A novel deadlock control algorithm for a heterogeneous fleet of autonomous transport vehicles in a collaborative intralogistics environment

Ashwini Rathi

Delft University of Technology

TU Delft

accenture

A novel deadlock control algorithm for a heterogeneous fleet of autonomous transport vehicles in a collaborative intralogistics environment

by

Ashwini Rathi

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of
Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

Student number:	5479797	
MSc Track:	Multi-Machine engineering	
Report number	2023.MME.8870	
Thesis committee	Dr. R.R. Negenborn	TU Delft committee chair, Faculty 3mE
	Ir. M.B. Duinkerken	TU Delft, supervisor, Faculty 3mE
	Ing. Marloes Hengeveld	Accenture Industry X, Company supervisor
	Dr.Ir. Y. Pang	TU Delft committee member, Faculty 3mE
Date	12 November, 2023	

TU Delft

Preface

Dear reader,

This thesis marks an end to my journey as a student. It is indeed a good ending, as I carry with me a bundle of immense knowledge, good old memories, and laughs. Out of all my student years, this project year has been a masterpiece of all with full of drama, ups and downs, and a happy ending. This thesis has challenged me academically and personally. There were times I could relate to Michael Scott when he said *Sometimes I will start a sentence and I do not even know where it's going. I just hope I find it along the way.* Penning this last section brings satisfaction, as I am proud to document my research that lies before you.

The two main candidates in this journey are TU Delft and Accenture Industry X. I would like to thank TU Delft for providing an academic opportunity with economic support and making me believe in the "Can do mentality". Thank you Accenture Industry X for believing in me and providing me the opportunity to connect and learn from the industry experts.

I have many people to thank for being a part of this adventure year. First of all, thank you Mark for being my supervisor for almost a year. You have guided me straight from literature study to date with insightful feedback and support throughout the work which helped me stay on track. Thank you Rudy for your expertise, feedback sessions, and wise inputs for the project. Thank you Marloes for being my IX supervisor and helping me find this project, introducing me to the digital industry experts, helping me up my game in presentations, and developing my networking skills in Accenture Industry X.

During this journey, I met people who turned friends and good friends. Friends that provided advice, necessary distractions, and mainly good food. I would like to thank my volleyball group for keeping me sane during the stressful days. I would like to thank my housemates for the good company over a cup of chai and stress-bursting cooking sessions.

Lastly, I would like to thank my family in Europe and India for your support, and therapeutic talk sessions over the phone throughout these 2 years.

Happy reading!

Ashwini Rathi
Delft,
November 12, 2023

Abstract

For high efficiency and flexibility, a fleet of Automated Guided Vehicles (AGVs) both homogeneous and heterogeneous are widely used automation products for material handling in warehouses and automated production lines. Given the layout capacity, the AGVs interact with each other, which provokes challenges in Driverless Transport Vehicle System (DTVS) traffic management in dynamic environments. One of the main challenges is how to avoid collision and deadlock between heterogeneous fleets of AGVs in a bidirectional layout. This research study proposes a deadlock detection and avoidance algorithm that follows the deadlock prediction and uses a dynamic rerouting strategy with Dijkstra to avoid deadlocks. The mission paths are checked in space and time for overlapping edges by four check conditions and cumulative weights and return a boolean value to avoid cyclic deadlocks. For the heterogeneous fleet of AGVs, a standard communication protocol VDA5050 is used to maintain a standard communication interface between vehicles and the traffic management module with cloud-based microservices for increased processing time and interoperability within the warehouse. This communication interface is used to communicate the novel deadlock control algorithm to a heterogeneous fleet of AGVs. The proposed algorithm not only improves the throughput by increasing vehicle operational time but also successfully avoids congestion and deadlocks with high traffic management efficiency in the logistic transportation system.

Contents

List of Abbreviations

List of Figures

List of Tables

1 Introduction

A fun fact, according to [67], by 2050, there will be 24 billion interconnected devices! Under 'Factory of the Future' (FoF), the large-scale factory layouts are envisioned to operate with a highly integrated and well-organized foundation of knowledge. With changes in technology, economy, demands, and aim for sustainability, the FoF must be adaptable to these changing trends. For this, incredible advancements are being made in computer and communications technology to have advanced manufacturing operations in operational layout collaboration with human operators on site. It is said that the key to success for the FoF with Industry 4.0 (I4.0) is software developments that connect many parts of the factory such as logistics and production to establish reliable and coherent communication [65]. In Logistics and production, as a part of industry automation, controlling a fleet of Automated Guided Vehicles (AGV) is gaining high priority. With growing demands in production diversity and labor shortages, a heterogeneous portfolio of autonomous material handling solutions is expanding.

Data and information exchange between AGVs, FMS, and physical production systems are essential for autonomous operations to optimize the processes in FoF. These systems are called Cyber-Physical Production Systems (CPPS) and include developments in computer science (CS), information and communication technology (ICT) and manufacturing science and technology (MST) which eventually point towards the direction of the Industry 4.0 revolution [44]. An interplay between CS, ICT and MST involves machines equipped with data-generating electronics and a web of communication technologies with which a heterogeneous fleet of AGVs can be efficiently managed in the shared operational area [60].

For diverse demands, AGV from different manufacturers are often required to accomplish tasks, however, the challenge lies in the interoperability of these heterogeneous fleets in the same operational area effectively [38]. In this thesis, a Heterogeneous Fleet of Vehicles(HFoV) is defined as vehicles with different operational functions, different manufacturers, and mainly different Fleet Management Systems (FMS). Here, the HFoV is distinguished by structural heterogeneity and functional heterogeneity. The vehicles are considered structurally heterogeneous if they differ in design and dynamics, for example, bulk body, aerodynamic body, vehicle speed, payload capacity, fuel consumption, etc. On the other hand, functionally heterogeneous vehicles if not all vehicles are executing the same field of operation. For instance, a cleaning vehicle equipped with cleaning instruction software functionally works differently from a service vehicle whose software controls are for service pick-up and drop-off. These vehicles are assigned to visit the source point and target point while cleaning visits all the points in the configuration space [58].

1.1. Research Motivation

Modern logistics includes a wide area of characteristics ranging from systematic industry, a combination of logistics and information technology, integration of supply and services, and network architecture of the intelligent transport system [10]. The vital technologies for these modern logistics are sensors, intelligent chips, and wireless communication networks, which all together come

under the Internet of Things (IoT). For interoperability, IoT requires standards, and protocols that are implemented in the horizontal networks in the warehouse, such as middleware for fleet management, adapters for interface between the middleware and the physical assets, and programming/communication protocols across different devices from multi-vendors. Interoperability is one of the keywords in this research project. Interoperability is essential for the decision-makers to have an integrated view of the operations/performances of different machines in the factory. It indeed improves performance, reduces costs, and provides useful insights into the data collected from different machines with a standard format. Standard protocols are used for the data to be discovered, managed, communicated, and authenticated over different vehicles from different vendors operating in the warehouse [59].

However, establishing coordination between HFoV is still an area of open problem. There are various challenges with its autonomy, but before this let's discuss the current state of interoperability in the below section:

- Challenge 1 - Redundancy in integration: A vendor vehicle establishes communication and information flow of tasks or orders via a warehouse management system or warehouse execution system which is unique to it. This communication is installed via proprietary Application Programming Interface (APIs). This means, that if there is a new vehicle that is introduced in the factory, it has to go individual integration efforts to install its management system. In addition, the integration costs are higher.
- Challenge 2 - Task execution: With multiple warehouse management systems as a result of multi-vendor vehicles, these two different vehicles see each other as dynamic obstacles. In case they are on the same path, it causes a deadlock with each vehicle unable to resolve this conflict. Hence, with a heterogeneous fleet of vehicles, intended task allocation and task execution are highly inefficient and unsustainable with multiple warehouse management systems.
- Challenge 3 - Co-working/shared space difficulty: When different robots from different vendors work in the same operational area, they lack situational awareness and hinder the operations of one another. Thus requires separate operational space for each to operate efficiently.
- Challenge 4 - Low potential when performing a collaborative task: In case of a complex, elaborative task requires different fleets of vehicles to work together, it usually is operable in the presence of human operators to manually control these fleets as the vehicles have zero access of the current status or planned tasks of different vehicles.

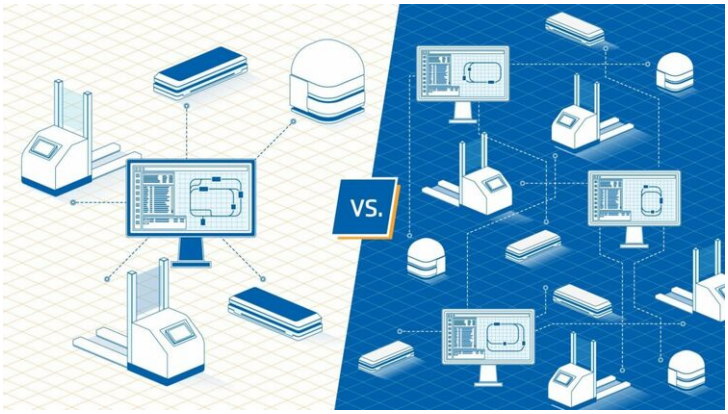


Figure 1.1: Compliant AGV to work together via one fleet management software (left), rather than different master controllers being needed for each brand of AGV working on-site (right) [62].

Figure 1.2: Navitec Fleet Control managing 4 different vehicles in the same fleet [66]

1.2. Research objective

To solve the challenges mentioned in 1.1, the traffic management techniques are developed and implemented with a communication network between the fleet of heterogeneous vehicles and FMS in order to automate the material handling operation as much as possible. Challenges 1,3 and 4 are seen as technical, while challenge 2 is seen as operational.

