination in multi-robot systems. Incorporating the temporal dimension into the planning process by TEA* facilitates the avoidance of conflicts in space and time, thereby enabling the execution of safe and efficient paths in densely populated environments. TEA* introduces mechanisms such as wait and give-way conditions to manage conflicts dynamically. However, despite these strengths, the model operates under simplifying assumptions, such as uniform edge traversal times and negligible external action costs. When the model is applied in realistic, heterogeneous environments, these assumptions can become bottlenecks.

This limitation becomes more pronounced when robots interact with shared resources or navigate through choke points, such as narrow corridors or access-controlled areas. Without a mechanism for the discrete allocation of temporal costs or for anticipating external actions, the coordination system may be subject to frequent recalculations of the path, an increase in idle times, or even an emergency stop due to deadlock.

Therefore, addressing these limitations is not merely an optimisation concern. It is a fundamental requirement for bridging the gap between abstract, simulation-driven planning frameworks and robust, real-world deployment of autonomous mobile robot fleets. It is imperative that TEA* is augmented with mechanisms that reflect these realities, if the full potential of multi-robot systems in practical large-scale applications is to be realised.

1.3 Objectives

The simplifications facilitated by TEA* can potentially compromise performance and reliability in multi-robot coordination systems. These limitations are particularly problematic in dynamic, shared environments where interactions with physical infrastructure are frequent and timing discrepancies can lead to system-wide inefficiencies or deadlocks.

To surmount the issues mentioned above, this dissertation puts forward two modular enhancements to extend the capabilities of TEA* while preserving its core structure and integration within existing fleet coordination frameworks:

- **Decomposer Submodule:** Introduces temporal discretisation into the environment graph by dividing edges based on traversal duration and a globally optimised time step. This enables more accurate synchronisation between agents, improves planning granularity, and reduces the likelihood of cascading replanning events caused by uneven edge costs.
- **Action Handler:** Implements a logic-based mechanism for managing time-sensitive interactions with shared resources using preconditions and postconditions. This allows robots to anticipate and coordinate with infrastructure elements such as doors, reducing idle time and preventing conflicts in bottleneck areas.

The objective of these components is to establish a connection between theoretical planning assumptions and the requirements of real-world deployment. This enables more robust, efficient and scalable fleet coordination in complex environments.

1.4 Contributions

One article was derived from the contributions of this work, more specifically, the Decomposer Submodule:

- F. Cardoso, D. M. Matos, M. Brilhante, *et al.*, “Enhancing mobile robot navigation: A graph decomposition submodule for tea*,” in *2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2025, pp. 152–157. DOI: [10.1109/ICARSC65809.2025.10970181](https://doi.org/10.1109/ICARSC65809.2025.10970181) [9]

1.5 Document Structure

The dissertation is structured into five additional chapters besides the introduction, Chapter 1. Chapter 2 reviews the state of the art in multi-robot coordination, covering key concepts such as state machines, deadlocks, action control mechanisms (including locks and semaphores), coordination models, and path planning approaches. Chapter 3 describes the baseline fleet coordination system developed by INESC TEC, focusing on its modular structure and implementing the TEA* algorithm. Chapter 4 introduces the Decomposer submodule, detailing the time discretisation methodology, edge splitting process, and the results of performance tests. Chapter 5 presents the Action Handler, explaining how it incorporates preconditions and postconditions to manage shared resources like doors, and evaluating its impact on execution time and scalability. Finally, Chapter 6 concludes the dissertation by summarising the main contributions and outlining potential directions for future work.

Chapter 2

State-of-the-art

The successful deployment of multi-robot fleet coordination systems in real-world environments heavily depends on integrating multiple interrelated technologies. This chapter provides a comprehensive overview of the existing concepts, methods, and system architectures that underpin the field to establish a solid foundation for the proposed solution. The chapter 2 commences with an examination of state machines, which are recognised as a foundational tool for modelling robot behaviour. This is followed by an exploration of deadlocks as a risk inherent in multi-agent systems, and the control mechanisms that have been developed to prevent such occurrences. Subsequent to this, Action control strategies, including locks, semaphores, and the utilisation of preconditions and postconditions, are examined to provide a contextual framework for the role of the Action Handler within such systems. Then the chapter undertakes an evaluation of centralised, decentralised, and hybrid architectures for coordinating autonomous agents, considering their respective strengths and limitations. Finally, it presents a comparative analysis of grid-based and graph-based path planning techniques, highlighting their implications for computational efficiency and planning precision.

This state-of-the-art review establishes the foundation for the design choices and methodologies introduced in the subsequent chapters and used to develop the Decomposer submodule and the Action Handler.

2.1 State Machines

State machines, or finite state machines (FSMs), are essential in today's technology. They are widely used in software engineering, embedded systems, and robotics, providing a means of modelling the behaviour of systems through a series of defined states and transitions, thereby ensuring predictable and manageable system behaviour. Given their extensive use in many fields, there are many variations, yet all adhere to the same fundamental principles. In this explanation, a semantic definition will be employed by the Unified Modelling Language (UML) norm [10].

A state represents a condition or situation in the lifecycle of an object where it satisfies some condition, performs some activity, or waits for an event. States are typically represented as rounded

rectangles, including entry, exit, and ongoing activities. In the UML, a vertex is a general term that encompasses both states and pseudo-states (such as initial or final states).

Figure 2.1 illustrates a state machine by the previously described UML definition.

A transition is a directed relationship between two states, representing the movement from one state to another in response to an event. Transitions are depicted as arrows between states and may include triggers, guards, and effects. An edge, on the other hand, refers to the connection or path between two vertices (states or pseudo-states).

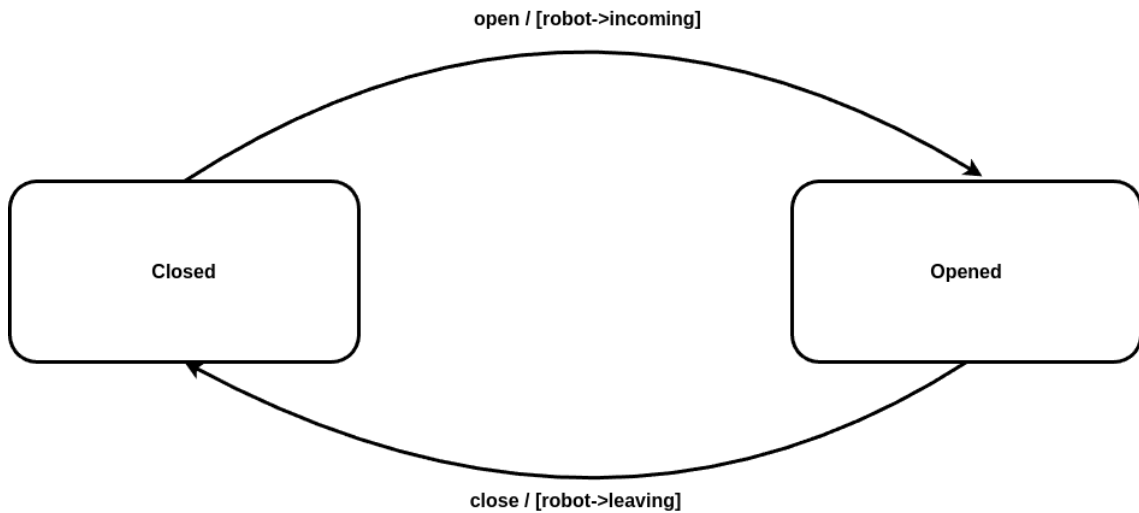


Figure 2.1: Simple State Machine Example

2.2 Deadlocks

To comprehend section 2.3, it is imperative to introduce the fundamental concepts of deadlocks.

In the context of an action handler, a deadlock scenario can arise when multiple robots endeavour to access a resource (an action) in a sequence such that the first robot requires the resource held by the second robot to proceed, and the second robot necessitates the resource held by the first robot to continue.

According to [11], the occurrence of a deadlock is contingent upon the fulfilment of the following four conditions:

- **Mutual exclusion:** This condition ensures that a resource is allocated to only one process at a time, making it non-shareable. If another process requests the resource, it must wait until the current holder releases it.
- **Hold and wait:** A process can hold one or more resources and simultaneously request additional resources currently held by other processes. This situation increases the risk of resource contention as processes accumulate resources without releasing them.

- **No preemption:** This condition ensures that once a process has acquired a resource, it cannot be forcibly taken away by the system; the process must release it voluntarily. This provides stability, but can lead to deadlock if a process refuses to release a needed resource.
- **Circular wait:** A closed chain of processes exists, where each process waits for a resource that the next process holds. This circular dependency creates a loop where no process can proceed, forming the basis for a deadlock.

Suppose an action handler is not meticulously designed. In that case, a robot may have difficulty opening a door because another robot may be attempting the same action on the opposite side of the door. In this scenario, the first robot successfully opens the door, yet it cannot traverse it due to the presence of the other robot. This happens because opening a door is a temporal cost not considered in the TEA* algorithm. Consequently, the first robot cannot release the door to the other robot for utilisation.

In the following Figure 2.2, the red robot is trying to hide in the left node after the door, but because the door took a unit of time (K) to open, which was not expected by the TEA*, it led to a deadlock situation. This scenario poses a significant challenge because it requires the system to avoid or recover from a deadlock situation.

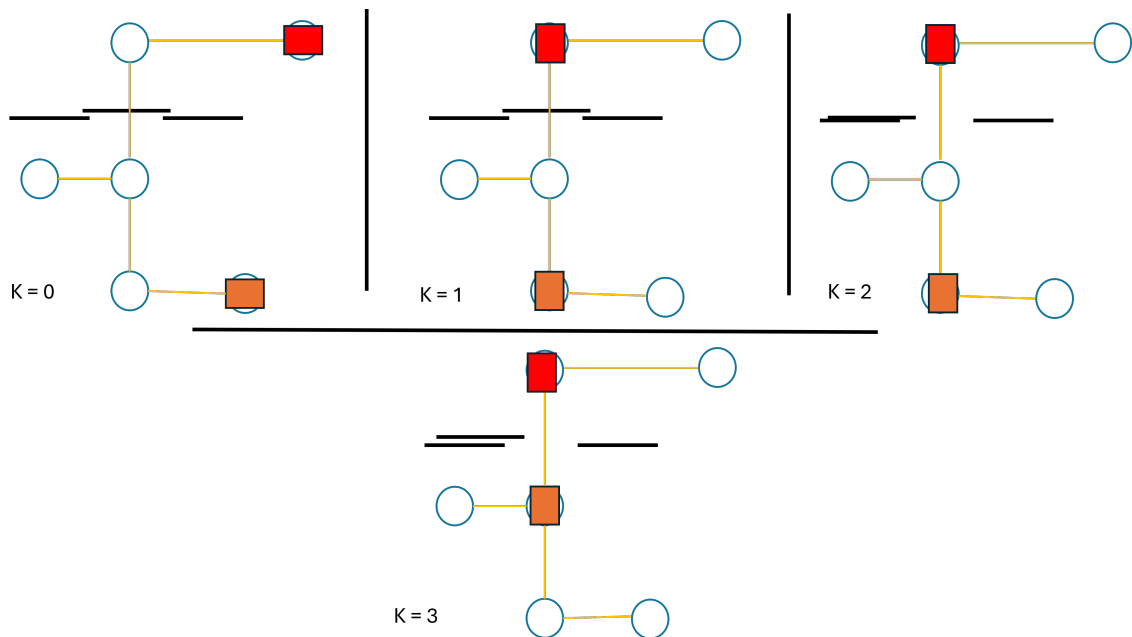


Figure 2.2: Deadlock Situation

2.3 Action Control Methods

In an autonomous factory floor, it is not uncommon for multiple robots to attempt to access the same door or elevator, resulting in downtime and, in some cases, robot crashes. To mitigate this issue, control methods can be implemented in the Action Handler to ensure mutual exclusion of actions and to avert deadlocks.

2.3.1 Locks and Mutexes

Locks are the most straightforward method for controlling concurrent access, allowing access to critical sections mutually exclusively [12]. That is to say, if a robot is accessing an action or edge, a lock would ensure that no one else can access that. A robot must acquire the lock before entering the critical section and release it upon exiting.

Mutexes represent a specialised form of lock that ensures mutual exclusion and provides mechanisms to avoid busy-waiting [12]. This occurs when a program cannot proceed due to a stuck thread trying to access a critical section, preventing the CPU (Central Processing Unit) from allocating resources to other threads or processes.

2.3.2 Semaphores

Semaphores augment locks' functionality by facilitating mutual exclusion and resource management. They employ a counter to regulate access to shared resources [12].

- **Binary Semaphores:** Function similarly to mutexes, ensuring mutual exclusion [12].
- **Counting Semaphores:** Manage a fixed number of available resources. For instance, a counting semaphore with a count of N allows up to N robots to access a resource simultaneously [12].

Semaphores are of particular utility in the context of elevator operations. If several robots attempt to call an elevator, the elevator's finite spatial capacity necessitates establishing a maximum number of robots that can be accommodated concurrently.

2.3.3 Precondition and Postcondition

A precondition is defined as a constraint or condition that must be satisfied before the occurrence of a transition [12]. This concept is frequently utilised as a guard condition, whereby the satisfaction of the precondition is indicated by the presence of a square bracket enclosing the condition in question, adjacent to the transition in question.

A postcondition is defined as a constraint or condition that must be true following the completion of a transition [12]. It describes the state of the system after the transition.

The precondition and postcondition will be implemented in the graph representation already created in the fleet coordination system. To illustrate, if an agent traverses an edge and the battery level is below a specified threshold, the system will trigger a flag. Furthermore, if the same agent approaches the task's final edge, the system will plan a new path to the nearest dock for charging. This is an example of a precondition and postcondition in the fleet coordinator system.

2.3.4 Action Control in the Literature

In the current literature, several strategies have been developed for managing multi-robot coordination [13].

The right-hand rule [14] is a simple yet effective method for resolving conflicts by enforcing a deterministic rule, where robots yield to others on their right, thereby minimising contention and deadlocks. However, its simplicity can lead to suboptimal robot paths and increased travel times in complex scenarios.

Another strategy is priority planning [15], which assigns hierarchical priorities to robots, enabling structured decision-making during resource contention. The primary advantage of this approach is its ability to ensure predictability and to reduce the potential for conflict. However, it should be noted that this method can sometimes result in lower-priority robots experiencing delays or starvation.

The concept of detour points [16] provides a degree of flexibility by allowing robots to dynamically deviate from congested paths, thereby ensuring a more efficient flow of traffic. While this approach does reduce the likelihood of deadlocks, it may also increase the overall travel distance and the computational overhead required for path replanning. The integration of these strategies establishes a multifaceted framework that seeks to optimise the balance between simplicity, efficiency, and computational cost in the context of multi-robot trajectory planning.

Most action planners employed first-order logical relations, such as precondition and postcondition, to represent actions and the sequence of steps necessary to fulfil a task. These action planners typically employ the STRIPS (Stanford Research Institute Problem Solver) method and PDDL (Planning Domain Definition Language) as a foundational framework [17].

In the STRIPS methodology [18], each action is defined as an operator consisting of three components: preconditions, effects to be added, and effects to be deleted. The preconditions delineate the logical conditions that must be satisfied for an action to be applicable. At the same time, the effects describe the subsequent alterations to the world consequent to the execution of the action.

The STRIPS planning process entails identifying a sequence of actions that facilitate the transformation of a specified initial state into a predetermined goal state. The global state is characterised by a set of logical predicates, with the planning algorithm undertaking a systematic exploration of the feasible action sequence space to derive a valid plan that achieves the objective while maintaining logical consistency. The Action Handler adapts the precondition and postcondition used in this method due to its capacity to minimise the necessity for alterations to the prevailing TEA* algorithm.

Alternatively, PDDL [19] is a domain-independent language designed to describe planning problems in a standardised, machine-readable format. PDDL unifies and benchmarks planning research by separating the description of a planning problem into two components: the domain file and the problem file. The domain file is responsible for defining the set of predicates and actions, including their parameters, preconditions, and effects, often following the STRIPS-style semantics. The problem file delineates the initial state of the world and the desired goal conditions.

PDDL can support advanced features such as temporal planning, numeric fluents and conditional effects. This facilitates the representation of complex and dynamic domains. It functions as the principal interface for numerous contemporary planning engines, enabling comparison and

evaluation of planning algorithms across standardised tasks. This is the most widely accepted approach within the AI (Artificial Intelligence) planning community; however, its focus is on AI, and it requires a layer to read and interpret the input, as it is unsuitable for more traditional methods, such as the TEA*.

2.4 Possible Architectures of a Fleet Coordination System

A fleet coordination system's architecture considerably impacts efficiency, scalability, and robustness because how control and decision-making are distributed will determine whether the system adheres to a decentralised, centralised, or hybrid coordination model. Each architectural design exhibits strengths and challenges, particularly in communication, task allocation, and path planning.

A concise exposition will be provided regarding the two predominant architectures employed within the industry to facilitate a more profound comprehension of Chapter 3.

2.4.1 Decentralised Coordination

Decentralised coordination, as shown in Figure 2.3, emerges as a viable approach for the design of a fleet management system. It is the most straightforward method that entails enabling all agents to perpetually plan their trajectories, taking into account the constraints imposed on them by the trajectories of other robots [20]. This approach eliminates the need for a centralised control unit, a hallmark of centralised systems, thereby reducing computational demands and communication bottlenecks.

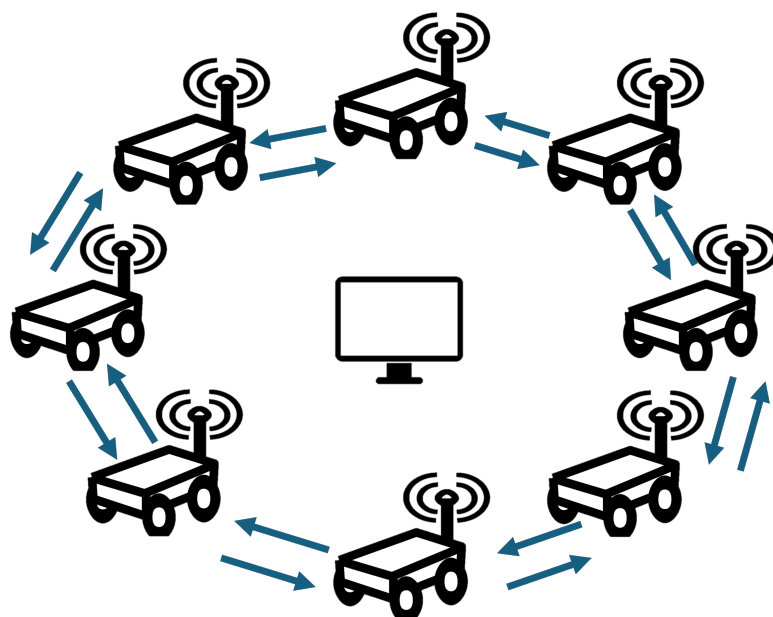


Figure 2.3: Decentralised System Exemple

A plethora of strategies have been devised to implement decentralised control for AGVs [21], one strategy uses agent-based architectures, which empower vehicles to function as intelligent agents capable of decision-making and interaction with other agents. Another notable strategy involves vehicles periodically broadcasting their states and treating others as static obstacles for path planning, ensuring collision avoidance and deadlock prevention through mutual exclusion zones and priority rules.

Additional methods address broader challenges, such as operating in partially known environments, managing task assignments, and ensuring safe trajectories under communication constraints. These approaches enable decentralised systems to function effectively in dynamic industrial settings. Environmental uncertainty and vehicle interaction complexity are prevalent in such environments, but optimisation is more challenging because decisions are made locally and may not account for the system-wide context.

A recent contribution [22] proposes an enhanced decentralised cooperative multi-agent pathfinding approach, customised for robots operating within constrained communication ranges. In this method, agents communicate exclusively with proximate neighbours within a predetermined radius. This facilitates coordination at the local level to address conflicts and avert collisions. The approach under discussion integrates dynamic path replanning, local motion coordination, and strategies to prevent deadlocks. It can be concluded that, in essence, the system demonstrates elevated scalability, thereby facilitating scenarios encompassing up to 1000 agents. In addition, the system has been shown to demonstrate robust performance in a variety of standard benchmark environments.

The distinguishing feature of this approach is its emphasis on real-world applicability. In contradistinction to numerous decentralised models that presuppose complete observability or deterministic motion, this methodology acknowledges the constraints imposed by incomplete information and sporadic communication. Agents can make decisions based solely on locally available data, a factor which engenders increased robustness in unpredictable or constrained environments.

Integrating decentralised coordination mechanisms into fleet management systems can enhance resilience and scalability, particularly when infrastructure or communication reliability is limited. Furthermore, the distribution of decision-making responsibilities reduces reliance on a central controller, eliminating single points of failure and enabling a more balanced computational load across the system.

2.4.2 Centralised Coordination

Conversely, centralised coordination, as shown in Figure 2.4, relies on a centralised control unit, which can be a robot or an external receptor such as a computer. This unit manages critical tasks, including mission scheduling, path planning, and motion coordination. The central unit monitors vehicle positions, transmits control commands, and oversees real-time system states.

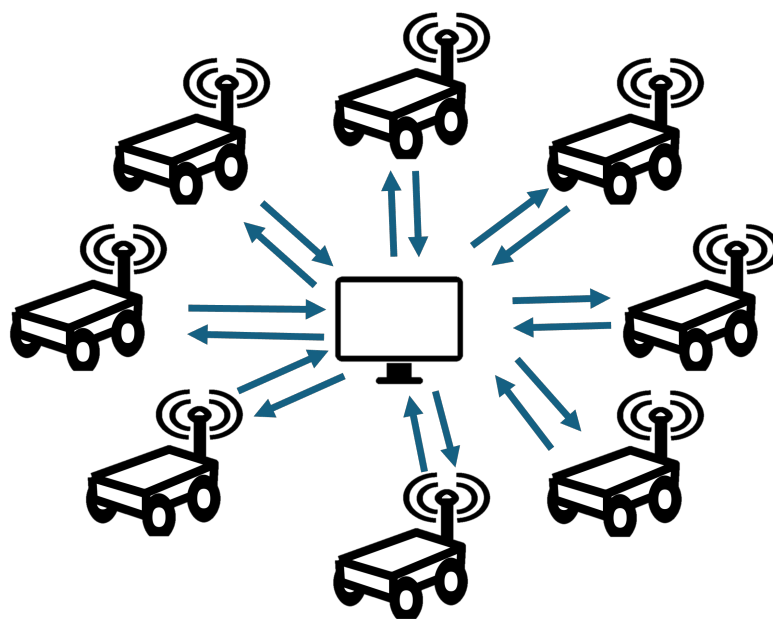


Figure 2.4: Centralised System Exemple

The centralised approach allows for complete motion planning and optimal coordination, ensuring high positioning accuracy, predictable behaviour, and enhanced safety. The benefits render centralised systems popular for practical AGV applications, particularly in environments requiring precise and efficient operations [21].

Centralised systems ensure unified decision-making and better optimisation because all the information and data are housed in one system. However, this approach introduces a single point of failure in the central hub. To mitigate this risk, additional security layers must be implemented.

This paradigm is further explored in [23], in which the authors propose a comprehensive centralised fleet management framework designed explicitly for AGV systems operating in industrial environments. The system under discussion features a modular architecture incorporating task allocation, traffic management, and route planning as separate components within the centralised control unit. The framework utilises a global perspective of the environment and the entire fleet to optimise decision-making and streamline operations.

The authors also emphasise the significance of incorporating simulation tools into the system testing and deployment framework. Integrating a centralised controller with a digital twin of the physical environment facilitates the validation and adjustment of the system before live operation, thereby ensuring reliability and mitigating implementation risks.

The evaluation results presented in the study demonstrate the effectiveness of the proposed framework. A series of simulations were conducted within a factory-like environment. Compared to non-coordinated or purely reactive systems, these simulations significantly enhanced AGV throughput, reduced idle times, and optimised overall resource utilisation. Moreover, the authors observe that centralised control facilitates superior management of complex logistics operations and scalability in structured environments.

While the system demonstrates robust performance in controlled settings, the paper also acknowledges that reliance on a centralised architecture introduces vulnerabilities, particularly concerning system robustness and fault tolerance. Consequently, subsequent endeavours are focused on augmenting the framework’s resilience, potentially through incorporating redundant control units or integrating decentralised fallback mechanisms.

2.4.3 Hybrid Coordination

A hybrid coordination approach, illustrated in Figure 2.5, integrates the strengths of both centralised and decentralised paradigms. In this architecture, high-level decision-making tasks such as global path planning, task allocation, and mission supervision are managed by a centralised unit. Meanwhile, lower-level behaviours, including local navigation, obstacle avoidance, and execution of assigned tasks, are handled autonomously by the individual robots.

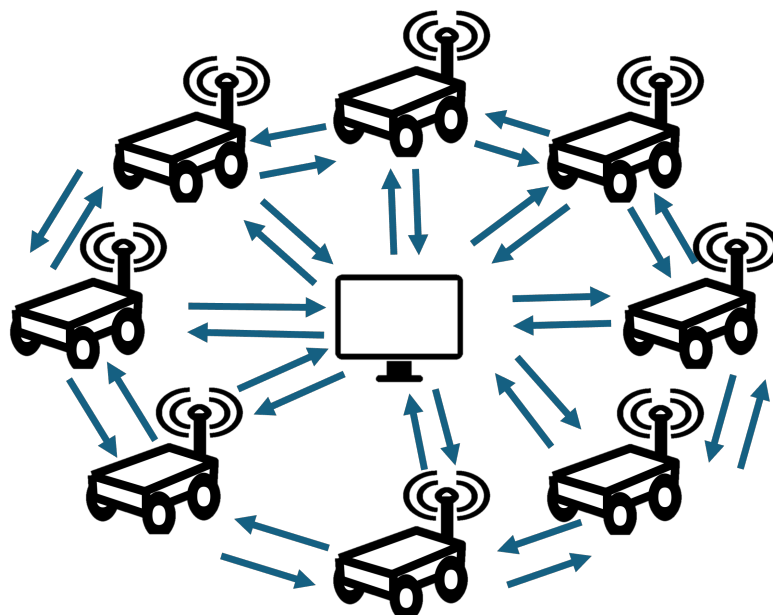


Figure 2.5: Hybrid System Exemple

This strategy aims to harness decentralised systems’ agility, adaptability, and fault-tolerance characteristics while benefiting from the global situational awareness and coordinated resource allocation of centralised control. Such an approach enhances operational efficiency, supports dynamic task reallocation, and allows for system-wide optimisation during mission execution [24].

Nevertheless, the adoption of a hybrid model introduces notable challenges and trade-offs. As inter-agent coordination’s complexity increases, so does the communication burden on the network infrastructure. This can result in latency, data congestion, or even partial system failures in bandwidth-limited environments. Moreover, hybrid architectures often struggle to be generic and modular, making them difficult to adapt across varied operational domains without extensive changes [25]. Therefore, the balance between centralised oversight and decentralised autonomy

must be carefully managed to ensure the system remains scalable, maintainable, and robust under real-world constraints.

Future developments in hybrid coordination may benefit from using adaptive communication protocols, scalable middleware solutions, and learning-based strategies to dynamically adjust the degree of centralisation according to mission demands and environmental conditions.

2.5 Grid-Based and Graph-Based Path Planning

One of the crucial components of autonomous navigation is path planning, which enables robots to find optimal and feasible routes from their initial position to the desired goal. The two most common approaches to this are graph-based and grid-based. These strategies employ diverse methodologies for modelling the environment and computing the path, with their efficiency contingent on the specific application.

The primary distinctions between these approaches pertain to how they represent the spatial domain. Specifically, graph-based methods employ a network model, characterised by nodes and edges. In contrast, grid-based methods utilise a discretised representation, characterised by cells and grids, which can be seen in Figure 2.6. Each method has specific advantages regarding path quality, computational complexity, and suitability for different environments.

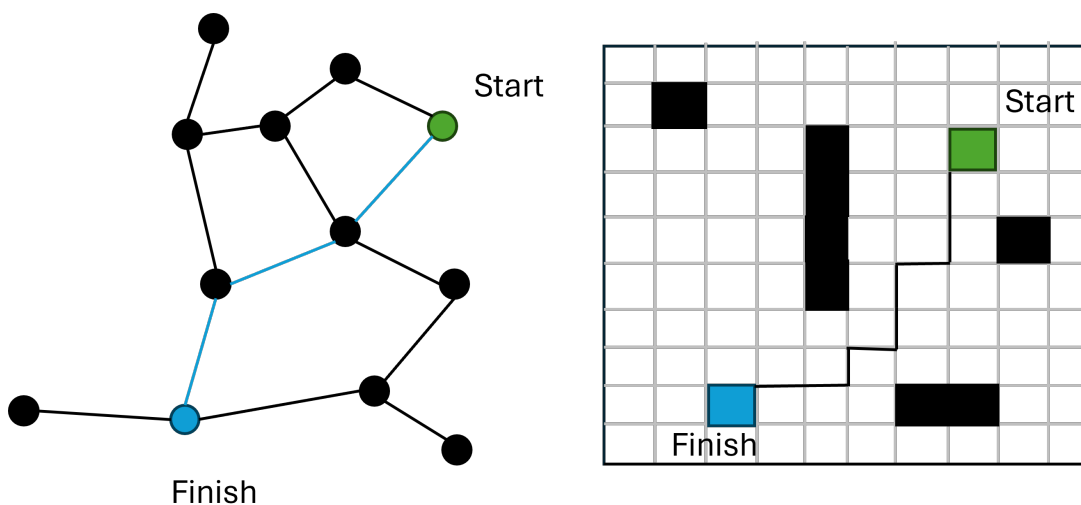


Figure 2.6: A Graph-Based Map and a Grid-Based Map

2.5.1 Grid-Based Path Planning

Grid-based path planning involves discretising the world into a two- or three-dimensional grid of cells of equal size. Each cell is marked as free or occupied, depending on whether an obstacle or robot is in that position. This facilitates the implementation of algorithms such as A* and Dijkstra's.

A significant benefit of this approach is its capacity to correspond to the grid and real spaces, which facilitates high-performance collision detection and environmental updates. The adaptation of A* that employs directional heuristics or environmental frameworks, such as V-Graph, has been proven to improve path quality without compromising the ease and stability of the grid [26] [27].

However, it should be noted that narrow passages or thin obstacles are not discernible to low-resolution grids; however, high-resolution grids incur substantial memory and computational demands. It has been demonstrated that low-resolution grid paths are prone to instability and may require post-processing. Moreover, grid-based methods are inherently challenging for continuous-time planning due to their tendency to represent motion discretely unless extended with specialised frameworks [28], as shown in Figure 2.7.

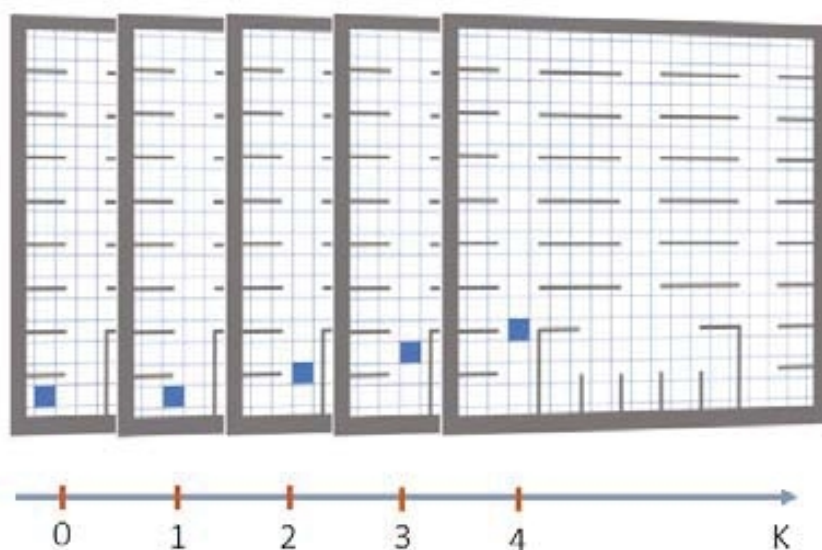


Figure 2.7: TEA* Algorithm Applied to a Grid-Based Map [28]

2.5.2 Graph-Based Path Planning

Graph-based methods use connected nodes to model the environment, representing reachable states. These nodes may be sampled from free space or predefined waypoints, with edges representing valid transitions. Algorithms such as Dijkstra's and A* operate efficiently on these graphs, particularly when enhanced with heuristics.

A significant benefit of this approach is its flexibility and scalability. Graphs can be sparse, which is advantageous in terms of reducing computational burden, and they can be augmented with semantic or temporal information. This approach is especially efficacious in large or multi-level environments where maintaining a full grid would be computationally prohibitive.

Graph-based methods also demonstrate excellence in multi-agent systems, facilitating coordination through constraint reasoning or prioritised planning mechanisms. In scenarios involving

multiple robots, using shared graphs facilitates decentralised and centralised planning methodologies, incorporating conflict detection and resolution capabilities. This approach enhances the system's resilience against potential deadlocks and collisions [29].

Notwithstanding these strengths, graph-based models rely heavily on pre-processing and environment abstraction. Constructing an informative and navigable graph requires substantial upfront computation or domain knowledge.

Temporal reasoning is also less intuitive in classic graph-based models, but this can be integrated through specialised enhancements, such as incorporating time windows or motion durations into node costs. These extensions facilitate planning in dynamic environments or scenarios involving deadlines and synchronisation [28].

2.5.3 Summary

This chapter comprehensively reviewed the concepts, architectures, and planning strategies relevant to multi-robot coordination. It began with finite state machines and the risks posed by deadlocks in shared-resource environments. Several action control mechanisms were introduced, including locks, semaphores and logical constraints. Particular emphasis was placed on preconditions and postconditions to regulate access to shared resources, such as doors and elevators. These mechanisms were employed within the designated Action Handler. Moreover, a comparison was made between centralised, decentralised, and hybrid architectures, which revealed the trade-offs between scalability, reliability, and optimisation. In conclusion, the evaluation of grid-based and graph-based path planning methods demonstrated that graph-based approaches are more appropriate for dynamic, large-scale, multi-agent systems. These findings established the foundation for introducing the fleet coordination system and its constituent modules.