1.2.1. Operational research objective

Challenge 2, *task execution* with a HFoV in intralogistics with dynamic obstacles involves the efficient establishment of traffic-control policies. For a flexible, heterogeneous operational space, control policies for HFoV require collision-free and deadlock-free operations to evaluate the operational performance of the warehouse automation systems. The operational research objective, also the prime research, for this project is to develop a deadlock control algorithm to control both dynamic and static obstacles and deadlock situations, especially for a heterogeneous fleet of AGVs, where all the vehicles are operating in a common collaborative layout.

1.2.2. Technical research objective

In order to successfully transmit the traffic-control policies to HFoV, a standard communication protocol between a HFoV for the efficient flow of information, and integration is used. Challenges 1,3 and 4 can be solved with a standard communication interface as represented in figure 1.1. The standard communication interface VDA5050 enables a single master controller for both systems, the new and the existing vehicles (challenge 1). In addition, it also enables parallel operations (challenges 3 and 4) of AGVs (also vehicles with different degrees of autonomy) from different manufacturers and inventory systems in the same warehouse environment. Figure 1.2 shows the industries getting their hands on managing different vehicles in the same fleet, with the help of VDA5050. The technical research objective is to communicate the traffic policies of the developed deadlock control algorithm to a HFoV by introducing a standard communication interface called VDA5050 between the Traffic Management module of FMS and a HFoV.

1.3. Problem description

For a flexible material handling system, it is important that a HFoV be integrated together operating in a shared space in order for a smooth and efficient task execution with no conflict or deadlock situation. However, in the case of heterogeneous fleets, traffic control with deadlock-free operations, and handling dynamic obstacles in real-time, is still an ongoing area of research with many questions to solve. Hence, this thesis aims to develop a novel traffic control strategy with a deadlock detection and avoidance algorithm for a HFoV to improve the operational performance of vehicles in operational layouts with dynamic obstacles. The *traffic management module* embedded in FMS and the HFoV communicate via an industrial standard protocol and modular communication framework operating in the warehouse, in order to establish an efficient CPPS.

Main research question:
How to improve the deadlock control algorithm for efficient traffic management and operational performance of a heterogeneous fleet of AGVs in collaborative intralogistics operation?

Sub-Research Questions (SRQ):

- SRQ 1: What is the ongoing area of research in traffic control management for HFoV in collaborative intralogistics operations?
- SRQ 2: What is the framework of an Intelligent Transport System in a warehouse with HFoV and Why is standard communication interface VDA5050 required for warehouses with heterogeneous fleets of AGVs?
- SRQ 3: What are deadlock conditions defined in the traffic control policy of warehouse operations and how are these deadlocks handled?
- SRQ 4: How to design and develop a deadlock control algorithm? What necessary changes are required in developing the novel approach for this study?
- SRQ 5: How can the operational performance of the developed deadlock control algorithm of a HFoV's intelligent traffic management module be evaluated?

1.4. Scope

The number of AGVs, functionality and the number of missions/operational time are user-based input. These vehicles either start from a charging station or a rest station.
The main aim of this thesis work is to develop a deadlock detection and avoidance algorithm for HFoV and establish a communication network between FMS and a diverse fleet of vehicles. This thesis does not concentrate on optimizing algorithms. Various assumptions are taken into account:

- Layout is tessellated grid-based, with the centre of each cell as node and edges connected to pick-up, drop-off, charge, and rest locations. Each of these nodes has a location defined by x and y coordinates. Nodes, connected to each other via edges are defined and can be adaptable if in case the layout is changed. The size of one cell represents one vehicle. Some cells are marked as buffer zones (for ex, around inventory) in order to simulate a practical environment in case of turns.
- AGVs are guided and not autonomous, i.e., AGVs follow the guide nodes connected with edges in a straight line and turn whenever prompted.
- Inventory is assumed to be infinitely available in case of pick-ups and has infinite space in case of drop-off.

- Cleaning vehicle and AGVs are squared-shaped vehicles with lightweight-load capacity transport. Automated guided forklifts are rectangular in shape occupying two cells and three cells in case of turns with heavy-load capacity.
- There is no priority in vehicles while operation. It is first in first out according to the list of mission executions.
- It is assumed that each vehicle carries its own capacity package. Inventory management is out of the scope of this study.

1.5. Outline

Keeping readers' point-of-view in mind, this outline contains descriptions and information of all the chapters of this report and helps to navigate through it efficiently.

Chapter 1 discusses the research motivation, objectives, problem to be studied in this work, the scope of the problem area carried out in this project, related research questions and a brief outline of the report.

Following the introduction chapter, chapter 2 provides thorough research inspection in the domain of FMS, AGVs and communication networks. A literature review answers SRQ 1 to learn the existing traffic control policies and ongoing areas of research, leading to the investigation of the potential research gap that will be studied and analyzed in this project.

Chapter 3 answers SRQ 2, discusses the framework and modules of the Driverless Transport Vehicle System. In the broad spectrum, a thorough system analysis of the framework of the Driverless Transport Vehicle System, Intelligent Transport System, different modules and standard communication interface VDA5050 in the communication server module is studied. This chapter also achieves the *technical research objective* mentioned in 1.2.2.

After studying and learning about transport systems and modules, chapter 4 focuses on the analysis of the important module of study in this project embedded in FMS, *Traffic Management module* and answers SRQ 3. Introduction to deadlocks in warehouse logistics, different deadlock handling strategies and comparison of these strategies is carried out. With multi-criteria analysis, a promising approach suitable for the problem at hand is selected. The *operational research objectives* mentioned in 1.2.1 are answered one-by-one in and following this chapter.

After analyzing the *traffic management module* from the previous chapter, chapter 5 models a novel deadlock control algorithm by first developing deadlock detection checks, followed by a deadlock avoidance strategy, thus answering SRQ 4. This chapter also discusses the technical implementation of the developed approach, simulation set-up for experiments and verification and validation of the simulation model.

In order the evaluate the operational performance of the developed deadlock control algorithm, in chapter 6 different simulation experiments are conducted and results are studied. The performance of the developed deadlock control is studied and compared with the output of the traditional deadlock control approach to answer SRQ 5.

Finally, chapter 7 concludes the research and provides recommendations for future research on this topic.

2
Literature Review

The literature review provides knowledge and baseline research of the topic of interest, in this study, deadlock control policies for a fleet of AGVs in a collaborative operational layout. It is conducted to collect key resources, findings and potential research gaps that can be studied and filled in this project. The chapter is divided into two sections; a literature review technique used for this study and a review of the papers and their findings. The SRQ 1 *What is the ongoing area of research in traffic control management for HFoV in collaborative intralogistics operations?* is answered in this chapter with a finding of a potential research gap.

Keywords used for the review are: Deadlock handling strategies, warehouse operations, AGVs, Interoperability, FMS, communication framework, standard protocol

2.1. Literature review techniques used

To have an insightful literature review, the following steps are followed:

- Finding relevant and reliable literature or source of information is key for establishing a useful literature review. An electronic database with keyword search, termed as pearl growing method, to study the resources was employed here. The primary database was IEEE Explorer, ScienceDirect, as it provides access to a large bibliographic database related to science and technology worldwide and is easily accessible by TU Delft login. The secondary database used was Google Scholar, a freely accessible web search engine. Apart from the vast collection of research and articles, it was also used to cite the references in this document. Thirdly, the TU Delft education repository was also used for literature review to check if there are researches being carried out in the field of study of this project so that the personnel in the university can be approached for more information if necessary. Lastly, the World Wide Web (www) was used to look for familiar informational news, recent technological updates, company brochures and full-text research articles.
- When using the World Wide Web, to find credibility if a particular news article or research article is true and not misleading, the resources were searched again in ScienceDirect and Google Scholar to check for credibility. Another way to check for the credibility or truthfulness of a particle article is the name of the author who was searched to find if a researcher has relevant research outputs in a similar field of interest as that of the research article of his/her.
- The information for this review was successfully found by using keywords: *deadlocks, traffic control, warehouse operations, Industry 4.0, interoperability, FMS, communication framework, standard protocol*. These keywords were used according to their relevance and appropriateness to this literature review subject. To narrow down and find the most appropriate articles, the combination of keywords with the boolean search operator 'AND' in between was used.
- To analyze the resources, the criteria set reviewed researchers from reliable technological institutes, companies and educational technologies, date of publication of the articles, and other conducted research. For about 90% of the literature in this study, the date of publication was

restricted to articles published after 2000 to keep the review as updated as possible.

- In addition to the resource search from databases, the snowball method was employed to find additional relevant papers that are present in the reference list of the primary/starting point paper. This helps to find concise and relatable research in a short amount of time. Another method, the citation method (backwards approach) was minimally employed.

2.2. Research paper review

I4.0 features interoperability, virtualization, decentralization, real-time capability, service orientation and modularity are aimed to be implemented to maximize the automation in the manufacturing and warehouse industry. According to [18], the material handling process accounts for up to 70% of total manufacturing costs, which eventually leads many industries to shift to highly automated solutions for higher turnover. One of the systems under automated material handling operation is AGVs. The performance of the AGV system highly depends on the layout of the system (both cyber and physical layout) and the strategies used for controlling the vehicles. Control strategies for such AGV systems should at least perform the following three functions: path planning, task assignment, and traffic control [16]. The main challenge with the heterogeneous fleet of vehicles lies in the area of navigation, deadlocks created in case two different vehicles fail to recognize each other as vehicles but navigational blocks and halt themselves, failing to complete the missions successfully. In addition, the vehicles in the fleet are from different manufacturers or with different functionalities and come with their own FMS. In this case, it is a challenging task to operate these heterogeneous vehicles with different FMSs. It, thus, turns out to be a non-sustainable approach in case of the factory floor expansion which demands for increase in the vehicle types. Therefore, a one-in-all FMS with vehicles communicating bidirectionally with standard protocols is the next research area in *FoF*.