Chapter 3

Fleet Coordination System

The INESC TEC-developed system will be utilised to assess the impact of the Action Handler and the Decomposer submodule effectively in a fleet coordination system. This system is known for its high flexibility and seamless integration with the Stage simulation, which enables result validation in a simulated environment before real-world deployment. Furthermore, it offers robust fault tolerance, allowing it to handle errors, replan paths, cancel and assign new tasks, and coordinate multiple agents dynamically [25]. The following modules constitute the system:

- **Graph Pre-Processor:** Detection of parallel edge and choke point identification
- **Path Planner:** Try to plan a path using the TEA* algorithm
- **Agent Supervisor:** Synchronise the robots to avoid collisions and account for delays
- **Graph Server:** Sends the graph used by all the system robots

A graphical representation of the communication and connections between modules can be observed in Figure 3.1.

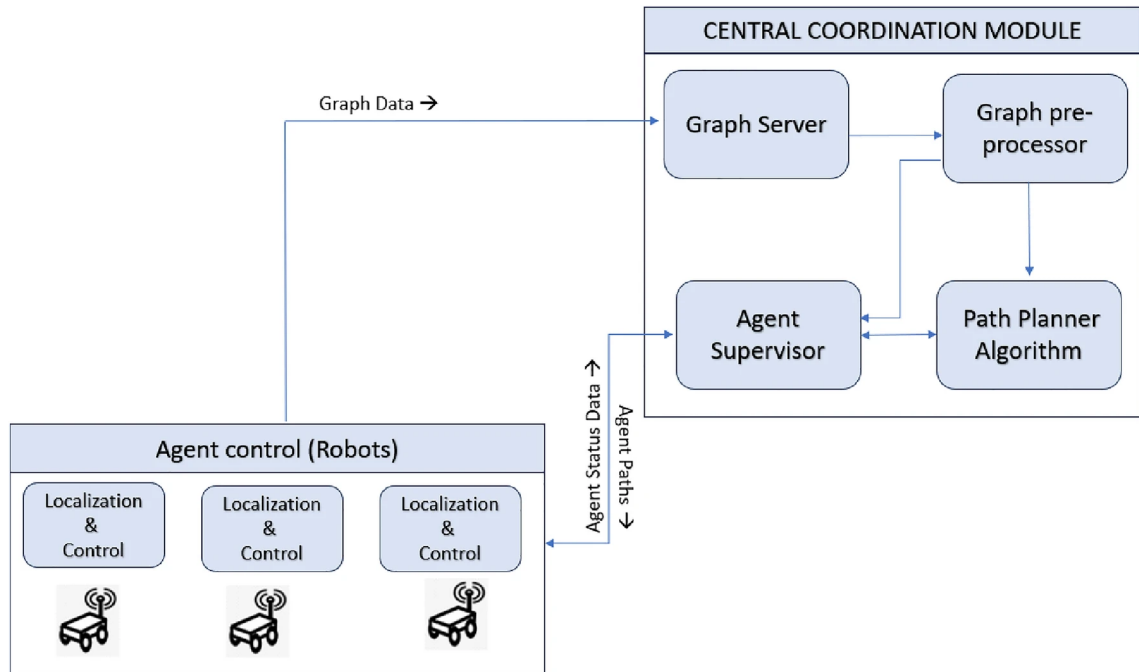


Figure 3.1: System Overview [25].

The Graph Pre-Processor is anticipated to be integrated into the Graph Server. However, it is currently being utilised as a separate entity for development and to facilitate comprehension.

3.1 Graph Representation

The fleet coordination system employs graph theory to plan optimal routes and to represent them in a visual format [25]. A graph is a structure comprising a set of nodes, or vertices, connected by links, or edges, as shown in Figure 3.2. The properties of vertices and edges define their respective behaviours and values. A vertex may be conceived of as a decision point, at which the agent may elect to traverse a particular edge connected to him [25]. This concept resembles an FSM.

- **ParamF**: Forwards parameter used in spline curve definition (see Fig. 3.3)

In these properties, the concepts of precondition and postcondition will be introduced to positively impact the system's overall performance.

3.1.1 Bézier Curves

The depiction of a graph map is achieved through parametric curves, particularly Bézier curves. This approach is superior to straight line segments, facilitating a more gradual, straightforward, expeditious, dependable, and viable trajectory. This method minimises interruptions and decelerations, thereby ensuring the consistency of the robot's dynamics [30]. Figure 3.3 illustrates constructing a Bézier curve.

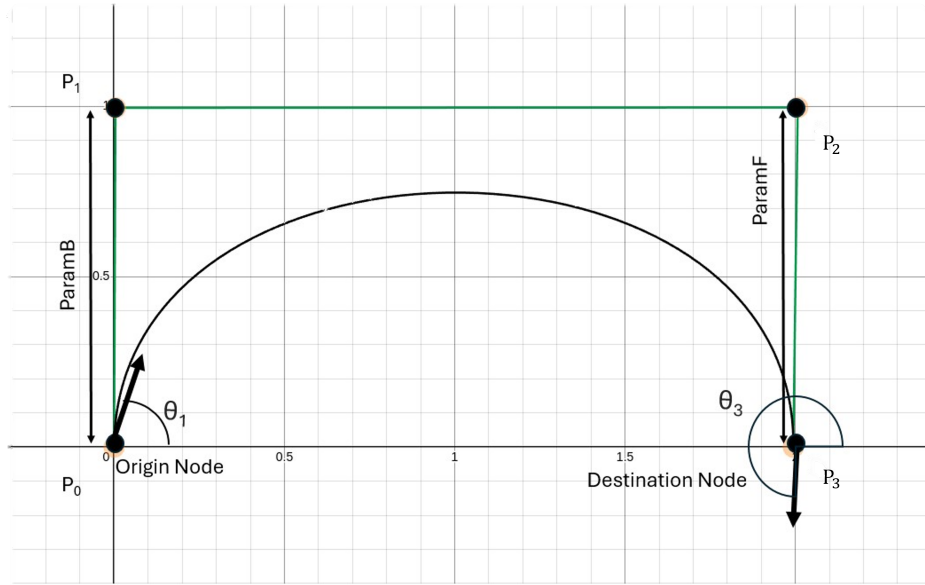


Figure 3.3: Bézier Curve Example

In the provided example, the following core components of the Bézier curve are identified: the points P_0 , P_1 , P_2 , and P_3 as well as the angles theta. The angle θ_1 establishes the vertex's orientation and the curve's initial segment. Conversely, θ_3 dictates the final segment of the curve. Utilising the θ_1 , θ_3 , $ParamB$, and $ParamF$, in conjunction with the specified points, enables the calculation of the Fx and Fy matrices through the following set of equations 3.1 3.2.

$$\begin{aligned} x(\lambda) &= a_x t^3 + b_x t^2 + c_x t + x_0, & 0 \leq t \leq 1 \\ y(\lambda) &= a_y t^3 + b_y t^2 + c_y t + y_0, & 0 \leq t \leq 1 \end{aligned} \quad (3.1)$$

where,

$$c_x = 3 \times (x_1 - x_0) \quad (3.2)$$

$$b_x = 3 \times (x_2 - x_1) - c_x$$

$$a_x = x_3 - x_0 - c_x - b_x$$

$$c_y = 3 \times (y_1 - y_0)$$

$$b_y = 3 \times (y_2 - y_1) - c_y$$

$$a_y = y_3 - y_0 - c_y - b_y$$

With those equations, it is possible to calculate the length of each edge and subsequently divide the result by the velocity at which the robot traverses the edge. This yields the edge cost employed in the TEA* algorithm.

3.1.2 Casteljau's algorithm

The Equations 3.1 and 3.2, the polynomial representation cannot divide the edge. Consequently, Casteljau's algorithm was implemented in the Edge Splitting phase to divide the edge and maintain the shape and speed.

The Casteljau's algorithm utilises the control points of the original edge, employing an interpolation technique between these points [31]. The equation to which this interpolation refers can be found in equation 3.3, in which t denotes the percentage of the edge to be divided.

$$P_{01} = (1 - t)P_0 + tP_1 \quad (3.3)$$

$$P_{12} = (1 - t)P_1 + tP_2$$

$$P_{23} = (1 - t)P_2 + tP_3$$

Subsequently, a new interpolation (Eq. 3.4) is applied to P_{01} , P_{12} and P_{23} .

$$P_{012} = (1 - t)P_{01} + tP_{12} \quad (3.4)$$

$$P_{123} = (1 - t)P_{12} + tP_{23}$$

With the points P_{012} and P_{123} , it is possible to find a division point B in the original by applying Equation 3.5.

$$B(t) = (1 - t)P_{012} + tP_{123} \quad (3.5)$$

With **Subdivision at t :**

- First (left) sub-curve control points:

$$P_0, \quad P_{01}, \quad P_{012}, \quad B(t)$$

- Second (right) sub-curve control points:

$$B(t), \quad P_{123}, \quad P_{23}, \quad P_3$$

Figure 3.4 provides a visual breakdown of Casteljau’s algorithm in action. On the left side of the figure, the complete cubic Bézier curve is shown, with its original control points P_0, P_1, P_2, P_3 , and the intermediate points $P_{01}, P_{12}, P_{23}, P_{012}, P_{123}$ used during the interpolation process. The red point marked $B(t)$ represents the evaluated point on the curve at the given parameter t , which corresponds to the division point.

In the centre, the figure isolates the **first half** of the curve, reconstructed using the control points $P_0, P_{01}, P_{012}, B(t)$. This segment maintains the original curve’s shape from the start up to $B(t)$, reflecting how the left sub-curve is formed using only the interpolated points on that segment.

The right side of the figure shows the **second half** of the curve, built from the points $B(t), P_{123}, P_{23}, P_3$. This part continues smoothly from the division point to the end of the original curve.

The figure clearly demonstrates how Casteljau’s algorithm splits a Bézier curve at any chosen value of t , resulting in two new Bézier curves that perfectly reconstruct the original shape together. This subdivision is essential for maintaining continuity and geometric accuracy during edge splitting.

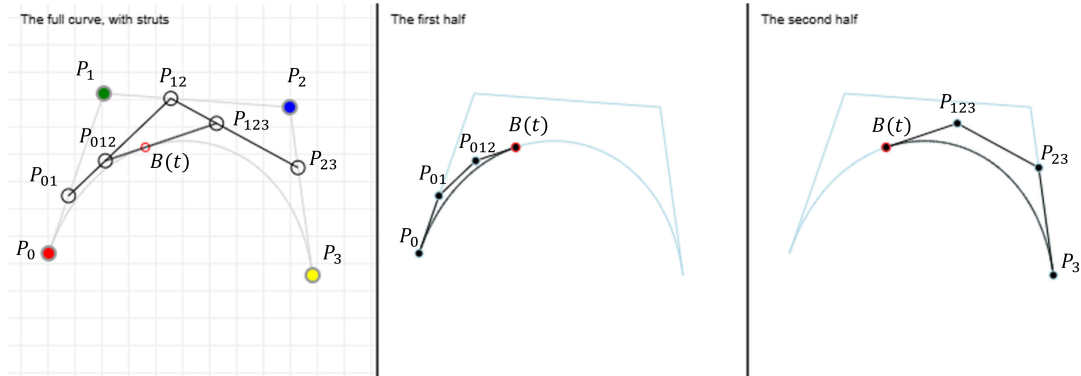


Figure 3.4: Division of a Bézier Curve Using Casteljau’s Algorithm

3.2 Graph Pre-Processor

The Graph Pre-Processor module is responsible for the initial data processing of all information received from the Graph Server. This data set includes the original graph map, the robot’s dimensions, and additional details. The primary functions of this module are as follows:

- **Parallel Edge Detection:** Identifying parallel edges is critical for robot navigation, as these edges, which emerge due to the agent’s size, can lead to potential collisions. The Path Planner leverages a dedicated function to detect and evaluate these edges to mitigate collision risks [25].

- **Choke Point Identification:** Recognising critical edges such as elevators and doors, known as choke points, which can impede flow and create bottlenecks [25].

The Decomposer submodule was integrated into this module, as it is the first to handle the original edge map. Runtime performance is not a concern, since it operates in conjunction with other one-time-use functions, such as parallel edge detection, that only execute when the original map changes. A ROS service was implemented to streamline the process. This service first runs the Decomposer, followed by the parallel path function. This aims to ensure that parallel paths are computed on the decomposed map.

3.3 Path Planner

Efficient path planning is a critical component in the operation of multi-robot systems, especially in dynamic industrial environments. Autonomous robots, such as AGVs, are increasingly utilised for material handling and transportation tasks. However, coordinating numerous robots within a shared workspace poses significant challenges, including avoiding collisions and preventing deadlocks. These problems become more complex in real-time settings, where dynamic obstacles and unexpected delays are prevalent.

3.3.1 Time Enhanced A* (TEA*)

To address these challenges, the Time Enhanced A* (TEA*) [7] [8] algorithm enhances the conventional A* pathfinding algorithm by incorporating a temporal dimension. By considering both spatial and temporal factors, TEA* facilitates the calculation of dynamic and collision-free paths for multiple robots. Adding this layer brings with it two main features [25]:

- **Wait conditions:** one or more robots wait for another robot to finish their planned path and then resume their path
- **Give away conditions:** A robot changes its path for a non-optimised one to let others pass

Figure 3.5 represents the temporal layer used in the TEA* algorithm as a K variable.

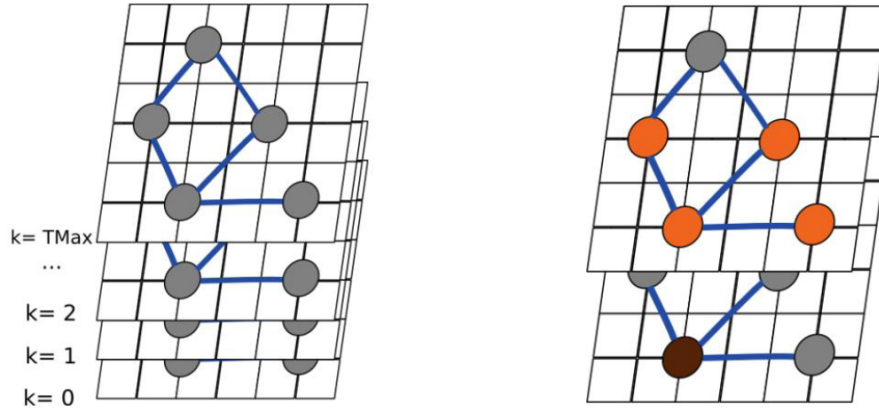


Figure 3.5: Graph Layering for TEA* [32]

TEA* is an innovative methodology that calculates the minimum path over the temporal layers. This calculation is based on the vehicles' positions and the environment's changes. One of the most notable features of this method is its online capability, which allows for constant recalculation of paths in a loop.

The interaction sequence can be explained as follows:

1. Calculate the path for the first robot
2. Updates the position of that robot for each k starting in $k = 0$
3. Blocks the path that the robot takes to avoid collisions
4. Repeats the process for the other robots

The updated version of the TEA* algorithm has been enhanced with some new features. The system has been modified to make the waiting cost dynamic. Consequently, the value is no longer fixed; it is now a variable depending on the situation presented. The implementation of parallel edge detection is also a feature of the system. This function is designed to prevent robots on different parallel edges from colliding due to their size. This algorithm iteration incorporates the dimensions of the robot's footprint to facilitate calculating the path.

3.4 Agent Supervisor

The system has been developed to ensure seamless execution of planned paths by handling both the transmission of trajectory data and the real-time monitoring of robot execution feedback. This

module is critical in maintaining system reliability and safety, particularly in dynamic or partially observable environments.

The module disseminates navigation instructions, which the Path Planner generates, and concurrently receives execution feedback from the Graph Server module, encompassing their present position, status, and any execution delays. Such delays may be attributed to some factors, including unanticipated environmental obstacles, sensor uncertainty, or temporary disruptions in communication.

A fundamental capability of this module pertains to its capacity for proximity analysis, which categorises interactions between robots following the potential for conflict. Should two robots be projected to occupy proximate or shared edges within a specified temporal window, they are designated "Danger Close". Conversely, if their trajectories remain sufficiently isolated, they are considered "Safe".

The system employs the TEA* algorithm to manage temporal coordination, introducing time as a discrete dimension via unit steps. The discrepancy in the execution timing of steps, termed `Delta_steps` [33] [28], is utilised as a metric to assess the synchronisation health of the system. Should the `Delta_steps` for any robot exceed a predefined threshold, particularly those designated as "Danger Close", the module will trigger a replanning request to the Path Planner. This mechanism ensures that trajectory adjustments are made proactively, preventing unsafe interactions.

In scenarios where multiple replanning attempts fail due to persistent conflicts, blocked paths or system-level issues, the Agent Supervisor activates a fail-safe protocol. The protocol in question forces the system into an emergency stop state, initiating an immediate cessation of all movement of the robots. This is done to prevent collisions and further escalation of the issue.

The Agent Supervisor module enables reliable and scalable multi-robot coordination by combining robust communication handling, predictive conflict detection, and safety replanning strategies.

3.5 Graph Server

This submodule plays a critical role in the overall architecture by managing the exchange and interpretation of spatial and functional data between the fleet coordination system and the individual robots. The core responsibility of the system is to act as a communication bridge, receiving the environment map from the robots. The maps delineate more than just the geometric structure of the environment; they also include vertices, edges, and travel velocity. Additionally, they encompass high-level semantic information that delineates the set of discrete actions each robot can perform. Examples of such actions include interacting with environmental elements such as the opening of automatic doors, the operation of elevators, or the engagement of docking stations for charging or loading/unloading purposes [25].

Upon receipt of the above information, the submodule initiates a preliminary processing phase, which analyses the spatial layout and the associated action data. This process entails calculating

traversal costs for each edge, which are determined by the edge's physical distance and estimated execution time.