One of the issues related to AGVs as a result of implementing the I4.0 features with diverse AGVs customized for specific needs: normal and heavy load, forklift, tugger, etc. with different versions and software stacks [1]. Under the European H2020 cluster digital, industry and space, [55] implemented the starting step of establishing a unidirectional communication interface between Cyber-Physical System (CPS) and FMS, mainly for the system mapping and navigation, but bi-directional communication i.e., status updates from the AGV to the FMS is still an open-field of research in addition to managing a heterogeneous fleet of vehicles.
Keyword search of FMS for a fleet of robots/vehicles leads to papers with optimal path planning, scheduling and fleet monitoring. In the paper [52], a novel open interface to establish communication between FMS and a fleet of AGVs is studied. Using HMI (Human Machine Interface), a set of missions is communicated by the operators and the status of these missions is received. This is a significant stepping stone in designing a communication framework between the operator and the AGVs, indeed opening a path for more horizontal integration with heterogeneous vehicles. [71] designed an agent-based system architecture with each AGV as an agent decentralized/distributed network with AGVs communicating to each other in conditions such as path conflicting with cloud-edge computing environment, with interesting concept on AGV agent taking over the scheduling if the network connection fails between the edge node and the agent-based AGVs. The network loss in the factory floor results in the loss of status updates of running vehicles and can cause fatal accidents like collisions or deadlocks. [47] talks about insightful 5G-based smart manufacturing technology and its open area of research. Deadlocks, explained in [16] discusses three ways to avoid deadlocks, deadlock detection and recovery, deadlock avoidance and deadlock prevention on tessellated layout and uses an algorithm that makes sure that the layout remains deadlock-free within the limited known future movements of vehicles. Term heterogeneous AGVs are defined as vehicles

with different load-carrying capacities and functions (forklift, tugger, or just mobile) and study a collision-avoidance and deadlock avoidance algorithm. However, the heterogeneous fleet of vehicles also includes vehicles with different versions, control module stacks, communication software stacks, FMS and also different manufacturers.

There are numerous research papers with task allocation algorithms, path planning and conflict-resolution algorithms. However, these strategies are applied to the fleet of vehicles with at most different functionality. This is due to a lack of standardization within the communication network between the management systems and the fleet of vehicles. [60] thesis work discusses one of the standard messaging protocols, pub-sub machine-to-machine interface called Message Queuing Telemetry Transport (MQTT) for heterogeneous fleet of vehicles. However, the study is limited to future work suggestions on conflict-free paths. In brief, a thorough literature search shows a lack of study on the control strategies of the fleet of vehicles with varying software versions for communication, FMS or manufacturers. In addition, an open interface standard of communication for vehicles with different versions, manufacturers and software stacks is currently an active area of research.
The performance of the AGV system highly depends on the layout of the system (both cyber and physical layout) and the strategies used for controlling the vehicles. Control strategies for such AGV systems should at least perform the following three functions: path planning, task assignment, and traffic control [16]. The main challenge with the HFoV lies in the area of navigation, deadlocks created in case two different vehicles fail to recognize each other as vehicles but navigational blocks and halt themselves, failing to complete the missions successfully. In addition, the vehicles in the fleet are from different manufacturers or with different functionalities and come with their own central controller. In this case, it is a challenging task to operate these heterogeneous vehicles with different central controllers. It, thus, turns out to be a non-sustainable approach in case of the factory floor expansion which demands for increase in the vehicle types. Therefore, a one-in-all central controller with vehicles communicating bidirectionally with standard protocols is the next research area in *factory of future*.

Deadlocks, explained in this article discusses three ways to avoid deadlocks, deadlock detection and recovery, deadlock avoidance and deadlock prevention and uses an algorithm that makes sure that the layout remains deadlock-free within the limited known future movements of vehicles. Deadlock prevention is an offline strategy, where the system schedules paths in such a way that is deadlock-free in prior. For bidirectional layout and higher throughput, a prevention strategy with faster network routing is beneficial in terms of cost reduction with a reduction in used areas by vehicles [19], but however limits in preventing deadlocks in case of dynamic disturbances in the operational layout. With time-based route planning, certain segments/cells in the route of a vehicle are allocated and the rest of the segments in the route are utilized by other vehicles. In the paper [32], each node in a cell of the operational layout has a list of time windows reserved by vehicles and a list of free time window that is available for the vehicles to reserve. In this way, the algorithm plans the vehicle route with free time windows in the proposed time-window graph instead of physical cell nodes of the operational flow path. The computation time is then $o(v^4n^2)$ where v is the number of vehicles and n is the number of nodes. This means, that with an increase in the number of vehicles, the computational complexity increases. This is suitable for only small transport systems. Factors such as acceleration, deceleration, and external obstacles in dynamic environments make it difficult to calculate time windows precisely in case of delays and can lead to unpredictable obstacles.
Similar to the prevention strategy, deadlock avoidance avoids the occurrence of the deadlocks which are passive in nature. This policy plans the mission operations dynamically depending on the system state such that it remains deadlock-free. Resource allocation graphs are useful for detecting deadlocks. Dynamic resource allocation can affect the resource/zone utilization and system throughput, and eventually increase the lead time and vehicle travel time [72]. In terms of transportation systems,

a Banker's algorithm is a resource allocation and deadlock avoidance strategy that predetermines if the system will remain in *safe state* or not by first simulating the allocation of node(s)/cell(s) to the mission paths. It then makes a *safe state* check before actually allocating the node(s)/cell(s) to the mission to navigate through them in order to avoid deadlock. In article [16], the central controller calls deadlock avoidance every time to check if a particular vehicle is allowed to *reserve* a set of cells/tiles required for its navigation. The algorithm determines if the reservation is allowed only if the order of movements of AGVs exists such that the system remains deadlock-free. The combination of steps of different vehicles is checked by the algorithm to evaluate the system state, which causes unnecessary deterring of AGVs and longer calculation time. A modified Banker's algorithm strategy proposed by [29] provides a solution to solve the low utilization of vehicles and longer waiting times. Under special circumstances, the vehicles are allowed to transverse unsafe states (deadlock existing states) to decrease the vehicle waiting times and mission execution times. This extended set of states is allowed only if all the missions can be executed safely, or else the vehicles wait until the state is safe. An extension of this modification is proposed by [24] for an improved near-optimal deadlock avoidance strategy. It consists of two stages, an offline algorithm that preprocesss guide-path and an online which combines preprocessed guide-path results with the vehicle status to evaluate the safety of the system dynamically. In order to avoid the vehicles waiting until the overlapped edges of two paths are released, alternative shortest paths are obtained. Before a path from alternative paths is allocated to a vehicle, a safety assessment of the system is carried out. If all these alternative paths lead to an unsafe state, the vehicle is instructed to pause and wait for the next scheduled check. This questions the trade-off between travel distance in case of alternative path allotment or waiting time until the system gets back to *safe* state.

Most of these papers discuss detecting deadlock edge-by-edge, and then prompting the deadlock handling strategy to either prevent/avoid them. However, detecting deadlocks in prior just after path planning and before vehicle movement execution is a factor to consider. Deadlock detection in prior can help in time management to control deadlocks, avoiding cyclic deadlock situations and can positively affect the system throughput. This is the potential research gap of deadlock detection in advance for the path and not edge-by-edge, which will be developed in this study and its operational performance will be evaluated.

3

Driverless Transport Vehicle System (DTVS) Analysis

A driverless transport system is made of different components that share information with each other in order to accomplish a mission or order. The purpose of this chapter is to study the framework of the system in a broader spectrum, understand the different modules and their functions, and also learn the intelligence that is introduced under Industry 4.0 in these modules for efficient warehouse operations. First, the framework of intelligent transport system for DTVS is represented with modules, these modules and their sub-embedded systems are explained with their working and functionalities. Secondly, in this project, the emphasis is on introducing a heterogeneous fleet of AGVs in the warehouse, which then prompts a standard communication interface VDA5050 to be used. The specifics and working of VDA5050 are also discussed.
This chapter answers SRQ 2 about the Intelligent Transport System framework and role of standard communication interface VDA5050. In addition, this chapter achieves the technical objectives discussed in 1.2.2.

3.1. DTVS-based warehouse operations

In manufacturing and warehousing, a DTVS consisting of a fleet of vehicles (homogeneous and heterogeneous) is becoming increasingly crucial for the efficient transport of goods. In comparison to belt-conveyor, sorter system or human-operated forklift, DTVS is proving to be a promising alternative with advantages in terms of flexible deployment, less human-prone errors in the operation with real-time routing and rescheduling.
With varying demands and supply for varied items, the heterogeneity among these vehicles is asked for. In order to evaluate the performance of the DTVS-based warehouse operations, warehouse engineers choose suitable traffic control policies and warehouse layouts according to the requirements. In this project, one of the evaluation factors, traffic control policies mainly focusing on collision-free and deadlock control during the operation is studied.

A fundamental challenge is implementing the traffic control policy in an operational environment with HFoV for collision and deadlock-free strategies and no manual intervention, with the aim of minimizing total cost in terms of computation and waiting time. For the traffic management module to plan collision and deadlock-free routes, it is necessary that the vehicle update its position. This position update can be vehicle software-oriented and differs in data format from vehicle to vehicle in the case of a heterogeneous fleet. To compute these different data formats can be computationally expensive. It is necessary to establish standard data communication protocols so that the data collected by the traffic management module is of the same format and can be analyzed and processed faster.
To address the computation challenge, an I4.0 technology to take benefit from to access unlimited processing power and storage is *cloud computing*. A worldwide network of interconnected objects that can be addressed uniquely, and based on standard communication protocols is termed as Internet of

Things (*IoT)*. This technology is used to establish a communication network with standard protocols. For example in warehouse/logistics, RFID(Radio-frequency identification)-enabled vehicles that use sensors and actuators can be connected to better access to the position, temperature, and status and gather useful results. This information can then be stored virtually in cloud-based services to be used by different system management (warehouse floor departments, inventory management system, execution system). Merging *cloud computing* and *IoT* to bring a new paradigm is termed as *CloudIoT* [8]. In this study, software stacks (navigation stack) of heterogeneous driverless vehicles form a network of interconnected devices that are connected to the Warehouse Management System with a standard communication protocol and for increased storage and processing units, cloud-based microservices are exploited. This arrangement is called as *Intelligent Transportation System.*

3.2. Framework of Intelligent Transportation System for DTVS

In this study, DTVS is an authorization-based system where vehicles need permission from DTVS for every action to perform. The system consists of three modules, a warehouse management module, a fleet management system (central controller) and a communication server (wired/wireless) as represented in figure 3.1.