The system has been designed to support heterogeneous fleets with varying capabilities and localised knowledge. However, for simplicity and to facilitate performance benchmarking during this work's development and evaluation phase, a design choice was made to provide all robots with a uniform map. This approach is designed to ensure that all agents operate on a consistent understanding of the environment, thereby allowing the focus to remain on the effectiveness of the Graph Decomposer and the Action Handler.

3.5.1 Summary

This chapter presents the INESCITEC-developed fleet coordination system, showcasing a modular architecture designed to handle real-time navigation of multiple robots. The system's reliance on a graph-based representation with Bézier curves allowed for smooth path planning and the inclusion of high-level semantics such as velocity constraints and edge costs. The TEA* algorithm enhances the traditional A* algorithm, incorporating a temporal layer to manage coordination over time. However, the TEA* algorithm's assumption of uniform time across edges introduced challenges in terms of synchronisation and reactivity. These observations indicated an evident necessity for a preliminary processing step to discretise temporal resolution across the map, thus motivating the development of the Decomposer Submodule. Furthermore, the capacity to incorporate logical conditions on edges facilitated the implementation of the Action Handler in subsequent phases.

Chapter 4

Decomposer Submodule

The Graph Decomposer component has been developed to address a fundamental issue within the TEA* algorithm: the effective breakdown of the newly introduced time dimension. In the context of a graph map, it is essential to note that edges may vary in length and traversal speed [34].

A limitation of the TEA* algorithm is its assumption of uniformity across all edges, which fails to consider variations in attributes such as distance, velocity, and traversal cost. This limitation can substantially impact performance, frequently necessitating multiple recalculations of the robot's trajectory mid-task.

The Decomposer module has been created to discretise the time dimension by uniformly segmenting edges throughout the map to tackle this issue. However, it is essential to note that achieving perfect uniform division is not always possible due to inconsistencies in edge length and speed. This may lead to variations in timing across different edges, which could result in minor discrepancies in synchronisation and collision avoidance.

To achieve this objective, the Decomposer is divided into two main parts:

1. **Step Calculation**, the Decomposer calculates the step with the lowest possible error.
2. **Edge Splitting**, after knowing the step with the lowest error, it will split all the edges according to this optimal step.

4.1 The Importance of Discretising the Time Dimension

Before elucidating the methodology and algorithm of the code, it is imperative to articulate the rationale behind discretising the time dimension, which is hypothesised to be of significance and consequence. It is important to note that the Agent Supervisor module has been developed to address some of the challenges faced by TEA*, including unexpected obstacles, delayed communication, slippage and mechanical delays. The Graph Pre-Processor module addresses other challenges, such as parallel edges and choke points. The latter two challenges are addressed by their ability to either block the robot's path or the robot's surrounding area as a security feature. However, it should be noted that if the time dimension is not discretised, this can lead to inefficiency and delays.

4.1.1 Reducing the Impact of Security Features

Implementing security features, such as the parallel edge or the obstruction of a surrounding area, can result in deadlocks, inefficiency, and delays. As illustrated in Figure 4.1, the obstruction caused by the parallel edges impedes other robots' movement along the corridor's upper section and access to the dock in the middle. The decomposition of the map is a potential solution to this issue, as it would result in the upper path and the middle dock becoming accessible for travel.

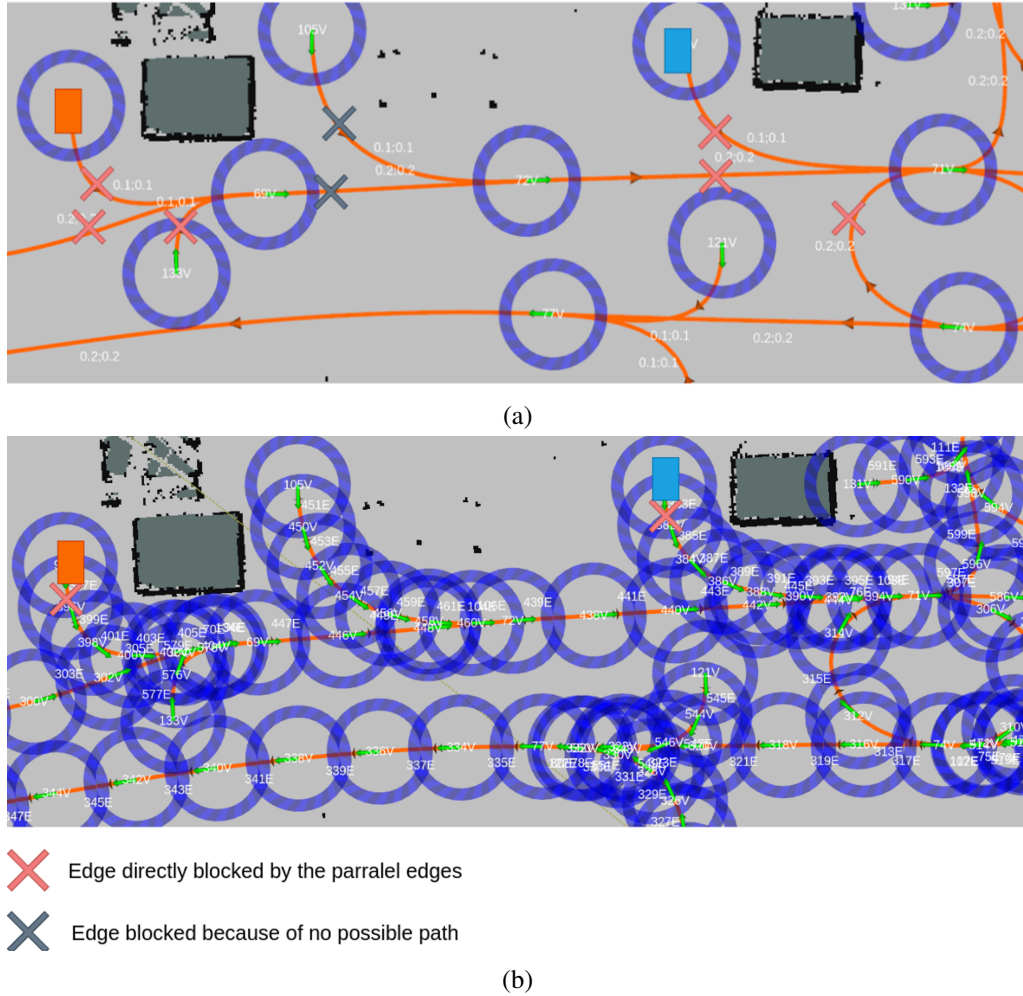


Figure 4.1: Impact of the Parallel Edge Feature: (a) Original map; (b) Decomposed map;

As depicted in Figure 4.2, the integration of equal temporal edges also allows the orange robot to halt in closer proximity to the blue robot, effectively minimising the frequency of replanning or in the event of an external factor, such as an individual passing by, the decomposed map affords the blue robot additional time to manage any potential collisions without the necessity of halting the orange robot.

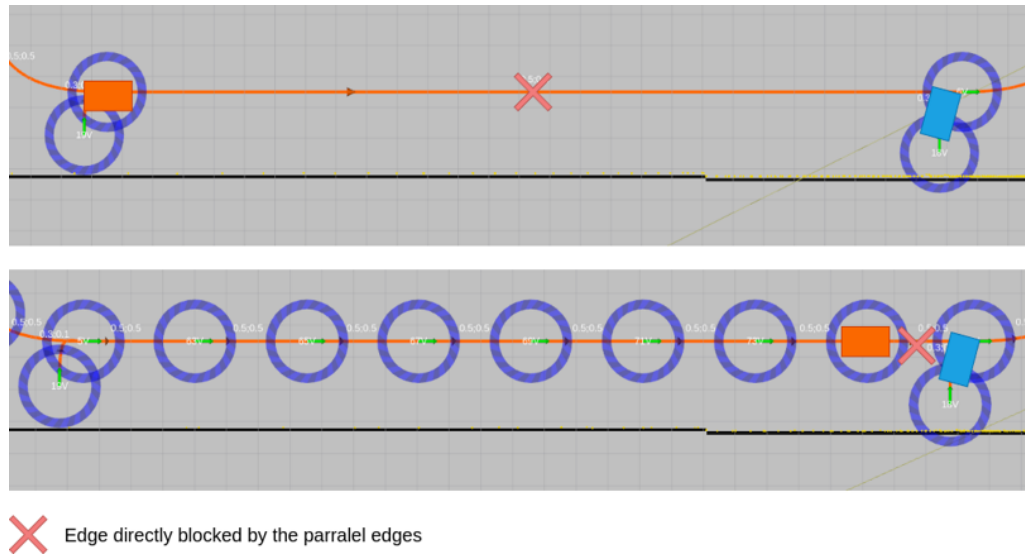


Figure 4.2: Impact of a Decomposed Map on Stopping Distance.

4.2 Step Calculation

To perform map decomposition effectively, it is essential to determine an optimal time step that standardises traversal durations across all graph edges. This uniformity reduces the occurrence of deadlocks and replanning attempts for the entire robotic fleet, a particularly salient phenomenon in dynamic environments.

The procedure delineated in Algorithm 1 is responsible for computing this optimal time step. It is a pivotal variable within the fleet coordination system, providing a foundation for temporal discretisation of the navigation graph.

The algorithm commences by analysing each edge in the original graph, encompassing the environment layout and traversal properties. The system calculates the minimum and maximum allowable step sizes using robot-specific footprint and speed data. This is achieved by dividing the physical length of each edge by the range of possible speeds, thereby generating a bounded interval of candidate step values.

After establishing the step size range, the algorithm undertakes an exhaustive search within this interval, incrementing by diminutive steps (e.g., 0.001 units). For each candidate step, the algorithm simulates the decomposition of every edge in the graph. The method involves calculating the number of discrete steps of that size that would be required to traverse each edge, and evaluating the error introduced by rounding these divisions to whole numbers. This error is indicative of the discrepancy between the intended and actual subdivision.

The fundamental objective is to minimise the aggregate error introduced by these approximations. The algorithm accumulates the individual errors across all edges and calculates the mean error for the current step size. If the average error is lower than previously tested values, the current step is stored as the most promising candidate. Following the exhaustive iteration of all potential step sizes, the algorithm selects the one that yields the lowest mean error as the final result.

Algorithm 1 Find Lowest Error Step by Average

Require: *graph_map* (A data structure that contains the edges and vertices of a graph, along with the properties of these elements)

Require: *robots* (Data structure containing the robot's information)

Ensure: *new_step* (An integer variable with the lowest possible error step)

```

1: function FINDAVGERROROFSTEP(graph_map, robots)
2:   step  $\leftarrow$  max_and_min_step(graph_map, robots.length)
3:   for currStep  $\leftarrow$  step.min to step.max do
4:     for CurrEdge  $\leftarrow$  graph_map.edges.begin()
5:   to graph_map.edges.end() do
6:     div  $\leftarrow$  new_edge_div(currStep, CurrEdge.cost)
7:     newEdgeErrorSize  $\leftarrow$  error_size(div, CurrEdge.cost)
8:     sumError  $\leftarrow$  sumError + newEdgeErrorSize
9:     nOfDiv  $\leftarrow$  nOfDiv + div
10:   end for
11:   currStepError  $\leftarrow$  total_step_error(sumError, nOfDiv)
12:   if currStepError < lowestStepError then
13:     lowestStepError  $\leftarrow$  currStepError
14:     new_step  $\leftarrow$  currStep
15:   end if
16: end for
17: return new_step
18: end function

```

This computed step size is then propagated to the Edge Splitting phase. The selection of the step that introduces the least error during graph discretisation is pivotal in ensuring the system's ability to produce high-fidelity approximations of real-world distances and durations. This is achieved while maintaining computational efficiency and synchronisation across robots.

4.3 Edge Splitting

As previously outlined in section 4.2, the subsequent decomposition phase entails dividing the original graph's edges into a series of smaller, uniformly timed segments. This phase is a crucial part of the map transformation process. It is implemented through Algorithm 2, which creates a new graph structure where all paths are segmented according to the chosen step granularity.

The algorithm's methodology involves the calculation of the optimal subsegmentation of each edge in the original map. This is achieved by determining the cost (represented by the ratio of edge length to travel velocity) of the edge, divided by the *new_step* value. As defined in the preceding phase, this value ensures the new segments align with the minimum error configuration, leading to more consistent and synchronised planning across multiple robots.

The algorithm employs Bézier curves in conjunction with Casteljau's algorithm for each subdivision to accurately interpolate new vertices along each edge. Bézier curves are selected due to their capacity to represent smooth and continuous trajectories, which is essential for realistic path following in robotic applications. Casteljau's numerically stable and geometrically intuitive algorithm provides a recursive means to divide these curves into precise sub-curves. This guarantees that the geometry of each edge remains faithful to the original map, even after subdivision.

Algorithm 2 New Decomposed Map

Require: *new_step* (An integer variable with the lowest possible error step)

Require: *graph_map* (A data structure that contains the edges and vertices of a graph, along with the properties of these elements)

Ensure: *decomp_graph_map* (A data structure that contains the new edges and vertices of a graph, along with the properties of these elements)

```

1: function CALC_DECOMP_MAP(graph_map, new_step)
2:   for CurrLink  $\leftarrow$  graph_map.edges.begin()
3:   to graph_map.edges.end() do
4:     DivisionSize  $\leftarrow$  currLink.cost / new_step
5:     for div  $\leftarrow$  1 to DivisionSize do
6:       if div  $\neq$  DivisionSize then
7:         NewNode  $\leftarrow$  get_new_node(div, CurrLink)
8:       else
9:         NewNode  $\leftarrow$  CurrLink.node_end
10:      end if
11:      decomp_grap_map.vertex.push_back(NewNode)
12:      if div  $\neq$  DivisionSize then
13:        get_new_link(NewNode, CurrLink, FirstHalf,
14:        SecondHalf)
15:        NewLink  $\leftarrow$  FirstHalf
16:      else
17:        NewLink  $\leftarrow$  SecondHalf
18:      end if
19:      decomp_grap_map.edges.push_back(NewLink)
20:    end for
21:  end for
22: end function

```

Each segment is processed sequentially where a new interpolated vertex is created using Casteljau’s method. These vertices are stored in the new graph structure and connected by new edges. In the final division, the terminal vertex of the original edge is reused to ensure continuity.

The algorithm differentiates between the two resulting Bézier segments following each iteration of the split process. The initial segment is utilised to establish the new edge that connects to the graph, while the subsequent segment is retained for subsequent recursive subdivision if additional segments are required. This distinction preserves the smoothness and curvature of the path even after multiple iterations of splitting.

The output of this process is a new decomposed map (*decomp_graph_map*) that retains the original topology, but now includes additional vertices and edges that conform to the temporal step constraint. This sophisticated map is indispensable to the functioning of the fleet coordination system, as it curtails the number of replanning attempts, optimises the path and utilisation of resources across a multitude of autonomous agents.

4.4 Decomposer Submodule Tests

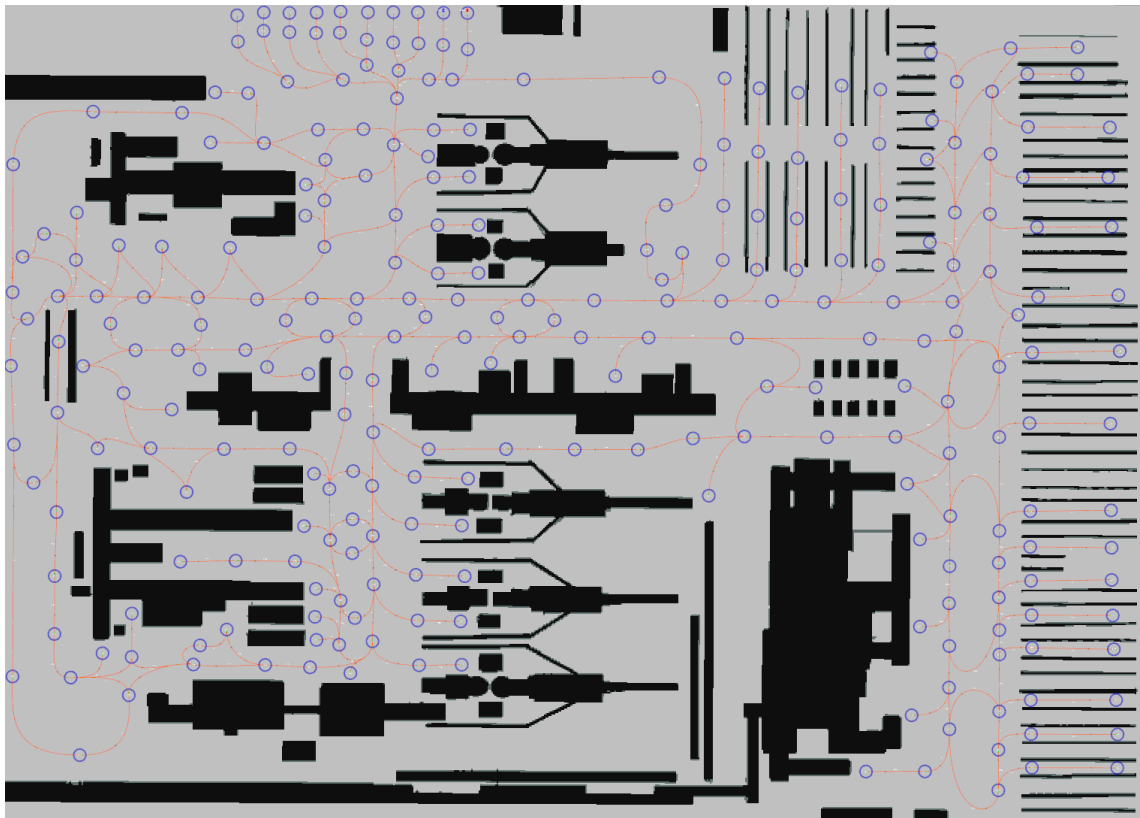
A targeted set of key performance indicators (KPIs) was defined and evaluated through simulation to evaluate the performance and practical benefits of the Decomposer submodule. The analysis concentrated on two core aspects: firstly, the effect of increased graph complexity, as measured

by a higher number of vertices and edges, on the computational time required to generate a path; and secondly, the success rate of task execution in a confined environment with multiple robots operating simultaneously.

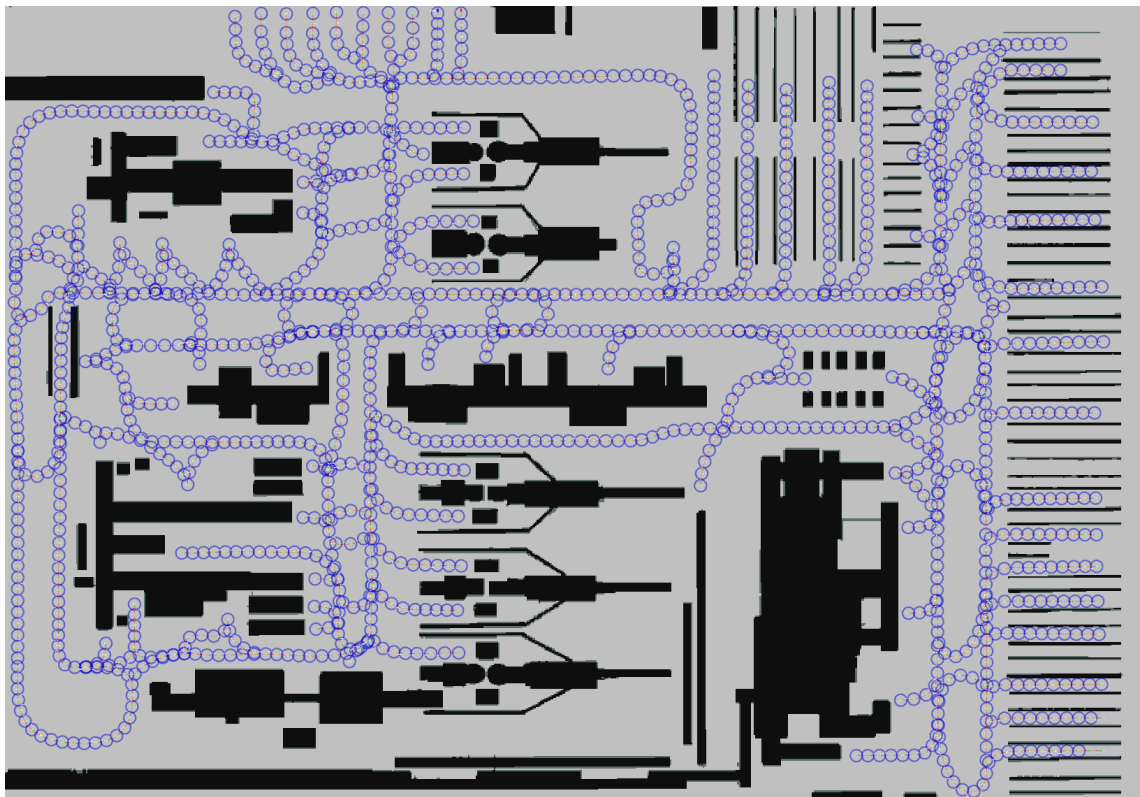
4.4.1 Time to Generate a Path

It is anticipated that the implementation of graph decomposition will result in an augmentation of structural complexity. This phenomenon can potentially directly influence the efficacy of the path planning algorithm. Specifically, the subdivision of original edges into smaller, uniform segments increases the number of vertices and edges in the graph, potentially raising the computational load. This can substantially influence the TEA* algorithm, given that its planning time may be increased due to the augmented search space. A considerable increase in planning latency could render using the Decomposer submodule impractical in real-time applications. It is therefore evident that one of the primary objectives of this evaluation was to ascertain the operational viability of the submodule under realistic conditions.

To achieve the aforementioned objective, a factory floor map of considerable size was selected for utilisation as the simulation environment, as illustrated in Figure 4.3, the transformation of the map following decomposition results in a significant increase in both vertex and edge density. In the original version, edge lengths were defined arbitrarily, with limited consideration given to the traversal time associated with each segment. Conversely, the decomposed map evidences a regular, homogeneous distribution of edge lengths, with each segment corresponding to a uniform time step. The temporal standardisation process introduces symmetry into the graph, ensuring that the time required to traverse each edge is consistent. This, in turn, enhances the accuracy and reliability of the TEA* algorithm.



(a)



(b)

Figure 4.3: Factory Floor: (a) Original; (b) Decomposed;

The test scenario entailed allocating challenging navigation tasks to a single robot, necessitating its traversal of the map from one extremity to the opposite corner. This configuration enabled a regulated evaluation of the Decomposer's impact on path planning duration, unencumbered by the presence of other agents. The experiment also provided an ideal case for stress-testing the planner, intending to ascertain its performance under high traversal demands. For that, multiple paths with different edge lengths were tested.

The following Table 4.1 presents the total planning time, which is defined as the time it takes the path planner to compute a path. This data was obtained through 10 repetitions of both the original and decomposed maps, and the average planning times for each were also presented, as well as the number of edges that the planned path had.

Table 4.1: Comparison of The Plan Time and The Number of Edges in The Large Factory Floor Map

Path Number	Original Map		Decomposed Map	
	Number of Edges	Avg. Plan Time (s)	Number of Edges	Avg. Plan Time (s)
1	26	0.422	138	2.283
2	19	0.420	108	2.254
3	15	0.406	88	2.246
4	18	0.414	87	2.234
5	18	0.417	74	2.224

These results are consistent with the prevailing expectations. As demonstrated by the collected data, the average planning time on the decomposed map is higher than that of the non-decomposed version. The data of the decomposed map indicates that the edge number does not significantly influence the planning time. For instance, a path with 138 edges takes approximately the same time as a path with 74 edges. This phenomenon occurs because the system initiates startup and generates a cost matrix, thereby facilitating the path planner's identification of an efficient route. The issue appears to be the search of the matrix, which is of complexity $O(1)$ [25]. A potential solution to this issue involves the division of the map into multiple smaller maps, thereby reducing the impact of the Decomposer module.

The following Table 4.2 shows that in a smaller map, the observed increase in planning time is considerably less pronounced and does not pose a significant limitation to system responsiveness. In addition, the advantages of a high task success rate and enhanced system scalability are deemed more important than the effect on planning time.

Table 4.2: Comparison of The Plan Time and The Number of Edges in The Laboratory Map

Path Number	Original Map		Decomposed Map	
	Number of Edges	Avg. Plan Time (s)	Number of Edges	Avg. Plan Time (s)
1	13	0.075	44	0.292
2	10	0.072	37	0.284
3	18	0.085	33	0.276
4	12	0.074	38	0.285
5	14	0.076	46	0.293

4.4.2 Impact on Task Execution

A laboratory-scale map was employed to evaluate the impact of the Decomposer on task completion success. In this scenario, the complexity was increased by introducing a fleet of eight robots, each assigned a fixed navigation task. The primary objective of this test was to determine whether integrating the decomposed map could reduce the number of emergency stops triggered by the path supervisor in response to navigation conflicts or coordination failures.

Figure 4.4 shows that the laboratory map was intentionally constructed without specific optimisations to avoid blockage scenarios. In particular, docked agents could easily obstruct access to other docking points. As outlined in subsection 4.1.1, the Parallel Edges detection mechanism marks all edges occupied by a robot's footprint as blocked, thereby preventing other agents from entering potentially unsafe regions. Additionally, due to only a single edge leading to the docking area, any agent occupying this edge would effectively block all adjacent edges connected to the same vertices.

The test procedure was straightforward: eight robots were deployed concurrently, with each robot assigned the task of moving from one docking station to another unoccupied one; this can be seen in Table 4.3.

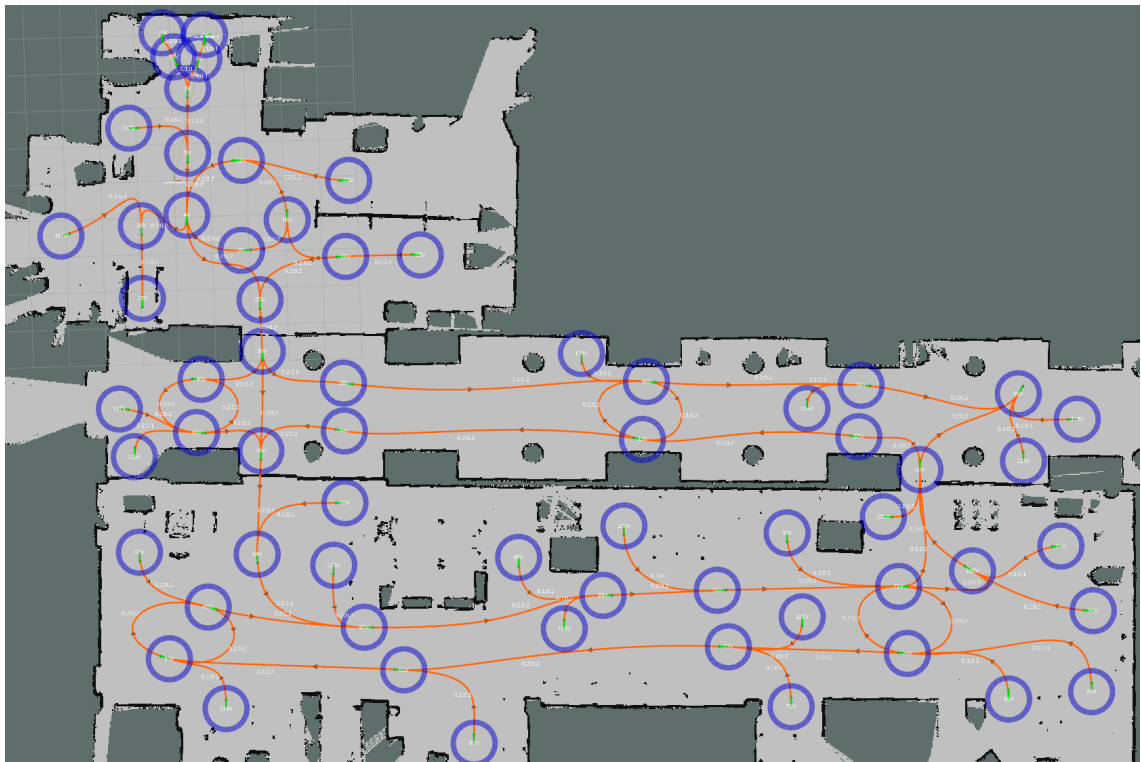
Table 4.3: Comparing Task Execution

Number of robots	Original Map	Decomposed Map
1	No problems to execute the tasks	No problems to execute the tasks
2	No problems to execute the tasks	No problems to execute the tasks
3	No problems to execute the tasks	No problems to execute the tasks
4	No problems to execute the tasks	No problems to execute the tasks
5	Task assignment problem	No problems to execute the tasks
6	Task assignment problem	No problems to execute the tasks
7	Task assignment problem and emergency stop	Task assignment problem
8	Task assignment problem and emergency stop	Task assignment problem

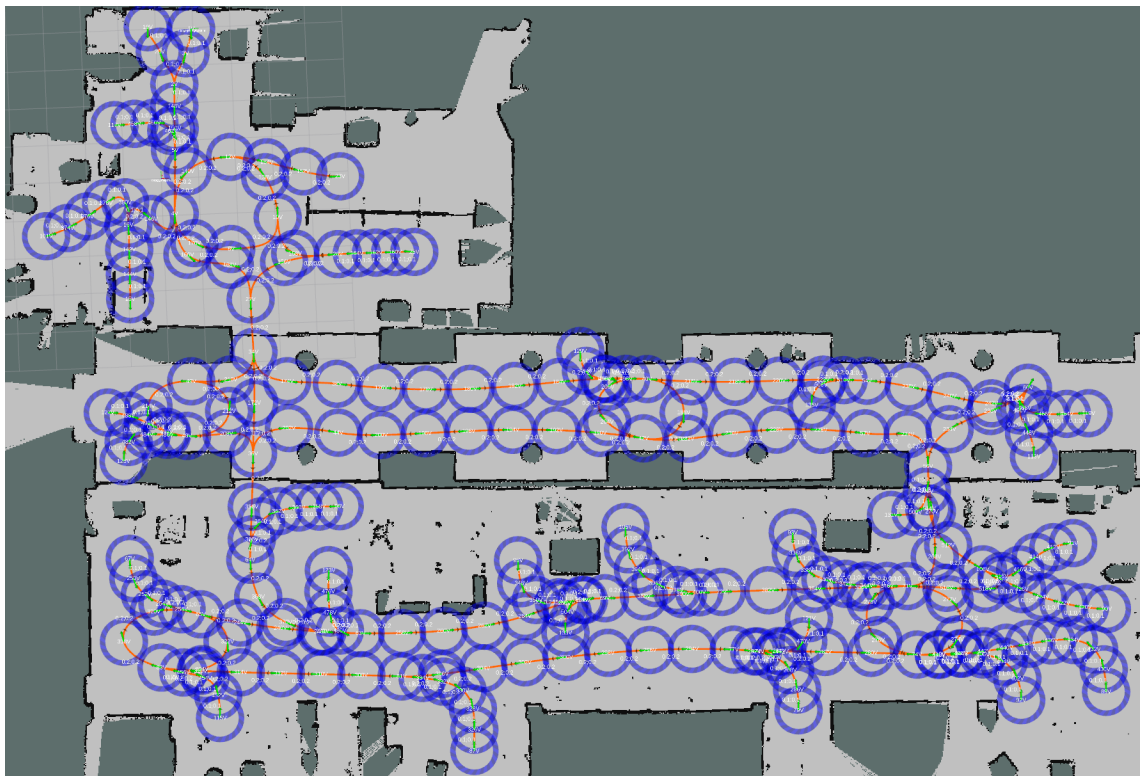
The utilisation of the decomposed map resulted in the successful completion of all test iterations, with no instances of emergency stop or navigation failure. Conversely, the test consistently failed when utilising the original, non-decomposed map. The system was compelled to initiate an emergency halt in each instance due to deadlocks and unresolved conflicts.

Furthermore, assigning tasks to individuals was identified as a significant challenge. In circumstances where specific tasks were assigned a higher priority level, this frequently resulted in deadlock and the need for emergency stops, particularly in scenarios characterised by high robot density. Despite endeavours to optimise task allocation during the trials, the escalating number of robots instigated unpredictability. In the absence of the implementation of more advanced algorithms, the determination of the optimal task prioritisation becomes progressively more arduous as the system scales.

Further tests were conducted using a greater number of robots to assess the scalability of the system. However, due to the spatial limitations inherent in the map, accommodating more than eight robots can prove challenging, yielding inconsistent outcomes.



(a)



(b)

Figure 4.4: Laboratory Map: (a) Original; (b) Decomposed;

4.4.3 Summary

This chapter introduced the Decomposer Submodule, which was developed to address inconsistencies in TEA*'s handling of time across edges of varying lengths and speeds. The submodule operates in two phases: firstly, it calculates the optimal time step with minimal error, and secondly, it applies that step to subdivide all edges using Casteljau's algorithm for Bézier curves. This transformation has improved temporal uniformity across the map, reducing desynchronisation, minimising deadlocks, and enhancing overall system reliability. The simulations demonstrated that, whilst the number of edges and planning time increased, the benefits of execution stability and task completion far outweighed the cost. The Decomposer is thus a foundational enhancement that ensures consistent temporal planning and prepares the coordination system for large-scale, real-time operation.

Chapter 5

Action Handler

The Action Handler is designed to reduce an optimisation problem faced by the TEA* algorithm: the impact of external factors such as doors, elevators and charging time on system performance.

Since different actions may require varying amounts of time to execute, the chosen approach is to proactively open the door or call the elevator in advance. This helps minimise the chances of having to stop, thereby decreasing the direct impact on overall performance.

The action handler was developed as a new software layer independent of the fleet coordination system, which is desirable because each door company may have a different API (Application Programming Interface). Hence, a standalone module is better for easy adaptation and change, and for better visualisation, a new ROS node called Door Simulation was created.

5.1 System Architecture

Creating the Action Handler necessitated the extraction of data from the pre-existing layers of the fleet coordination system. This included information such as the number of robots, their respective positions within the graph map, and their scheduled trajectories. In Section 5.1.1, a more comprehensive discussion will be presented on the methods employed to obtain this data and the modifications essential for implementing them within the fleet coordination system.

The philosophy adopted during the development of the Action Handler was to strive for simplicity and ease of implementation, to facilitate integration in the event of the emergence of new actions. To better visualise the impact of the new Action Handler layer and the Door Simulation node, Figure 5.1 presents an updated version of the system architecture.