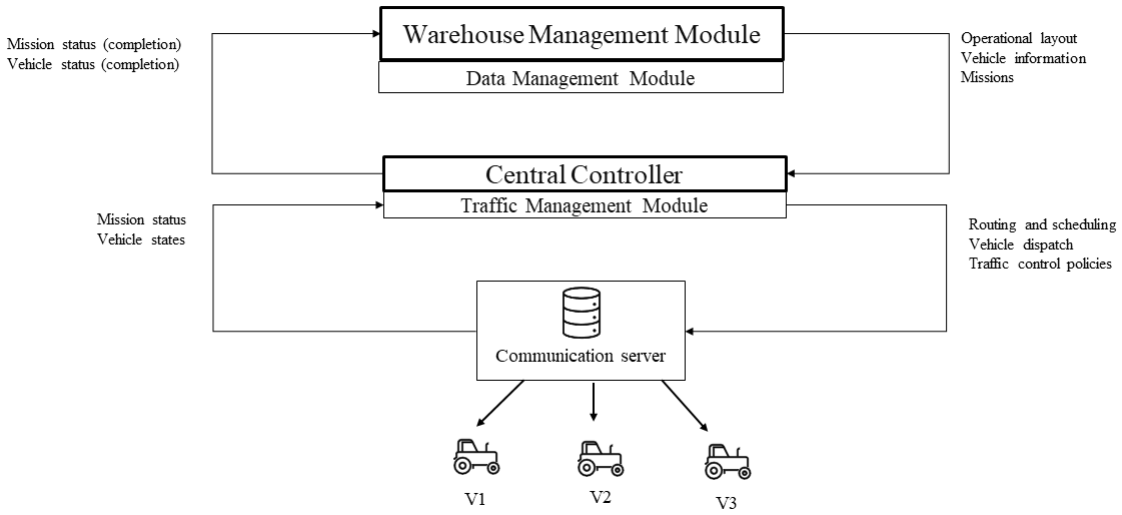


Figure 3.1: Driverless Transport Vehicle System modules

- Warehouse Management Module: The data management module consists of information about each vehicle in the fleet, mission requests and operational space information. This data is input into the fleet management system. The data consists of the x and y coordinates of nodes defined in the operational layout, and weights on the edges connecting the nodes. Weight can be the distance between nodes or the time taken for the vehicles between two connected nodes. Here, weight is the distance.
- Fleet Management System (central controller): It is considered as *high-level fleet management* which supervises the fleet of vehicles by planning, coordinating and controlling the fleet. It provides necessary instructions to the vehicles to accomplish the order pick-up/drop-off. The traffic management module is sub-embedded in the Fleet Management System.
- Communication server: In order to send/receive the mission information to and from a fleet of vehicles, communication servers are used. A standard interface of communications is required for a fleet of heterogeneous vehicles.

Important modules to focus on in this project are a central controller with the traffic management

module which runs the traffic control policies and a Communication server module to establish *CloudIoT*-based standard communication protocol. A representation of this Intelligent Transportation System is shown in figure 3.2. The Intelligent Transportation System modules are only used in this study to implement the developed deadlock control algorithm HFoV.

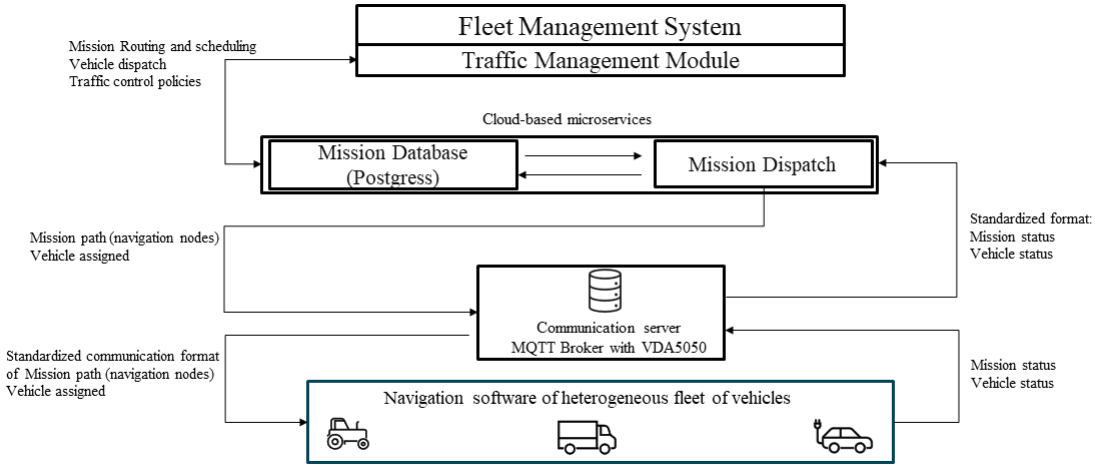


Figure 3.2: Intelligent Transportation System with *CloudIoT*

The system architecture represented in figure 3.2 can be divided into 3 layers, the back end, the communication server and the front end. The back-end consists of a Fleet Management System with a Traffic Management Module. The front-end is a simulation-based Graphical User Interface(GUI) of the navigation of the HFoV.

3.3. Intelligent Transportation System: Back-end module

3.3.1. Fleet Management System

FMS, for short, is usually called 'high-level' fleet management, which provides a wide range of solutions comprising different fleet-related applications in the transportation, logistics and supply chain industry.
A vehicle transportation system can be divided into two areas; transportation system design and transportation system management [41], depending on tactical and operational issues. Transportation system design consists of improving the system performance by studying and estimating guide-path design, the number of parking, pick-up and drop locations in the operational layout and the number of vehicles required. Transportation system management (Fleet Management System) includes vehicle dispatching, routing and scheduling, traffic control for collision and deadlock avoidance and maintenance strategies.

- Mission routing and scheduling: Missions. orders received from the Warehouse Management system, the mission route is first planned and assigned to the vehicle. The scheduling takes into account the departure and arrival time from the pick-up location to the delivery location, accounting for the cumulative weights (distance between two connected nodes) from the pick-up to the drop-off location. This cumulative weight calculation is used for collision and deadlock-free scheduling.
- Vehicle dispatching: Dispatching is when an order is assigned to the vehicle or vehicle to the order. Dispatching methods can vary from time constraints, priorities, or the nearest idle vehicle available. The simple dispatching rule is assigning an order to the idle or next available vehicle with the shortest or nearest travel distance/time from the order location. This rule is

followed in the current study.

- Traffic control policies: A planned route is feasible for vehicle operation with appropriate traffic control policies to account for collisions and deadlocks. These control strategies are closely related to dispatching, routing and scheduling. Collisions are prevented by the posing rule that no two vehicles can occupy the same zone/position at the same time. However, this rule can cause congestion and lead to deadlocks.

Based on available transportation resources and constraints at the application/operation site, it has different functions ranging from mission routing and scheduling, vehicle dispatching and traffic management. A study [73] describes the clear objective of FMS is to reduce security risk, increase the quality of service of operation and in turn increase the operational efficiency by minimizing the costs.

From a system control perspective, the system architecture has two categories, centralized and distributed [20]. A centralized architecture meets the objectives of this research that are suitable for modern vehicle fleet planning and coordination. However, centrally processing large amounts of planning and coordination data sounds inefficient, but this is also a good approach to obtain globally optimal solutions in case of task planning, allocation and execution, thus providing an effective decision-support tool. Another problem with centralized architecture is computational expensiveness. This can be solved by outsourcing cloud-based infrastructure for computationally heavy processes, In addition, these systems come with sophisticated cooling systems that tick green with energy-efficient system goals.

One major disadvantage of centralized architecture is single-point failure. This system is less robust with system faults. For ex, the vehicles can establish communication with each other only via the central master controller. In case the central controller faces network downtime, then this poses a single-point failure problem with mission-critical operations. It is assumed that a duplicate version can act as clone-FMS in case of system downtime [6]. In case of network loss/disconnectivity, the mission database is a cloud-based microservice that can rebuild the states and continue from the information stored in the database, so the mission is not lost, reference figure 3.2. It can be believed that the system, instead of leading to downtime, will ensure that it maintains operability. Cloud-based services are discussed below.

3.3.2. Cloud-based microservices

It is 'low-level' fleet management that runs in the cloud. The microservice module manages communication with the software stacks of the vehicles (navigational stack here). In addition, it is responsible for executing the mission, keeping track and collecting feedback information such as position, status, and error from the vehicles and feeding it to the FMS.

Cloud-based microservice: Mission database
The database hosts REST API endpoints, basically, any mission submitted is stored in this database and is managed persistently. This is done using Postgres (also called PostgreSQL), which is an open-source object-relational data management system. It manages a locally hosted database, here mission database. In case of connectivity loss, the services crash and come back up for FMS, the missions in the database can rebuild the states and continue from the information stored in the database, so the mission is not lost. The APIs are used to create, post, and get the mission states and updates from the API mission/vehicle object.