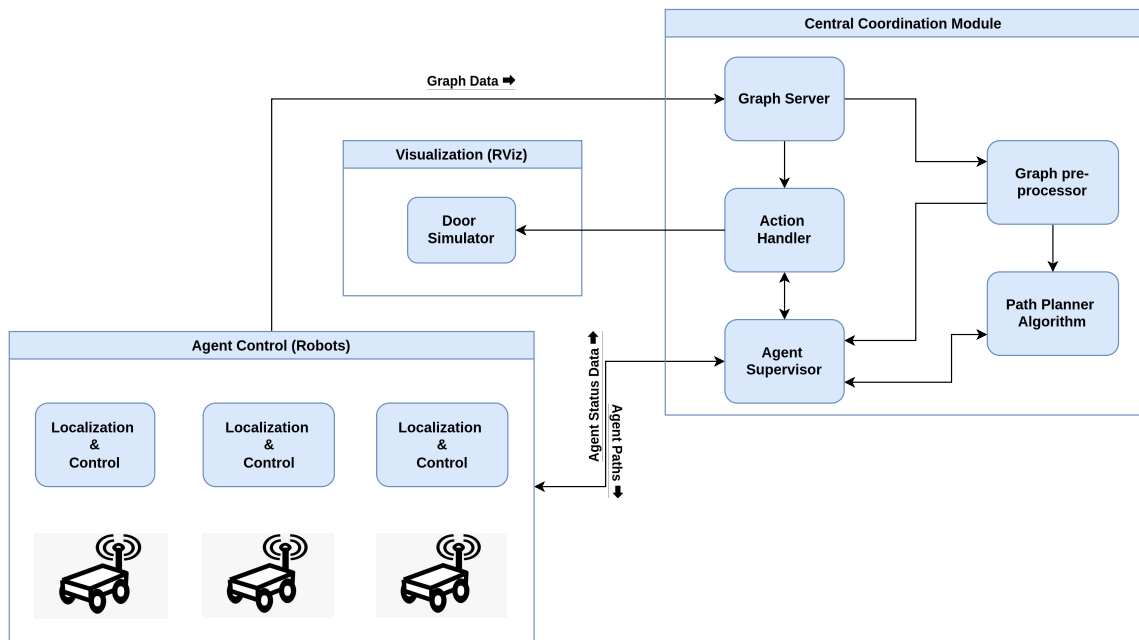


Figure 5.1: Updated System Architecture

The state machine employed in the system is a universal component that can be utilised irrespective of the action to be executed. The Action Handler can process any action expressed in terms of preconditions and postconditions.

The structured top-level state machine that governs the Action Handler (see Figure 5.2) actively attempts to obtain information about the robots in the system and their position. The corresponding state machine is activated if a robot is on an edge with a condition.

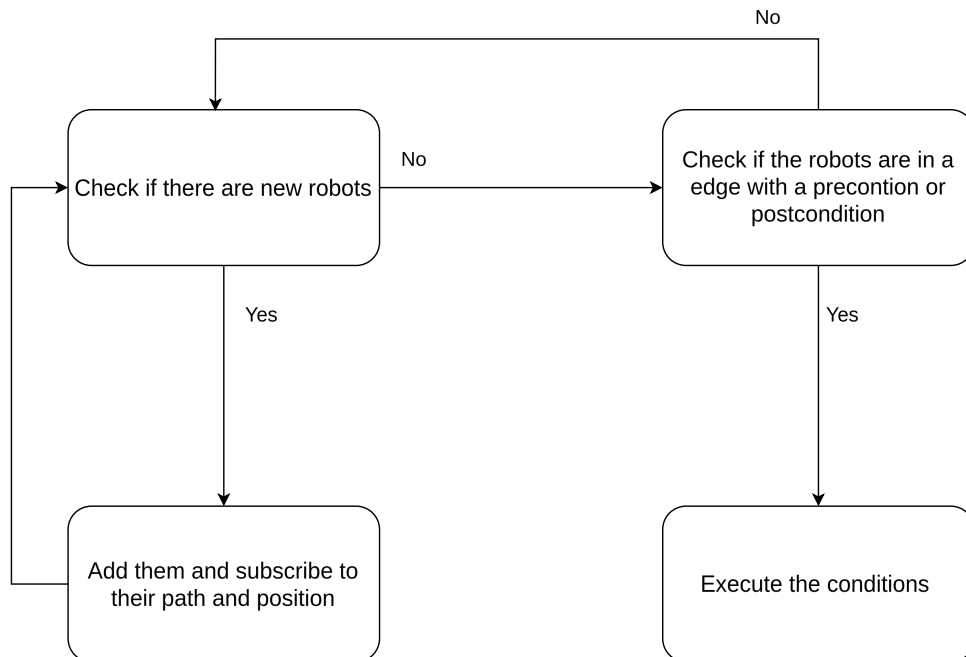


Figure 5.2: Top-Level State Machine of the Action Handler

5.1.1 Connection With The Fleet Coordination System

The Action Handler plays an essential role in synchronising high-level decision-making processes with the navigation status and behaviour of AGV. This synchronisation is accomplished through ROS communication mechanisms, particularly topics and services, that allow the Action Handler to monitor, interpret, and influence robot activity within the fleet.

Upon initialisation, the Action Handler subscribes to the `execution_route_states` topic published by the Graph Server module. This topic provides a structured view of the symbolic position of each AGV, including its current edge identifier, progress along the path, and a namespace that uniquely identifies the requesting robot. Through attentive listening, the Action Handler can detect the presence of AGVs in the system, track their movements in real time, and update their internal states accordingly.

Upon reception of a message from an unfamiliar namespace, the system identifies it as a new robot joining the coordination framework. The Action Handler has been developed to dynamically create a new internal entry for this AGV, with the subsequent initialisation of its relevant state information and subscription to a dedicated topic corresponding to its namespace (e.g., `/robot_10/navigation_handler2/execution_route_states`). Route information assumes great significance in decision-making processes, such as conflict resolution and the allocation of resources in shared spaces.

The topic entitled `edge_to_block` is utilised to obstruct and eliminate the availability of an edge for utilisation by the Path Planning module during the process of planning a path.

The module also integrates with coordination-level services provided by the Agent Supervisor module. This integration enables a set of control actions, which in turn allow the Action Handler to exercise influence over the behaviour of AGVs. The following services are offered:

- **`/server/path_supervisor/stop_robot`** – to remotely halt an AGV;
- **`/server/path_supervisor/force_replan`** – initiate a path replanning operation;
- **`/server/path_supervisor/force_replan_action`** – to issue a path replanning command as a ROS action, enabling asynchronous control;
- **`/server/path_supervisor/force_stop`** – to enforce an emergency stop.

These services enable the Action Handler to issue targeted interventions based on real-time operating conditions, providing centralised oversight and fine-grained control over individual robot behaviours. The replan operations are operational, but in their current state, the action handler is not utilising them because the Agent Supervisor is already ensuring the replanning in the event of a delay.

5.2 Implementing Precondition and Postcondition

The implementation of preconditions and postconditions necessitates the utilisation of a class designated as `Condition`. Each instance of the specified class will possess an edge, which will be

associated with information about the preconditions and postconditions it may have. These will include the type of resource being managed (for example, a door, an elevator, or a charging station), the service that can stop multiple robots, a vector to store the robots that were stopped, and a state machine to manage the trigger that instigated these conditions.

The Action Handler module was conceptualised to facilitate modularity. In the event of the necessity for a new action, it is only required to create a state machine for said action and a new Ain't Markup Language (YAML) file comprising the information necessary to complete the Condition class.

The subject of Section 5.3 is managing a specific resource, namely the automated doors. Due to time constraints, the only action that was fully developed and tested was the management of computerised doors.

5.3 Controlling and Managing Doors

The control and management of automated doors is contingent upon the information that the door is capable of communicating. The simulation does not represent computerised doors in the original version of the fleet coordination system. Therefore, it was necessary to create information about the simulation of doors, as outlined in Section 5.3.2.

As previously stated, the state machine depicted in Figure 5.2 is universally applicable to all conditions, and consequently, it is employed to manage automated doors. The sole variable is how the state machine utilises the information provided as a precondition and a postcondition. Details of the state machine are elucidated in Section 5.3.1.

The information about the automated doors is given in a YAML file containing the following elements:

- **Edge_id:** Indicates which edge is a door.
- **Precondition:** A vector with all the edges that will trigger a precondition, in this case, open the door.
- **Postcondition:** A vector with all the edges that will trigger a postcondition, in this case, close the door.
- **Type:** Identifies which type of action is, in this case, 0 to indicate a door.

5.3.1 State Machine

The state machine created can be divided into two major components: the precondition state machine and the postcondition state machine.

In the precondition state machine (see Figure 5.3), if a robot enters a precondition edge, it will initiate an action to open the door. Upon approaching the edge that represents a door, the robot will ascertain whether the door has already returned to its open state. If so, the robot will continue

its trajectory as usual. Otherwise, it will halt, restrict the door edge, initiate another call to open the door, and await the door's return to the open state. After this, the robot will unlock the door edge, thus enabling the supervisor to replan for that robot.

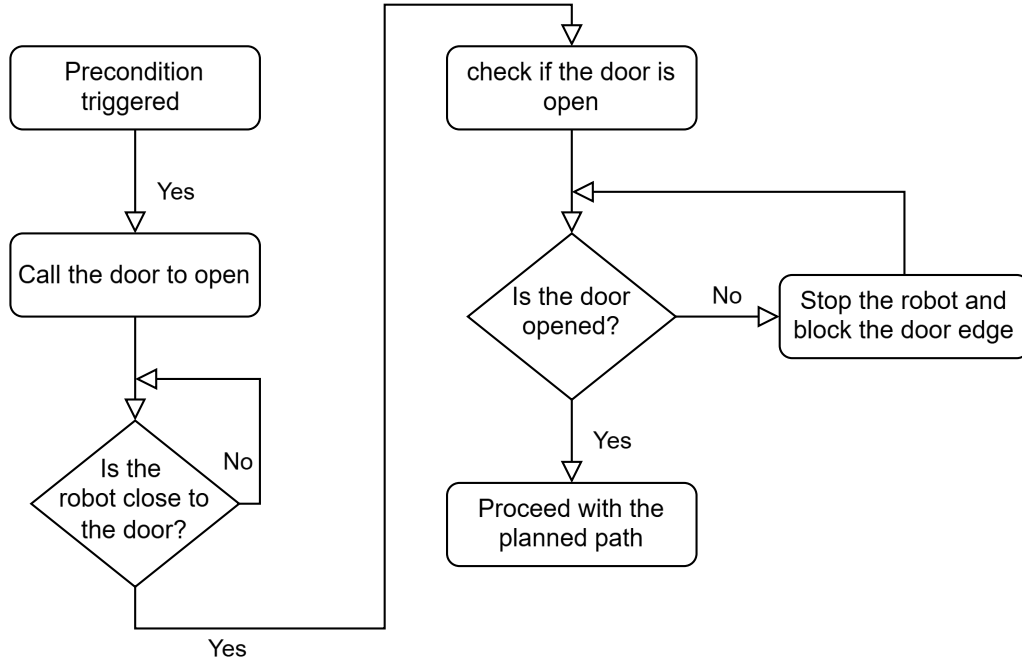


Figure 5.3: Precondition Diagram

The initiation of the door closure is only triggered by the postcondition state machine when the robot is in a postcondition edge, thereby signifying that it has exited the door.

5.3.2 Door Simulation

A new ROS node was developed to better visualise and test the Action Handler, simulating the automated doors' behaviour visually and interactively. The main goal of this node is to facilitate the validation of the Action Handler, enabling realistic testing without requiring a physical environment.

In addition to setting up the action servers, initial (x, y) positions are assigned to each door, and a dedicated thread is launched to publish visualisation markers continuously. These markers graphically represent the doors in a 2D environment using RViz (Figure 5.4), and depending on whether the command is to open or close the door, the x position of the door is incremented or decremented, simulating longitudinal movement.

Throughout this process, both the door's transformation and its visual marker are continuously updated. Feedback messages are also published at each stage, reflecting whether the door is opening, open, closing, or closed.

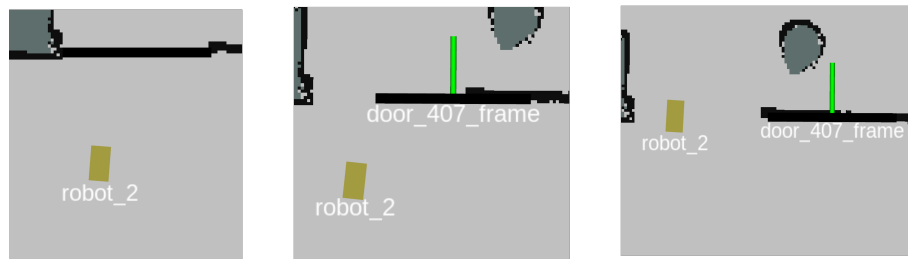


Figure 5.4: Example of a Door Opening on RViz

The simulation of door movement is governed by a state machine (Figure 5.5) that ensures realistic transitions and consistent behaviour when executing commands. When a command to open a door is received, the simulator adheres to a structured sequence of checks and actions that emulate the operational logic of an automated system.

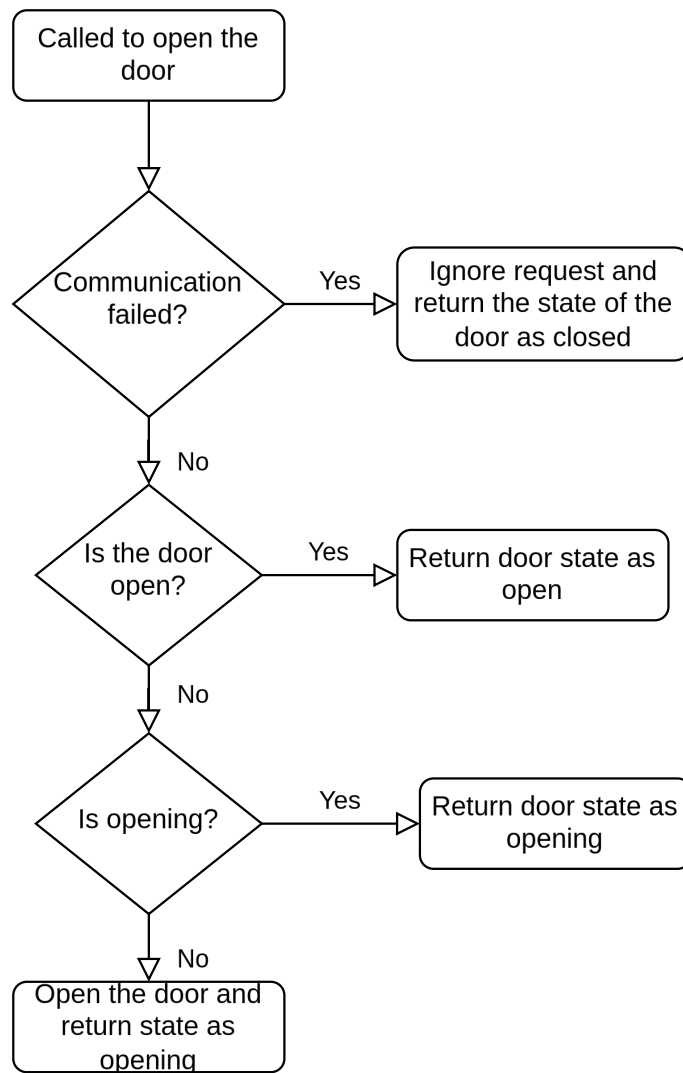


Figure 5.5: Door Simulation to Open a Door

First, it verifies whether the system is capable of processing the request. Whenever communication with the simulation node fails due to timeouts, invalid IDs or resource contention, the request is ignored, and the door is considered to remain closed. This design ensures resilience to malformed or untimely commands, avoids conflicting operations on the same entity, and introduces a chance of communication failure to test the system's resistance.

If communication is successful, the simulator checks the current status of the requested door. No further action is taken beyond reporting its state if the door is already open or closed. This prevents unnecessary or redundant operations, maintains efficiency, and reduces simulation overhead. If the door is in the process of opening or closing, the system returns the current transitional state. This prevents interrupting or overlapping commands during active motion, thus aligning the simulation with the behaviour of real-world hardware, where commands are not executed instantaneously and physical states change over time.

The movement logic is executed asynchronously, enabling multiple doors to operate independently. State feedback is provided continuously throughout the motion, reflecting transitions from open to closed and vice versa when the inverse operation is requested.

This abstraction of door behaviour is managed through a finite-state model and implemented with non-blocking operations to ensure a high-fidelity simulation. It mimics how real-world actuators respond to commands and provide feedback, making it ideal for integration testing with autonomous agents or higher-level task planners.

5.4 Action Handler Tests

A set of tests was designed and executed to evaluate the efficacy and reliability of the Action Handler module. The objective was to assess the module's impact on the performance and scalability of the fleet coordination system within a simulated environment featuring automated door mechanisms.

Three primary tests were made:

- **Basic Functionality Test:** Served as a proof of concept to validate the fundamental operation of the Action Handler in a controlled environment. This test ensured that door coordination using preconditions and postconditions was functioning as intended, as well as the doors' visualisation before proceeding to more complex performance and scalability evaluations.
- **Performance Benchmark:** Compared task execution times with and without using preconditions and postconditions for door interaction to evaluate the efficiency introduced by the Action Handler.
- **Scalability Stress Test:** Assessed the system's behaviour under increased operational load, intending to determine how well the coordination framework performs when scaled to multiple robots.

The impetus for this testing strategy is rooted in the imperative to ensure that the integration of the Action Handler into the coordination workflow does not impede task efficiency or compromise system responsiveness. Furthermore, validating its benefits in the management of preconditions and postconditions is imperative for confirming its utility in future real-world deployments.

5.4.1 Basic Functionality Test

Integrating a new module into the existing fleet coordination system is a more complex process than adding a new submodule to an existing one. Consequently, utilising a rudimentary environment to evaluate the connection with the system in its entirety, in addition to the implementation of novel functionalities, is paramount to facilitate the identification of potential issues and thereby reduce the time expended on debugging the new module.

As illustrated in Figure 5.6, the novel environment comprises a basic oval graph with two docking points for the system's two robots and two doors. The purpose of the doors is twofold: firstly, to facilitate the testing of the precondition and postcondition state machines; and secondly, to ensure that the system is capable of working in a parallel fashion. That is to say, the system must be capable of receiving multiple requests to different doors simultaneously without encountering any issues. Furthermore, the system should prove whether the visualisation made in RViz for the doors is functioning correctly.

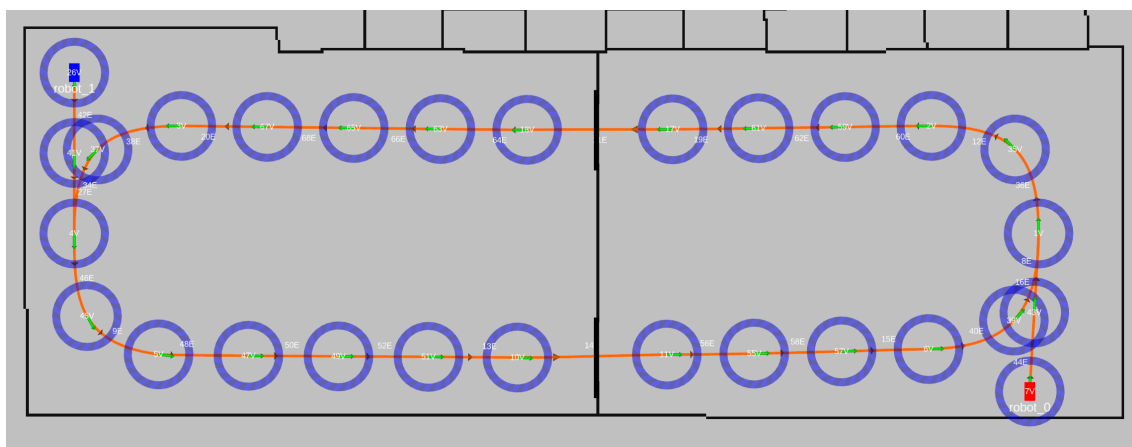


Figure 5.6: Simple Map Created to Evaluate the Action Handler

The experiment's configuration is straightforward: both robots must traverse the door and stop after it is closed. As demonstrated in Figure 5.7, the sequence of actions undertaken by the robots during the execution of the designated tasks is illustrated. The red robot is not using precondition and postcondition, whereas the blue robot employs them.

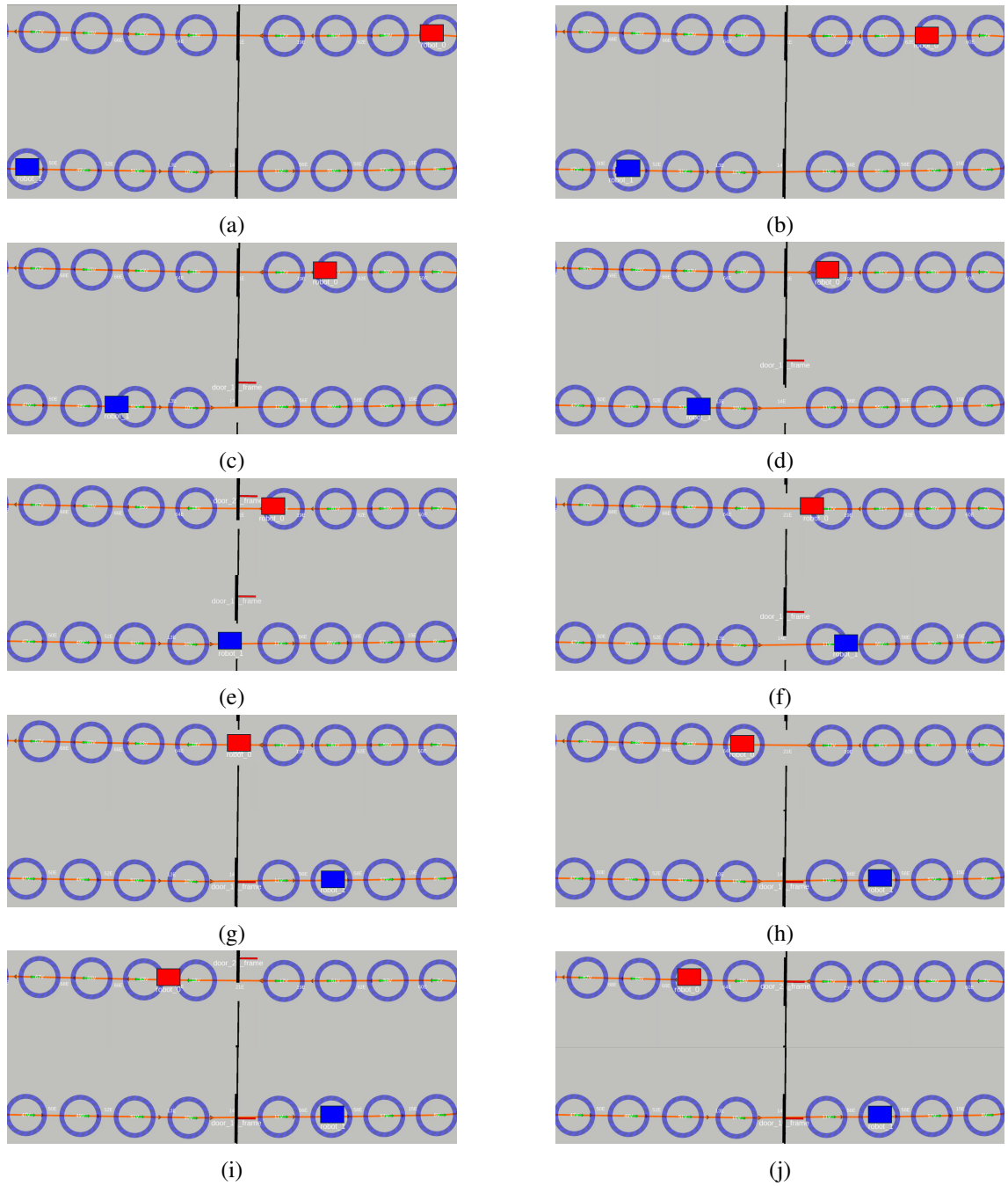


Figure 5.7: Simple Scenario with and without Precondition and Postcondition: step-by-step progression (a–j).

These figures demonstrate the benefits of calling the doors pre-emptively using the precondition and postcondition. It shows that the red robot must stop when it reaches the door, creating an unnecessary delay. In contrast, the blue robot shows that the doors are called pre-emptively, meaning the robot doesn't need to stop when approaching the door. This reduces delay and increases the system's efficiency.

5.4.2 Performance Benchmark

To quantify the performance gains introduced by the Action Handler, a comparative test was conducted using the previously introduced decomposed laboratory map (refer to Figure 4.4). In this scenario, the objective was to program five autonomous mobile robots to perform a delivery operation. Each robot was required to navigate from a distinct initial position to a shared target location designated for cargo drop-off and return to its starting point.

Two test configurations were defined:

- **With Action Handler:** Door operations were managed using preconditions and postconditions defined within the Action Handler module. This ensured that each robot checked and interacted with doors in a context-aware and sequentially safe manner.
- **Without Action Handler:** Door interactions were not regulated by preconditions and postconditions. Robots attempted to navigate the environment without pre-emptively calling to open the doors, relying on default behaviours of trying to open upon encountering closed doors.

The spatial configuration of the experiment is depicted in Figure 5.8a, which shows the robots' starting and ending positions, and in Figure 5.8b, which illustrates the cargo delivery location.



(a)



(b)

Figure 5.8: Robots' Positions in Performance Benchmark: (a) Initial Position; (b) Target Position;

Each configuration was executed ten times, identified as interactions, to ensure the statistical validity of the study and account for any potential anomalies. This repetition facilitated identifying and removing outliers and provided a consistent basis for task completion time. The results derived from this benchmark are intended to demonstrate the operational efficiency gains attributed to using the Action Handler module in managing complex task dependencies involving interactive infrastructure.

The left side of the Table 5.1 provides a quantitative representation of the temporal requirements for the completion of tasks in the absence of a pre-emptive door-calling action by the robot. The sequence of actions entails the robot's arrival at the door, its halting of movement, the initiation of a door-opening call, and the subsequent request for a path to the system. Following the reception of a new path, the robot commences traversal of the door.

In contrast, the temporal representation provided on the right side of the Table 5.1 is through the utilisation of the Action Handler, whereby the robot initiates the door-opening sequence in advance. In such cases, upon arrival at a door, the robot will typically find it open, thus obviating the need to halt and recalculate its trajectory. In the event of delays in communication, a robot can arrive at the edge of a door without the door being fully open. This will result in the robot stopping and requesting a new path. However, this is a rare occurrence, and the door will usually be open when the robot stops.

Table 5.1: Execution Times With and Without the Action Handler (in seconds)

	Without Action Handler		With Action Handler	
	Average	Standard Deviation	Average	Standard Deviation
robot 0	242,5	2,6	241,3	1,7
robot 1	280,4	4,5	266,4	2,5
robot 2	410,6	8,6	384,1	1,8
robot 3	347,3	8,9	273,6	1,7
robot 4	125,8	8,0	111,1	3,2

The Action Handler and the utilisation of precondition and postcondition contribute to reducing overall task execution time. It has been demonstrated that the Action Handler results in improved mean completion times for all robots.

As demonstrated in Figure 5.9, a more effective visualisation of the enhanced performance is provided. The Action Handler exhibits superior execution time compared to its absence, with an average reduction of 30 seconds observed in the execution time. This substantial reduction enhances the system's efficiency and the path planner's accuracy.

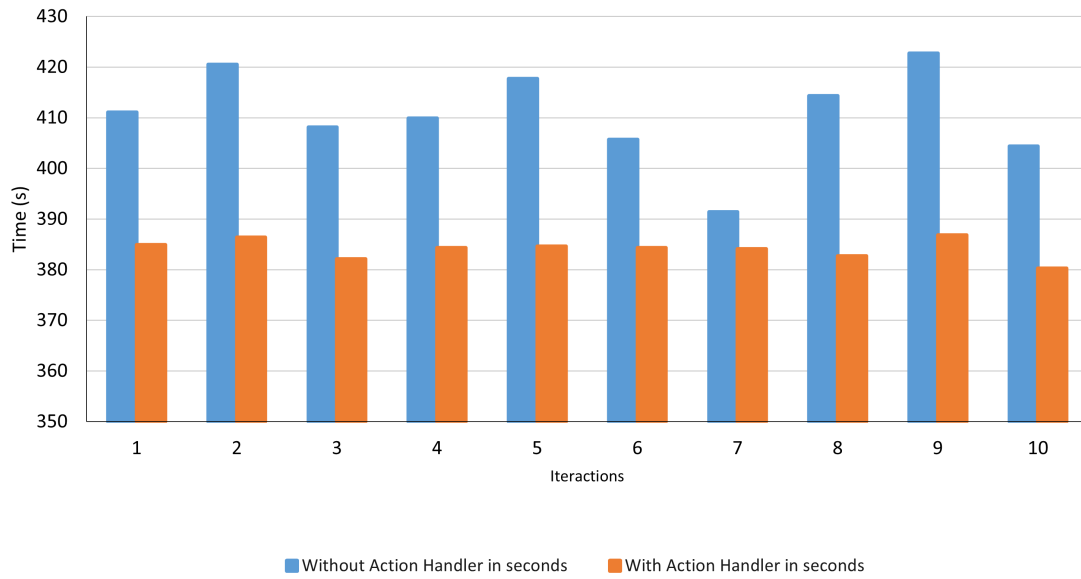


Figure 5.9: Comparing Overall Execution Time

5.4.3 Scalability Stress Test

A scalability stress test is a type of test that is used to determine the limits of a system. The aim is to identify potential bottlenecks or issues that may not be visible during regular operation. In this experiment, the number of robots increased incrementally. The task assigned to the robots was to proceed to an edge and return to their initial position. As demonstrated in Table 5.2, the number of robots capable of operating within the simulation is influenced by several factors.

Table 5.2: Number of robots in the system and their impact

Number of robots	Observations
1	No problems to execute the tasks
2	No problems to execute the tasks
3	No problems to execute the tasks
4	No problems to execute the tasks
5	No problems to execute the tasks
6	Task assignment problem
7	Task assignment problem
8	Task assignment problem

In scenarios involving six robots or more, allocating and prioritising tasks became a matter of concern. In the present fleet coordination system iteration, tasks are prioritised by the first-in, first-out (FIFO) method. This signifies that the initial task entered into the system will have a higher priority level than subsequent tasks. Consequently, the efficacy of task assignment is contingent upon the user initiating the task. However, when the number of robots is augmented, thereby increasing interactions between them, evaluating which task sequence is more efficient without prior testing becomes challenging.

Figure 5.10 illustrates the initial and target positions of the robots used during the test.

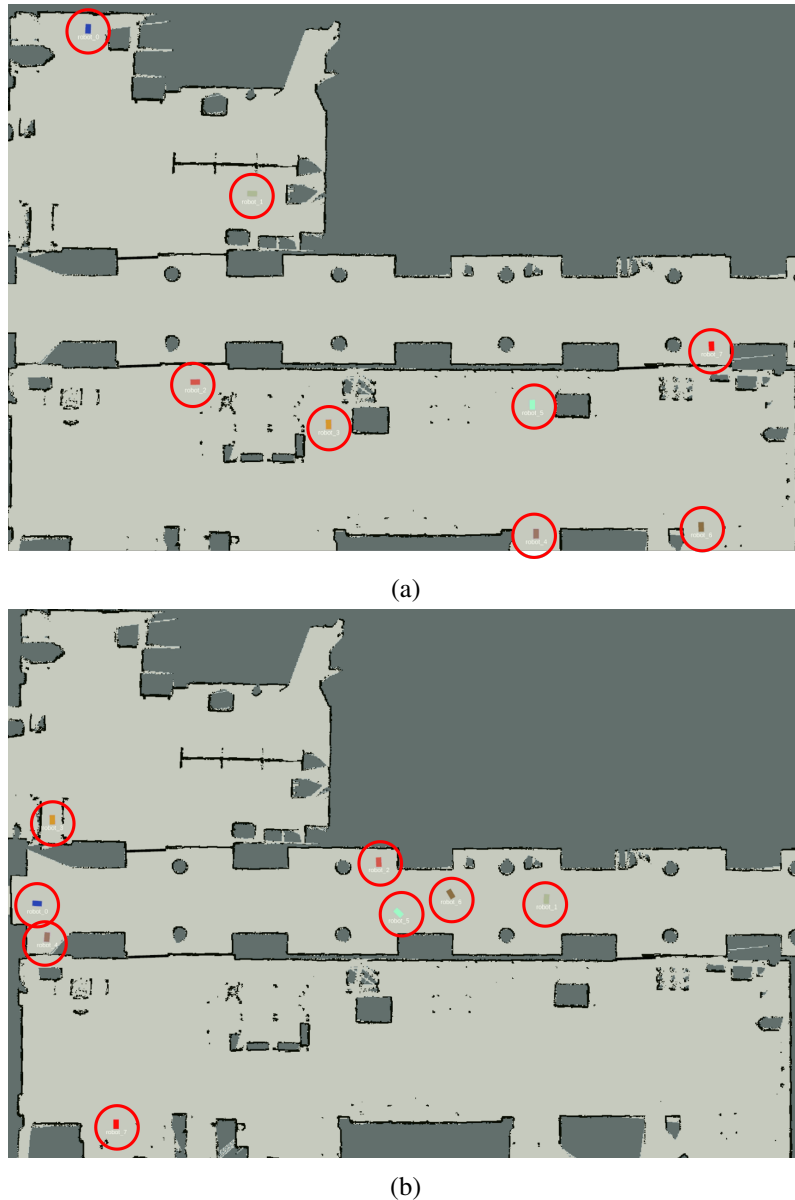


Figure 5.10: Robots' Positions in Stress Test: (a) Initial Position; (b) Target Position;

As illustrated in Subfigure 5.10b, the issue of spatial constraints begins to manifest. In the middle corridor, the positioning of robot targets is so close that it becomes impractical for other

robots to utilise that corridor. These areas are defined as choke points, which are characterised by high traffic and significantly impact the system’s performance. To better visualise what a choke point is and why task priority will impact the performance of the system if the number of robots is increased, Figure 5.11 shows that if a significant number of robots are present in a room, the door effectively functions as a choke point. In the absence of adequate features to address choke points or enhanced task execution, the likelihood of a deadlock is considerably high.