Cloud-based microservice: Mission dispatch
This sub-module manages the mission state transition of the vehicles and handles communication

between the FMS and the fleet. It enables FMS to submit the planned mission to the fleet of vehicles. It provides a connection between the FMS and the fleet but does not handle control strategies like vehicle dispatch with mission or traffic control. Mission dispatch module maps missions in the form of a series of tasks which are translated to VDA5050 Order messages and sent to the vehicle for execution. The connection between this cloud control service and vehicles relies on an industry-standard called VDA5050 and uses MQTT as a light-weight pub-sub, machine-to-machine network protocol explained below.

3.4. Intelligent Transportation System: Communication server

A communication server allows industries to build networked infrastructure to deploy communication systems (wired/wireless) for the transfer of information or data.

3.4.1. MQTT broker

MQTT broker is used for communication between the Mission dispatch module and the navigation stack of the vehicle. MQTT is a lightweight, text-based message exchange protocol (message broker) with IoT publish/subscribe model and consists of clients that are divided into subscriber and publisher. The subscriber is a message message-receiving client who is registered with the broker and notifies it to receive specific types of messages. The publisher is a message message-sending client who sends a message to the subscriber when asked through the broker [26] [25]. Here, Mosquitto is used as MQTT broker.

Mosquito is a lightweight MQTT broker with the capability to exchange large amounts of data over low network overhead, with limited network bandwidth and interrupted communication. Thus, it can be implemented on low-power devices like micro-controllers used in remote IoT sensors. MQTT allows distribution of messages to sub-channels termed as *topics*. Clients (here mission dispatch module and software on vehicle) subscribe to these topics to receive required information that interests them. The topics and clients with pub/sub are represented in figure 3.3. Topics concerning to current study are Order (Communication of driving orders like nodes to navigate from FMS to vehicle), Action (Action to be executed sent from FMS to vehicle), instantActions (any immediate actions to be executed), state (vehicle state) and factsheet (vehicle setup). An explanation of these is provided in the next section.

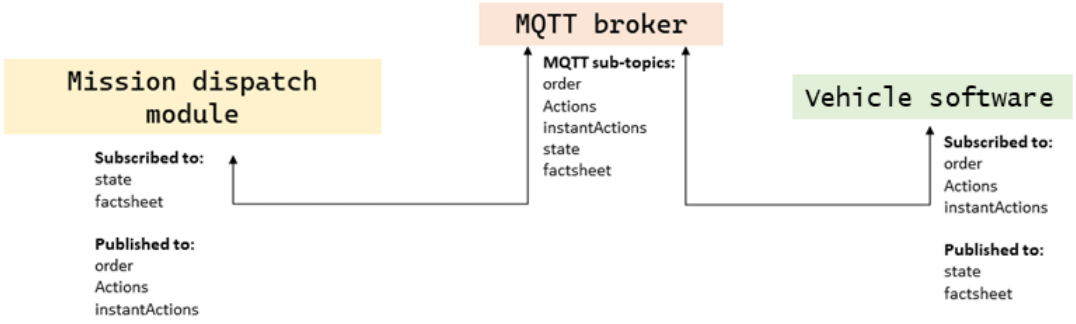


Figure 3.3: MQTT broker Mosquitto with publisher/subscriber

3.4.2. Industrial standard protocol - VDA5050

The communication server acts as a bridge between the cloud/edge microservices like the Mission Dispatch module and the vehicle in order to exchange topics (order, status) between the FMS and

software on vehicles. This interface has some requirements such as it has to be agnostic with different software stacks such as **ROS!** or Isaac SDK. In addition, the interface should facilitate a simplified connection strategy of new vehicles into an existing FMS, and enable parallel operation between vehicles from different manufacturers and inventory systems in the same operational environment. It should be cloud-friendly and scalable to large numbers of vehicles. This interface helps achieve the technical research objective drawn in 1.2.2 and provides solutions to challenges 1, 3 and 4 mentioned in 1.1.

To meet these requirements, an interface, VDA5050, was established in 2022 between the Verband der Automobilindustrie e.V. (German abbreviation VDA) and Verband Deutscher Maschinen-und Anlagenbau e.V. (German abbreviation VDMA) with an aim to create universally applicable interface tool. It defines a messaging structure and uses MQTT network protocol to publish/subscribe the message structures. There are two objectives of the interface that are of interest in this work:

1. Increase in *Plug & Play* capability the system by using uniform, overall coordination logic between all transport vehicles, models and manufacturers
2. Using a common interface between FMS and vehicle control will increase vehicle manufacturer's independence

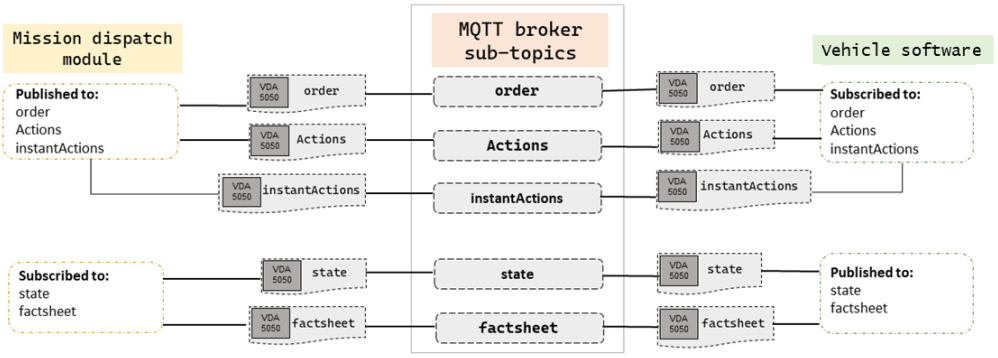


Figure 3.4: Information flow of pub/sub topics between modules with VDA5050 specification

In [42], [31], are informative references where a Digital twin is used to build a VDA5050-compliant communication interface between the controller and the vehicles. To execute missions, the Mission dispatch module receives control strategies from FMS and publishes the order, Actions or instantActions to corresponding VDA5050 subtopic with VDA5050 compliant message structure. The vehicle software stack receives the order, Actions or instantiations via MQTT broker. Update vehicle state and missions are published by the corresponding state, factsheet which are subscribed by Mission dispatch module via MQTT broker. The flow of this information is represented in figure 3.4. Information flow designed by VDA and VDMA, about VDA5050, message schema of these subtopics, and mission tree message structure is explained in appendix 9.

3.5. Intelligent Transportation System: Front-end modules

Front-end modules are the PC onboard vehicles connected to the PC operated by the operator offsite. Operators can input necessary information like the operational warehouse layout map, the missions to be completed manually and vehicles required for the missions or connect it with the Warehouse Management System for automatic generation of the information.

3.6. Summary

Driverless Transport System framework consists of three modules, *Warehouse Management module* with embedded *Data Management Module*, *Central Controller* with embedded *traffic management module* and *Communication server* which is in direct communication with the homogeneous fleet of vehicles. In order to introduce a heterogeneous fleet of vehicles, the driverless transport system is upgraded to an Intelligent Transport System. The Intelligent Transport System has two added features, *cloud-based* services with mission database and dispatch in the cloud and communication server embeds MQTT broker with VDA5050 industrial standard communication interface for a heterogeneous fleet of vehicles.

For faster processing, an upgrade of the driverless transport system to *intelligent* driverless transport system is necessary. I4.0 technologies combination; cloud-based and IoT called as *CloudIoT* is implemented in the framework. In these cases, an adaptable traffic management module with a robust, scalable algorithm with traffic control policies is ongoing research. This Intelligent Transport System is centralized management in order to achieve global optimization. For the computational expensiveness of processing information in centralized architecture, an intelligent system with cloud-based infrastructure, and for real-time mission and vehicle position updates an interconnected network of devices is established in an intelligent system.

In addition, an intelligent system also provides a standard communication interface VDA5050 as heterogeneous vehicles share information in different, manufacturer-specific formats. Each vehicle from different manufacturing comes with its unique warehouse management system. VDA5050 helps to solve the redundancy in the integration of new vehicles into the existing Warehouse Management System, hence eliminating the individual integration of each new vehicle to install its management system and reducing integration-related costs. If vehicles from different manufacturers are operating in the warehouse, they lack situational awareness and cause collisions. Due to this, these vehicles are allocated separate bounded operational areas for each. By managing a heterogeneous fleet under one warehouse management system, VDA5050 allows communication of each operating vehicle status in the system to be shared up to date with one warehouse management system, hence allowing a heterogeneous fleet of vehicles co-working in shared space. In case a particular mission requires a heterogeneous fleet of vehicles, this can be possible with VDA5050 standard communication and also without the need for the operator to manually control the fleet.

4

Traffic Management Module System Analysis

The previous chapter discussed Intelligent Transport System modules on a broader spectrum and achieving the technical research objective of this project. To narrow down and study the prime purpose of this project, traffic management module system analysis is carried out to study the traffic control policies, deadlock conditions, and deadlock handling strategies required to control deadlocks. This chapter will also draw necessary insights into existing deadlock control strategies providing knowledge to develop a novel deadlock control algorithm for a heterogeneous fleet of AGVs operating in collaborative warehouse logistics. This chapter consists of analyzing transport system design, vehicle management, conditions for deadlocks, different deadlock handling strategies, and their multi-criteria analysis which helps find weak spots in the existing deadlock handling approaches. Furthermore, to evaluate the operational performance, key performance indicators are discussed. This chapter helps answer SRQ 3 about different deadlock conditions and their handling strategies.