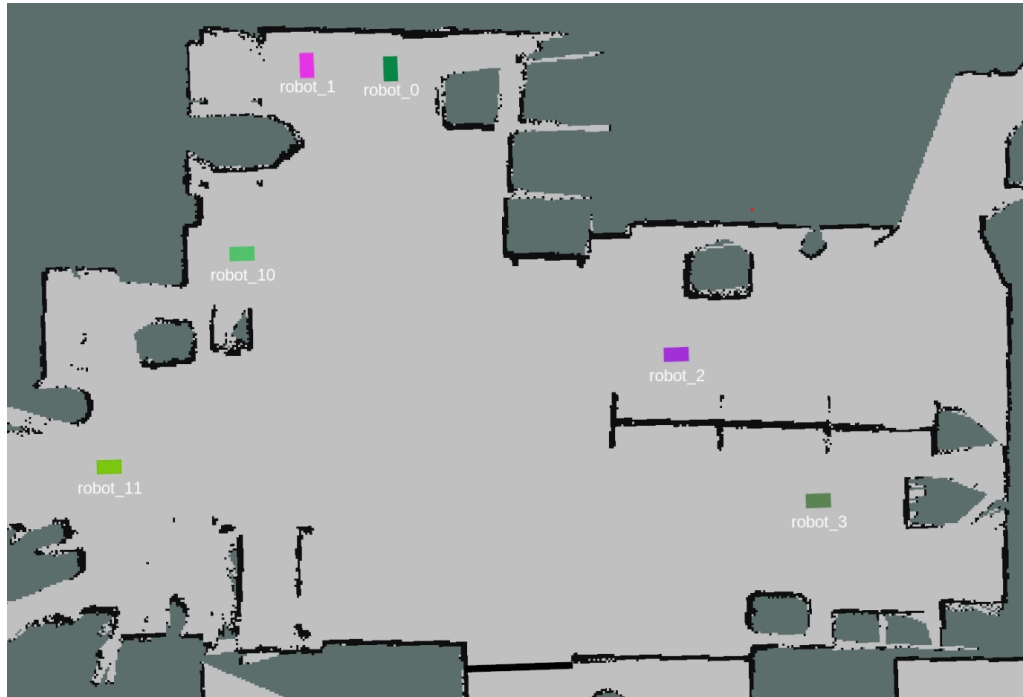


Figure 5.11: Increased Number of Robots in a Room

5.4.4 Summary

The Action Handler was developed to manage external actions, including the control of doors and elevators, which have the potential to influence the robot’s trajectory and execution time. The construction of the system as a modular ROS node facilitates its integration with real-world APIs. A key design objective was to ensure its operation could be independent of the central path planner. Utilising a state machine-based approach in conjunction with YAML, configured preconditions and postconditions facilitate proactive interaction between robots and shared resources. This approach has been shown to reduce delays and enhance execution flow in environments where timing is critical. The experimental results confirmed that the Action Handler significantly reduced task completion times and improved scalability in dense robot deployments. In conjunction with the Decomposer, it constitutes the evolution of the system into a coordination framework that is time- and resource-aware, and capable of robust operation in dynamic industrial settings.

Chapter 6

Conclusion and Future Work

This final chapter provides a concise overview of the primary outcomes of the research conducted and contemplates its contributions to enhancing the TEA* algorithm and the domain of multi-robot coordination. In addition, this study will provide insights into the future work required to improve the robot coordination system and the research undertaken in this dissertation.

6.1 Goals Accomplishments

The present dissertation addressed critical limitations in the TEA* algorithm, particularly its assumptions of uniform edge traversal times and negligible external action costs, assumptions that often fail in realistic, dynamic environments. Two modular components were developed and integrated into a fleet coordination system to overcome these constraints and enhance real-world applicability: the Decomposer Submodule and the Action Handler.

The Decomposer Submodule introduces temporal discretisation into the graph representation by dividing edges according to traversal durations and robot size, ensuring that all the edges have the same time to complete. This enhancement resulted in significant advancements in path planning accuracy, temporal synchronisation across agents, and reduced the probability of deadlocks and cascading replanning events.

The Action Handler was designed to incorporate logic-based control over time-sensitive interactions with shared resources, such as automatic doors or elevators. Integrating preconditions and postconditions within the planning process has been demonstrated to enhance the efficiency of the path planning algorithm TEA*, reduce delays, and improve task execution time.

Both components' performance was validated through simulation scenarios, which demonstrated significant benefits in system scalability, execution stability, and response to complex coordination tasks. It is also worth noting that the work resulted in a publication at a peer-reviewed conference [9], thus underlining the research's relevance and contribution.

6.2 Future Work

Notwithstanding the findings of this dissertation, some areas merit attention in future research:

- **Broader Action Support:** The action handler should be extended to encompass the management of additional resource types, including but not limited to elevators and charging stations, with due consideration for integrating these elements with real APIs and sensors.
- **Advanced Task Allocation:** The current FIFO task prioritisation mechanism should be improved by the introduction of superior methods of task prioritisation, or indeed entirely new methods that utilise artificial intelligence, for example.
- **Real-World Deployment:** The system should be tested in physical robot fleets operating in industrial environments to validate the modules under real-time constraints and variable sensor feedback. This work was impossible because of an unexpected setback in the usable state of the laboratory robots due to a mechanical problem.
- **Simulation Environment Enhancements:** The simulation framework must be improved to enhance large-scale testing support. This encompasses the optimisation of performance for elevated robot counts, the refinement of visual feedback within RViz, and the utilisation of the Stage simulation stack.
- **Dynamic Door Allocation:** Improve the Door Simulation module to support dynamic insertion and configuration of doors based on the environment map, rather than fixed coordinates. This will increase flexibility in testing different environments and make it easier to adapt to changes in real-world infrastructure.

References

- [1] G. Ullrich, “The history of automated guided vehicle systems,” *Automated Guided Vehicle Systems*, pp. 1–14, 2015. DOI: [10.1007/978-3-662-44814-4_1](https://doi.org/10.1007/978-3-662-44814-4_1).
- [2] K. Ellithy, M. Salah, I. S. Fahim, and R. Shalaby, “Agv and industry 4.0 in warehouses: A comprehensive analysis of existing literature and an innovative framework for flexible automation,” *International Journal of Advanced Manufacturing Technology*, vol. 134, pp. 15–38, 1-2 Sep. 2024, ISSN: 14333015. DOI: [10.1007/s00170-024-14127-0](https://doi.org/10.1007/s00170-024-14127-0).
- [3] K. Cai, C. Wang, J. Cheng, C. W. D. Silva, and M. Q. .-. Meng, “Mobile robot path planning in dynamic environments: A survey,” Jun. 2020. DOI: [10.15878/j.cnki.instrumentation.2019.02.010](https://doi.org/10.15878/j.cnki.instrumentation.2019.02.010). [Online]. Available: <https://arxiv.org/pdf/2006.14195>.
- [4] C. Rema, P. Costa, M. Silva, and E. J. S. Pires, “Task scheduling with mobile robots—a systematic literature review,” *Robotics 2025, Vol. 14, Page 75*, vol. 14, p. 75, 6 May 2025, ISSN: 2218-6581. DOI: [10.3390/ROBOTICS14060075](https://doi.org/10.3390/ROBOTICS14060075). [Online]. Available: <https://www.mdpi.com/2218-6581/14/6/75/htm%20https://www.mdpi.com/2218-6581/14/6/75>.
- [5] R. M. Gago, M. Galassi, F. Silveira, and G. M. Freitas, “A survey on robot fleet management systems for industrial processes,” *Proceedings of the 2024 Latin American Robotics Symposium, LARS 2024*, 2024. DOI: [10.1109/LARS64411.2024.10786445](https://doi.org/10.1109/LARS64411.2024.10786445).
- [6] J. Hažík, M. Dekan, P. Beňo, and F. Duchoň, “Fleet management system for an industry environment,” *Journal of Robotics and Control (JRC)*, vol. 3, pp. 779–789, 6 Dec. 2022, ISSN: 2715-5072. DOI: [10.18196/JRC.V3I6.16298](https://doi.org/10.18196/JRC.V3I6.16298). [Online]. Available: <https://journal.umy.ac.id/index.php/jrc/article/view/16298>.
- [7] P. Luís and C. G. D. Costa, “Planeamento cooperativo de tarefas e trajetórias em múltiplos robôs,” 2011. [Online]. Available: <https://repositorio-aberto.up.pt/handle/10216/62107>.
- [8] J. Santos, P. Costa, L. F. Rocha, A. P. Moreira, and G. Veiga, “Time enhanced a: Towards the development of a new approach for multi-robot coordinations,” *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 2015-June, pp. 3314–3319, June Jun. 2015. DOI: [10.1109/ICIT.2015.7125589](https://doi.org/10.1109/ICIT.2015.7125589).
- [9] F. Cardoso, D. M. Matos, M. Brilhante, P. Costat, H. Sobreira, and C. Silva, “Enhancing mobile robot navigation: A graph decomposition submodule for tea*,” in *2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2025, pp. 152–157. DOI: [10.1109/ICARSC65809.2025.10970181](https://doi.org/10.1109/ICARSC65809.2025.10970181).
- [10] Omg, “An omg ® unified modeling language ® publication omg ® unified modeling language ® (omg uml ®) omg document number: Date,” 2009. [Online]. Available: <https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi>.

- [11] M. N. Y. Ali and M. Z. H. Sarker, “An algorithm for avoiding deadlock,” *2005 Pakistan Section Multitopic Conference, INMIC*, 2005. DOI: [10.1109/INMIC.2005.334477](https://doi.org/10.1109/INMIC.2005.334477).
- [12] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, 1.10. Arpaci-Dusseau Books, Nov. 2023, Chapters 28–31. [Online]. Available: <https://pages.cs.wisc.edu/~Eremzi/OSTEP/#book-chapters>.
- [13] Y. Chen, C. Wang, M. Guo, and Z. Li, “Multi-robot trajectory planning with feasibility guarantee and deadlock resolution: An obstacle-dense environment,” *IEEE Robotics and Automation Letters*, vol. 8, pp. 2197–2204, 4 Oct. 2022, ISSN: 23773766. DOI: [10.1109/LRA.2023.3248377](https://doi.org/10.1109/LRA.2023.3248377). [Online]. Available: <https://arxiv.org/abs/2210.04231v2>.
- [14] C. Toumeh and A. Lambert, “Decentralized multi-agent planning using model predictive control and time-aware safe corridors,” *IEEE Robotics and Automation Letters*, vol. 7, pp. 11 110–11 117, 4 Oct. 2022, ISSN: 23773766. DOI: [10.1109/LRA.2022.3196777](https://doi.org/10.1109/LRA.2022.3196777).
- [15] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, “Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor,” *IEEE Robotics and Automation Letters*, vol. 7, pp. 4869–4876, 2 Apr. 2022, ISSN: 23773766. DOI: [10.1109/LRA.2022.3152702](https://doi.org/10.1109/LRA.2022.3152702).
- [16] M. Abdullhak and A. Vardy, “Deadlock prediction and recovery for distributed collision avoidance with buffered voronoi cells,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 429–436, 2021, ISSN: 21530866. DOI: [10.1109/IROS51168.2021.9636609](https://doi.org/10.1109/IROS51168.2021.9636609).
- [17] H. L. Chen, B. Hendriks, E. Torta, H. Bruyninckx, and R. van de Molengraft, “Behavior adaptation for mobile robots via semantic map compositions of constraint-based controllers,” *Frontiers in Robotics and AI*, vol. 10, p. 917637, Aug. 2023, ISSN: 22969144. DOI: [10.3389/FROBT.2023.917637/BIBTEX](https://doi.org/10.3389/FROBT.2023.917637/BIBTEX).
- [18] R. E. Fikes and N. J. Nilsson, “Strips: A new approach to the application of theorem proving to problem solving,” [Online]. Available: <https://www.cs.cmu.edu/~mmv/planning/readings/strips.pdf>.
- [19] A. Barrett, D. Christianson, M. Friedman, *et al.*, “Pddl | the planning domain definition language,” 1998. [Online]. Available: <https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf>.
- [20] V. R. Desai and J. P. How, “Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4956–4961, 2011, ISSN: 10504729. DOI: [10.1109/ICRA.2011.5980392](https://doi.org/10.1109/ICRA.2011.5980392).
- [21] A. Krnjak, I. Draganjac, S. Bogdan, T. Petrovic, D. Miklic, and Z. Kovacic, “Decentralized control of free ranging agvs in warehouse environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, pp. 2034–2041, June Jun. 2015, ISSN: 10504729. DOI: [10.1109/ICRA.2015.7139465](https://doi.org/10.1109/ICRA.2015.7139465).
- [22] A. Maoudj and A. L. Christensen, “Improved decentralized cooperative multi-agent path finding for robots with limited communication,” *Swarm Intelligence*, vol. 18, pp. 167–185, 2-3 Sep. 2024, ISSN: 19353820. DOI: [10.1007/s11721-023-00230-7](https://doi.org/10.1007/s11721-023-00230-7).

- [23] M. Berndt, D. Krummacker, C. Fischer, and H. D. Schotten, “Centralized robotic fleet coordination and control,” in *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*, 2021, pp. 1–8. [Online]. Available: <https://ieeexplore.ieee.org/document/9657192>.
- [24] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, “The hybrid path planning algorithm based on improved a* and artificial potential field for unmanned surface vehicle formations,” *Ocean Engineering*, vol. 223, p. 108 709, Mar. 2021, ISSN: 0029-8018. DOI: [10.1016/J.OCEANENG.2021.108709](https://doi.org/10.1016/J.OCEANENG.2021.108709).
- [25] D. M. Matos, P. Costa, H. Sobreira, A. Valente, and J. Lima, “Efficient multi-robot path planning in real environments: A centralized coordination system,” *2024*, 2024, ISSN: 2366598X. DOI: [10.1007/s41315-024-00378-3](https://doi.org/10.1007/s41315-024-00378-3).
- [26] B. Zhu, C. Li, L. Song, Y. Song, and Y. Li, “A algorithm of global path planning based on the grid map and v-graph environmental model for the mobile robot,” *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, vol. 2017-January, pp. 4973–4977, Dec. 2017. DOI: [10.1109/CAC.2017.8243661](https://doi.org/10.1109/CAC.2017.8243661).
- [27] A. Yang, Q. Niu, W. Zhao, K. Li, and G. W. Irwin, “An efficient algorithm for grid-based robotic path planning based on priority sorting of direction vectors,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6329 LNCS, pp. 456–466, PART 2 2010, ISSN: 1611-3349. DOI: [10.1007/978-3-642-15597-0_50](https://doi.org/10.1007/978-3-642-15597-0_50). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15597-0_50.
- [28] P. Moura, P. Costa, J. Lima, and P. Costa, “A temporal optimization applied to time enhanced a,” *AIP Conference Proceedings*, vol. 2116, Jul. 2019, ISSN: 15517616. DOI: [10.1063/1.5114225](https://doi.org/10.1063/1.5114225).
- [29] H. Ma, “Graph-based multi-robot path finding and planning,” *Current Robotics Reports*, vol. 3, pp. 77–84, 3 Jun. 2022. DOI: [10.1007/s43154-022-00083-8](https://doi.org/10.1007/s43154-022-00083-8). [Online]. Available: <https://link.springer.com/article/10.1007/s43154-022-00083-8>.
- [30] Z. Duraklı and V. Nabyev, “A new approach based on bezier curves to solve path planning problems for mobile robots,” *Journal of Computational Science*, vol. 58, p. 101 540, Feb. 2022, ISSN: 1877-7503. DOI: [10.1016/J.JOCS.2021.101540](https://doi.org/10.1016/J.JOCS.2021.101540).
- [31] Z. Šír and B. Jüttler, “On de casteljau-type algorithms for rational bézier curves,” *Journal of Computational and Applied Mathematics*, vol. 288, pp. 244–250, Nov. 2015, ISSN: 0377-0427. DOI: [10.1016/J.CAM.2015.01.037](https://doi.org/10.1016/J.CAM.2015.01.037).
- [32] J. Santos, P. Costa, L. Rocha, K. Vivaldini, A. P. Moreira, and G. Veiga, “Validation of a time based routing algorithm using a realistic automatic warehouse scenario,” *Advances in Intelligent Systems and Computing*, vol. 418, pp. 81–92, 2016, ISSN: 2194-5365. DOI: [10.1007/978-3-319-27149-1_7](https://doi.org/10.1007/978-3-319-27149-1_7).
- [33] D. Matos, P. Costa, J. Lima, and P. Costa, “Multi agv coordination tolerant to communication failures,” *Robotics 2021, Vol. 10, Page 55*, vol. 10, p. 55, 2 Mar. 2021, ISSN: 2218-6581. DOI: [10.3390/ROBOTICS10020055](https://doi.org/10.3390/ROBOTICS10020055).

- [34] D. Pereira, D. Matos, P. Rebelo, F. Ribeiro, P. Costa, and J. Lima, “Multi-robot coordination for a heterogeneous fleet of robots,” *Lecture Notes in Networks and Systems*, vol. 590 LNNS, pp. 229–240, 2023, ISSN: 2367-3389. DOI: [10.1007/978-3-031-21062-4_19](https://doi.org/10.1007/978-3-031-21062-4_19). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-21062-4_19.

Appendix A

Video Links

A.1 Action Handler in a Simple Map

- Without the use of Preconditions and Postconditions: <https://youtu.be/sgJtrT0S5mU>
- With the use of Preconditions and Postconditions: <https://youtu.be/5JuT09fNSZM>

A.2 Action Handler in the Laboratory Map

- Five robots using Preconditions and Postconditions: <https://youtu.be/0H338Cs4qX0>
- Five robots not using Preconditions and Postconditions: <https://youtu.be/sMdte1ynkLQ>
- Eight robots using Preconditions and Postconditions: <https://youtu.be/RNRV1Bm2xbU>