4.1. Traffic management module of DTVS

An embedded module *Traffic Management Module* has the functionality of traffic control strategies with rules and algorithms, one of the evaluating factors of the operational performance of DTVS. It mainly focuses on collision and deadlock handling during the operation of AGVs. Automated fleet operations in warehouses have a stronger foundation with infrastructures like robust and resilient fleets of heterogeneous and diverse vehicles. One fundamental problem with heterogeneous multi-vehicles is a multi-vehicle traffic control strategy for collision-free paths or deadlock-free path planning to each vehicle in the network [46]. Hence, collision and deadlock handling strategies with multi-vehicle, especially heterogeneous systems have a critical study discussion in this thesis.
For collision and deadlock-free paths, the traffic management module of DTVS plans mission dispatch, routing, and scheduling and are handled concurrently. A baseline function in these three functionalities is path planning. Path planning for the vehicle to transverse from source to target and re-planning in case of obstacles in order to avoid collision and deadlock [36] [12]. It is a non-deterministic polynomial-time (NP)-hard problem because, with an increase in the degree of freedom of the system, the complexity increases. Suitable factors such as layout design, vehicle control, and information flow architecture based on system requirements are to be selected in order to increase the DTVS transportation efficiency. A combination of DTVS transportation system design, vehicle management, and control is studied below.

4.1.1. Transportation system design

Transportation system design discusses the structure of flow-path layouts to design the transportation of vehicles. Flow-path layouts are classified into *unidirectional*, *bidirectional*, and *multi-lane* flow paths. With a unidirectional path, no opposite traffic is allowed, hence requiring simple controls. One-way traffic has less layout utilization and a higher vehicle traveling distance with less

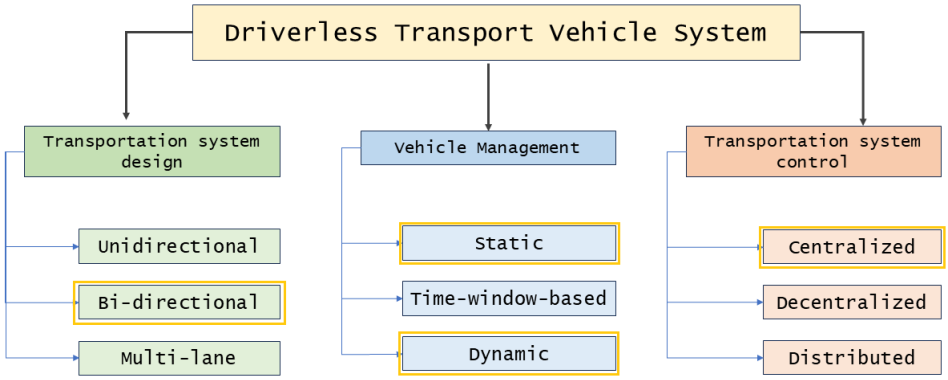


Figure 4.1: DTVS transportation, management and control design selection

number of missions completed. Bidirectional flow allows two-way traffic, facilitates transportation cost reduction, and minimizes the area used by increasing the transportation network and hence higher throughput. A bidirectional path with a vast networked layout requires a smaller number of vehicles and is more advantageous than a unidirectional one. Due to bidirectional traffic, congestion increases and system complexity increases. This then requires an adequate traffic control algorithm. In a bi-directional graphical network layout, vehicles travel in opposite directions. It is then necessary to check the accessible route available for vehicles in order to avoid collisions or deadlocks when traveling in the opposite direction. That means mission scheduling and vehicle dispatching are closely related to vehicle routing. It is hence necessary logically to integrate mission routing, scheduling, and vehicle dispatching as an entire solution for the DTVS control [69]. A suitable and adequate traffic control algorithm for the bidirectional flow of vehicles is required.

4.1.2. Transportation system control

Transportation system control is classified as *centralized*, *decentralized* and *distributed*. In decentralized control, the decisions are made based on local information. Here, the vehicles are aware of their state and the state of neighboring vehicles. The traffic management is handled and communicated between these vehicles and negotiated by themselves. Decentralized control has low computation and a simpler solution approach but has low efficiency. A distributed control, algorithm uses a combination of global and local information. In the Intelligent Transport System framework set in chapter 3, only global information flow is implemented. Local information flow increases the system's control complexity. For this study, a simpler global information flow is considered and hence distributed control is out-of-scope of this study. In centralized control, all the information about the vehicles, positions, planning, and coordination of transportation systems are stored and computed in one place. It is highly efficient but requires longer computation time. In order to reach a globally optimal solution, centralized control is a feasible approach. For the computational expensiveness of centralized control, outsourcing cloud-based infrastructure for computationally heavy processes is employed as discussed in section 3.3.1.

4.1.3. Vehicle Management

In vehicle management, the updates received from the vehicle and mission are used to plan and control traffic. Traffic control algorithms are classified as *static*, *time-window-based* and *dynamic*. Static algorithms plan all the mission routes in advance and fail to react in case of a dynamic traffic condition. This method is suitable for smaller transportation systems with low throughput and global planning. In time-window-based algorithms, each vehicle with a flow path is divided into

segments, where one segment is partially for that time is occupied by one vehicle and the rest of the other segments can be shared by other vehicles. This allows for better flow-path utilization and higher transportation system throughput. When all the orders, sources, targets, and schedule times are known prior to the operation, the system employs static and time-window-based algorithms. Dynamic algorithms plan the path based on real-time information about the traffic. In case of missions are submitted or arrive in a sequential manner and route decisions are made without the information about the missions arriving later, the algorithm is capable of real-time operations.

The transportation system design is set to bidirectional with a networked layout with the aim of higher system throughput. The control of the Intelligent Transport System is centralized with cloud-based services. Control algorithms are required for vehicle and traffic management, these are selected and modeled depending on the components of the system (here HFoV, obstacles, and kind of deadlocks to be handled in the system. The rest of this chapter will deep dive into the deadlock conditions and handling deadlocks occurring in HFoV in the DTVS. A morphological overview of all possible deadlock handling strategies with control policies and multi-criteria analysis of these strategies is carried out to select the most suitable strategy/strategies in the context of the related problem of this project.

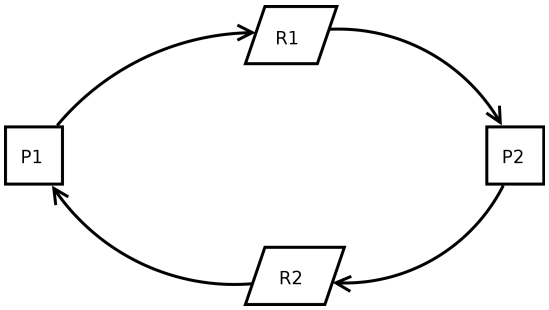
4.2. Conditions and categorization of deadlocks

It is interesting to notice that the term deadlock was first coined not from logistics but from computer science, where this problem was recognized and discussed in the late 1960s and early 1970s [11] [23]. During primary parallel/ multi-programming, deadlocks occurred and thus strategies were developed to break the conditions for deadlock occurrence. In multiprogramming, when two programs in the computer share the same resource, they eventually prevent each other from accessing that resource effectively causing the operational halt of the program or technical ceasure. Here, resources can be defined as devices (tape/disk, drives, card readers, etc.), processors, or storage media in the context of computer science and can be information data (tables, files, etc), programs, or routes in the context of logistics. Coffman et al. proposed four conditions for deadlock occurrence and since then have been used to investigate the occurrence of deadlocks and their handling strategies in logistics [5].

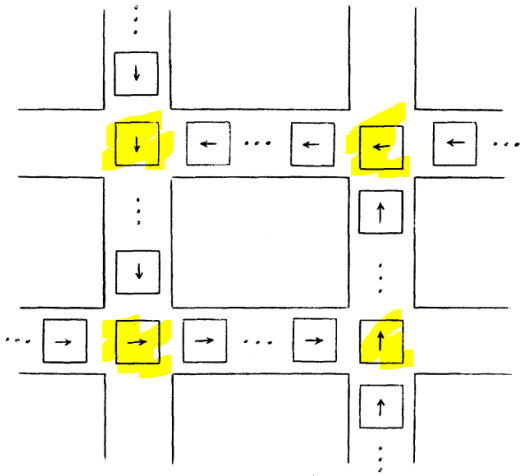
The same deadlock situation occurring in the computer system can be related to a deadlock occurring between two driverless vehicles in a traffic management system of the warehouse. The resources shared between these vehicles can be data, cells/routes in the operational layout or positions that can lead to potential deadlock. When two vehicles reserve the same cell to traverse through the planned route, it causes deadlock as each vehicle strives to travel to that reserved cell which is currently occupied by another vehicle [11] and is represented in figure 4.2.

For a situation to be posed as deadlock, Coffman coined four conditions as explained below [34]:

- *Mutual exclusion*: This condition states that one cell can be occupied by one vehicle at a time. It ensures that if two vehicles share a common path, then one cell can be reserved by one vehicle at a time.
- *Hold and wait*: This condition holds true when one vehicle waits for a cell to be free which is currently occupied by another vehicle.
- *Non-preemption*: This condition is met when the cell is released only when the vehicle (V1) occupying that cell has left it completely. It is infeasible to remove the vehicle (V1) occupying the cell until the completion of the assigned order at that particular cell.



(a) Deadlock in computer science, process 1 (P1) has resource 1 (R1) and requires additional resource 2 (R2) which is currently being used by process 2 (P2) [56]



(b) Vehicles in deadlock [11]

Figure 4.2: Deadlock representation in computer science (a) and its analogy in logistics (b)

- *Circular wait*: This condition holds true when a circular chain of vehicles waiting for each other to move to the cell that is currently occupied by the next vehicle in the chain

Coffman says that in order to prevent the occurrence of deadlocks, at least one of the four above-mentioned conditions should be broken. In figure 4.2b, all four conditions are satisfied, thus creating deadlocks. In DTVS in the warehouse, the first three conditions are always true in physical systems. Therefore, in the context of the problem of this study, the breaking of Coffman's fourth condition *Circular wait* will avoid the occurrence of cyclic deadlocks.

4.2.1. Categories of deadlocks:

Type 1: *Active deadlocks*:
A deadlock caused by irrational motion coordination strategies where vehicles are already in a deadlock can be termed an active deadlock. This kind of deadlock can be prevented or resolved by the motion coordination program running on the vehicles, controlling the movements locally, and correcting the passing sequence. This deadlock category is out of scope for the current thesis as the focus is on global planning and co-ordination systems.

Type 2: *Passive deadlocks*:
On the other hand, a deadlock caused by poor path planning is termed a passive deadlock. It can be resolved by a reasonable path-planning strategy that minimizes the forming of deadlocks by deadlock-free routing or re-routing some vehicles. This is the category of deadlocks looked into in this study.

To handle deadlocks, the next section discusses three basic strategies/approaches to address deadlocks in DTVS operation.

4.3. Strategies to handle deadlocks

The deadlock handling strategies are classified into three; deadlock detection and recovery policy, deadlock prevention policy, and deadlock avoidance policy.
Deadlock detection and recovery handles active deadlocks. That means this policy allows the occurrence of deadlocks. These deadlocks are detected and recovered by another algorithm that re-plans the path of at least one vehicle that is in a deadlock situation. This approach does not prevent

deadlocks in advance. While deadlock prevention and avoidance handles passive deadlocks.
Deadlock prevention is an off-line traffic control policy that aims at the complete avoidance of any situation that may lead to a deadlock by pre-planning 100% deadlock-free paths prior to the mission execution. It follows the static approach and computes routes that do not require change and are robust against disturbances.
Deadlock avoidance is an online control policy in real-time that avoids the occurrence of deadlocks in the next event by dynamically allocating the vehicle's mission path based on the information of the current state of the system. This approach is used for dynamic routing. In case of delays and disturbances, this approach then requires re-computation of vehicle routes which can be computationally expensive.

4.3.1. Deadlock detection and recovery policy

This policy handles deadlocks in two steps; detection and recovery. For detection, a system's actual updated state should be represented all the time whenever there is a significant change in the status of the system, here DTVS. The system is represented as graph-theoretic / matrix-based or Petri-net-based. According to [35], this policy is termed as *lazy optimistic strategy* which reserves or schedules a route, if available, hoping that the planned route will be deadlock-free. The strategy uses *resource allocation graph* which is used in computer science, also called *wait-for-graph* to detect deadlocks. The cycle where one vehicle is waiting for a cell to be free is currently occupied by the next vehicle in the chain of the cycle as represented in figure 4.2a. A cycle in wait-for-graph indicates deadlock. Once the cyclic deadlock occurs in the operational layout, it persists until the policy detects and resolves it. The resolution of deadlock is done by reallocating resources, for instance, paths or cells. This policy works successfully for the systems with occasional deadlocks.

A major disadvantage is that this strategy is unable to predict deadlock even after certain information on deadlock occurrence in the future is available in the system. For example, in case a vehicle is broken in between the operations, and some other vehicles are in the operation and will traverse the route or cells where the vehicle broken is currently halted. This strategy does not predict the deadlock situation of the operating vehicles but detects a system deadlock when it actually happens [37]. It is time-consuming and increases the vehicle waiting time to resolve the deadlock.

4.3.2. Deadlock prevention policy

Deadlock prevention policy prevents the occurrence of deadlocks which are passive in nature. With prior knowledge of the system, missions and vehicles, a prevention strategy can provide fine discretization with vehicles moving on edges and a faster network of routes for a large number of vehicles but is computationally expensive. For prevention strategy, it has to prevent one out of four deadlock conditions (4.2) to prevent deadlocks in the operational layout.

Deadlock prevention by system design and path planning:
Transportation system design
By careful transportation system design, the possibility of deadlock occurrence is excluded completely before the mission execution/operation. It is an offline strategy, where the system schedules paths in such a way that is deadlock-free in prior. For bidirectional layout and higher throughput, a prevention strategy with faster network routing is beneficial in terms of cost reduction with a reduction in used areas by vehicles [19].

Static path planning
With static route planning, the factors required for a mission such as vehicles, routes and location

are acquired before the vehicle movement or operation. A path from the source to the target is determined during the routing and scheduling and remains occupied until the vehicle completes the mission. In addition, this path remains the same for that particular set of source and target. This means, that in case of disruptions or irregularities in the system, the static vehicle management fails to modify the path.
In order for faster network routing, static planning can be relaxed so that different combinations of paths can be chosen between the same locations. With mesh or grid-based topology, the path with the shortest distance can be chosen.

Time-window-based path planning
With time-based route planning, certain segments/cells in the route of a vehicle are allocated and the rest of the segments in the route are utilized by other vehicles. In [41], chains of elementary reservations, where vehicles dynamically add the reservation of cells/segments at any time. After adding these reservations to the queue, vehicles can commence their transport missions. If two vehicles share a reservation, then one vehicle waits for another to finish its mission.
In the paper [32], each node in a cell of the operational layout has a list of time windows reserved by vehicles and a list of free time window that is available for the vehicles to reserve. In this way, the algorithm plans the vehicle route with free time windows in the proposed time-window graph instead of physical cell nodes of the operational flow path. The computation time is then $o(v^4n^2)$ where v is the number of vehicles and n is the number of nodes. This means, that with an increase in the number of vehicles, the computational complexity increases. This is suitable for only small transport systems. Article [57] proposes a dynamic routing method by taking into account active missions and their respective priorities for multiple AGVs. The paths are evaluated by time windows and overlapping tests to check the feasibility. As the number of active missions with corresponding vehicles changes with time, the proposed strategy makes the shortest path feasible for missions by time window elongation which results in collision and deadlock-free mission paths. However, the proposed method is limited to undirected graph topology. Factors such as acceleration, deceleration, and external obstacles in dynamic environments make it difficult to calculate time windows precisely in case of delays and can lead to unpredictable obstacles in dynamic environments [74].

Deadlock prevention by eliminating deadlock conditions

- Eliminate Mutual exclusion: Prevention rules that can state no two paths can share a common node at the same time.
- Eliminate Hold and Wait: The Prevention algorithm plans all the mission paths with required nodes and edges *before* the start of the execution of those missions. This leads to low utilization of nodes in the operational layout. For example, if mission 2 requires navigating through a particular node at a later point in time, but the prevention algorithm has assigned that node for another mission before the start of execution, then that node will remain blocked for mission 2 until the other mission has completed its execution. This solution might lead to increased waiting time.
- Eliminate No preemption: Prevention strategy can preempt the node(s) if those node(s) are required by a high-priority mission. The low-priority mission then has to make new requests for the mission path node(s). This can lead to long request queues. In the current study, no priority in the mission is considered.
- Eliminate circular-wait: To eliminate this condition, each node will be assigned a weight/number depending on its priority. The algorithm then only allocates nodes to the mission path depending on the priority weight/number. For example, if the mission 1 path has node 5 assigned, and due to deadlock, it then requests node 4 which has lower priority than node 5 to

navigate through, the algorithm does not make the change as node 4 may already be assigned to another mission path. This can increase the mission execution waiting time. In addition, assigning priority to nodes can be different from mission to mission or application.

Deadlock prevention strategy can eliminate condition 1, however requires additional computation time to eliminate other conditions. In addition, the deadlock prevention strategy performs well for small transportation systems with low utilization of resources but fails to perform efficiently in the case of large systems where traffic dynamics change and disturbances are higher.

4.3.3. Deadlock avoidance policy

Similar to the prevention strategy, deadlock avoidance avoids the occurrence of the deadlocks which are passive in nature. This policy plans the mission operations dynamically depending on the system state such that it remains deadlock-free. The avoidance policy requires that the system is not initially in a deadlock state. This strategy works by knowing deadlock dynamics. Dynamics is defined by *state*. A state is defined by the available and assigned cells/nodes, and by the required cells per mission. A state can be defined as *safe state* if an algorithm is able to assign the required number of cells/nodes for all the missions in any order in a way that does not lead to deadlock. *Unsafe state* is when some nodes/zones, even after careful scheduling, may cause deadlocks. Not necessarily all the time, it may or may not cause deadlocks represented in figure 4.3. Deadlock avoidance techniques consist of a resource allocation graph and Banker's algorithm with a further wait-for approach.

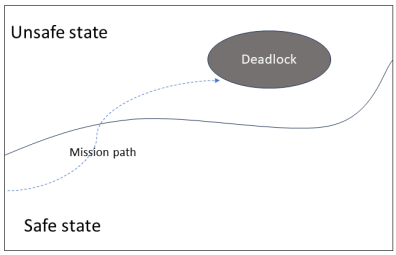


Figure 4.3: Deadlock dynamics State

Resource allocation graph
Most of the deadlock avoidance algorithms use Petri-net-based methods. Modeling a Petri-net or matrix-based is time-consuming. Whenever there is a small modification or change in the physical layout, it takes time to incorporate these changes and revise the model. Revising the model can cause changes system-wide in the existing operational model [72]. To avoid the drawbacks of Petri-net, a graph-theoretic approach is proposed. One of the graph-theoretic approaches, a resource-allocation graph is used to visualize the system's current state. It includes information on all the vehicles assigned with mission routes, routes, and also the routes planned for the vehicles which are yet to operate. It counts for *safe* and *unsafe* states in order to reserve/allocate the cells/nodes for a mission path. Based on this information, the traffic control algorithm decides if the mission path should be reserved or allocated with the next nodes(s)/cell(s) or should wait in order to avoid the deadlock. With fewer missions, the deadlocks in the graph can easily be spotted and avoided. Resource allocation graphs are useful for detecting deadlocks. Dynamic resource allocation can affect the resource/zone utilization and system throughput, and eventually increase the lead time and vehicle travel time [72].

Banker's algorithm
Devised by a famous Dutch scientist E. Dijkstra, Banker's algorithm follows a resource allocation strategy that is motivated by the banker's way of handling loans. Bankers used to ensure that they

lent out resources in the form of loans in a way to satisfy all the customers. Bankers would not loan a little money to start constructing a house unless they are assured that they will later be able to lend out more money in order to complete the house construction [13].

In terms of transportation systems, a Banker's algorithm is a resource allocation and deadlock avoidance strategy that predetermines if the system will remain in *safe state* or not by first simulating the allocation of node(s)/cell(s) to the mission paths. It then makes a *safe state* check before actually allocating the node(s)/cell(s) to the mission in order to avoid deadlock. It is based on tables, unlike resource allocation which is based on graphs. In case the number of missions and nodes is more, Banker's algorithm is useful. A disadvantage of Banker's algorithm is that it fails to store necessary information about the failed executions and the reasons behind the failure that led the mission to an unsafe state. It only knows that a particular mission is in an unsafe state. It also fails to show which mission needs what particular node(s)/cell(s) in order to solve the deadlock.

In the case of a multi-AGV system, a resource is associated with edges/nodes/cells and the process is with missions. Each vehicle requests algorithm allocation of edges on its path from source to target if only that edge is not occupied by another vehicle. Banker's algorithm checks by simulating the request if the allocation will allow other vehicles to complete their missions in *sequential* manner one-by-one. If the new state requested by the vehicle simulates to be *safe*, then the requested edge is granted or else in case of *unsafe* state, the vehicle movement will be forbidden and enters the *wait-for* simulation loop until the state gets to *safe*. This causes low utilization of vehicles and longer waiting times. In addition, simulation and check can be computationally expensive.

Wait-for approach
When an algorithm successfully predicts the deadlocks, it instructs to stop mission execution. The algorithm has to decide if the vehicle has to wait for the deadlock to clear or whether a new plan needs to be drawn. Most of the traditional deadlock avoidance go through closed loops and apply a wait strategy until the deadlock is cleared.
In article [16], the central controller calls deadlock avoidance every time to check if a particular vehicle is allowed to *reserve* a set of cells/tiles required for its navigation. The algorithm determines if the reservation is allowed only if the order of movements of AGVs exists such that the system remains deadlock-free. The combination of steps of different vehicles is checked by the algorithm to evaluate the system state, which causes unnecessary deterring of AGVs and longer calculation time.
A modified Banker's algorithm strategy proposed by [29] provides a solution to solve the low utilization of vehicles and longer waiting time. Under special circumstances, the vehicles are allowed to transverse unsafe states (deadlock existing states) to decrease the vehicle waiting times and mission execution times. This extended set of states is allowed only if all the missions can be executed safely, or else the vehicles wait until the state is safe. Verification of a safe system is graphical, with polynomial complexity associated with a number of missions $O(|M|)$.
An extension of this modification is proposed by [24] for an improved near-optimal deadlock avoidance strategy. It consists of two stages, an offline one that preprocesses guide-path and an online which combines preprocessed guide-path results with the vehicle status to evaluate the safety of the system dynamically. In order to avoid the vehicles waiting until the overlapped edges of two paths are released, alternative shortest paths are obtained. Before a path from alternative paths is allocated to a vehicle, a safety assessment of the system is carried out. If all these alternative paths lead to an unsafe state, the vehicle is instructed to pause and wait for the next scheduled check. This questions the trade-off between travel distance in case of alternative path allotment or waiting time until the system gets back to *safe* state.

This raises questions like; in case of non-re-routing, is it possible that vehicles wait indefinitely? In some scenarios, this is not the best possible solution. The wait strategy can cause persistent cyclic

deadlocks (single and multiple) as vehicles involved in this scene are instructed to wait and rely on each other for movement. An alternate solution is to compare paths and replan or re-route the vehicles. This further raises some questions. If a re-route is initiated, is there always an alternate path available? If two vehicles have the same source point, does one vehicle have to wait for a very long time until the zone is deadlock-free or is it possible to assign a different mission with a different source point? The answer to these questions is to come up with the most suitable strategy that can solve most of these issues.

In case for the system to be deadlock-free, Coffman's fourth condition *Circular wait* should be eliminated. The algorithms for avoidance focus on waiting until a mission path has a safe state. The straightforward way to implement a deadlock avoidance strategy is with vehicle re-routing instead of waiting. Re-routing by path planning also helps to avoid Coffman condition 4 of *Circular wait* by allowing the mission to break from the cyclic deadlock and to re-route and continue operation. In paper [33], authors propose a dynamic routing algorithm that avoids deadlocks efficiently. The dynamic routing calls the Dijkstra algorithm for active path reservation and considers waiting time in the cost, making it possible to choose between waiting until no overlap or detouring. However, the performance of this method is limited to the flow path of AGVs i.e., reserving nodes one-by-one after occupancy and finding the next node before claiming it. In addition, the approach is limited to the layout used in that study. A possible deadlock control approach that is adaptable to the layout, number of vehicles and heterogeneous fleet follows two-step; Step 1 of deadlock detection in the path planned in advance prior to vehicle movement provides a deadlock control strategy with enough time to provoke step 2 and step 2 of avoiding the detected deadlock by avoidance by re-planning the vehicle path. This strategical approach will be further analyzed with multi-criteria analysis.

4.4. Mission Path Planning and Deadlock avoidance

Path planning, being one of the core tasks of the FMS, can be defined as planning the best possible path for a vehicle to traverse from source, re-plan in case of deadlocks and repeating it until the vehicle reaches its target point. Path planning is considered to be a non-deterministic polynomial-time (NP)-hard problem, with an increase in the degree of freedom of the system, increasing the complexity.

4.4.1. Path planning: Classical approach [2]

- Cell decomposition: This method divides the entire operation map/graph into discrete, non-overlapping regions called cells, together forming an entire configuration space called a C-space. The system generates a transverse path through adjacent cells called connectivity-graph and forms a path with a sequence of cells for the vehicle to transverse from source to target.
- Road Maps: The graph produces a set of 1D curves by forming connections between the vehicle's free spaces to path out a road map. The system then uses the set of these standardized curves to find the optimal path. The nodes are used as waypoints for the vehicle to transverse from source to target. In the case of new information, road maps are difficult to regenerate as the whole map now needs to be reconstructed.
- Artificial potential field: An approach where the vehicle is the center of influence and the goal points are considered to emit attractive forces and obstacle points emit repulsive forces to the vehicle in the vicinity of those points. This is local planning where a system forward-simulates the vehicle motion and uses that simulated path as the planned path. With potential fields, however, local minima can cause delays or prevent reaching global minima.

Above mentioned classical methods find feasible paths only with known knowledge about the operating environment. However, these are less capable of handling unknown or partially known dynamic environments, like in the current study and tend to be computationally heavy.

4.4.2. Path planning: Meta-heuristics approach

There are various meta-heuristics methods like Particle Swarm Optimization(PSO), Ant Colony Optimization(ACO), Whale Optimization Algorithm (WOA). These algorithms aim at optimal solutions and reducing computational time.

- PSO: A most common meta-heuristic algorithm for multi-robot path planning. It is based on a stochastic optimization approach by behavioral analysis of swarms. Drawing a balance between exploitation and exploration of swarms, it has both global and local search abilities. It aims at minimizing the mission time, however has shown low scalability and execution ability [39].
- ACO: Inspired by the colony of ants in designing optimal path planning solution, this approach is inspired by the trail leaving ants to find the shortest and collision-free path. The path with higher trails is defined as the optimal path [39][30].
- WOA: One of the bio-inspired swarm-based algorithms that mimic the behaviour and bubble-net hunting of humpback whales. It is a recently introduced algorithm that exhibits superior performance compared to other meta-heuristic methods [53]. It is considered a high-level algorithm for this thesis as of now.

4.4.3. Path planning: Heuristic methods

- Dijkstra algorithm: The algorithm begins by finding the shortest path from the source to the closest nodes, and keeps finding the next closest nodes to the earlier node. It maintains the closest nodes in the priority queue and stores only the intermediate node to form the shortest path from source to target. Modified Dijkstra algorithms are implemented depending on the application. Traditional Dijkstra relies on a greedy search for path planning [2] [39].
- A*: A* search algorithm is one of the graph search algorithms that use the breadth-first search method to find the least cost path. It uses a goodness function that consists of two metrics, one to find the shortest path from the source to the current node and another metric to find the shortest path from the current node to the target node. It is computationally heavy and time expensive when the configuration area is large and if new information flows from the vehicle's on-board sensors then it takes time to restructure [2] the system to introduce and consider new information.
- D*: A dynamic A* called as D* algorithm is basically an extension of A*. It differs from A* in a way where A* computes one path at the start, while D* computes more than one path and hence easier to switch between paths. It also updates the map with dynamic changes if any. It is indeed computationally expensive but efficient as it sticks to the heuristic rule of focusing only on the portions that are relevant to repairing the current found solution path from a given state to target [2].

The focus of this thesis is not on optimizing the path planning algorithm, but just using a simple yet computationally efficient algorithm that finds the collision and deadlock-free shortest path from source to target. Therefore, a path planning heuristic method, i.e., the Dijkstra algorithm which considers these aspects is taken into account as a good primary algorithm approach for the problem at hand. In terms of layout, the objective of the FMS to be more efficient is by increasing system

throughput and decreasing the vehicle waiting time. In order to achieve that, the vehicles travel in bidirectional flow paths [41]. The following section discusses in detail the working of the Dijkstra algorithm.

Proposed by Edsger Dijkstra in 1959 [50], the Dijkstra algorithm is easily implementable and adaptable to topology or configuration space change and dynamic environments.
The configuration graph is divided into nodes and connected to each other by edges. A graph can be undirected, directed or weighted. Each weight can either be a distance or time between the two connected nodes. In this study, the layout flow path is bidirectional.

It is a famous solution for the shortest path problem developed by Dijkstra. It is a greedy algorithm that solves the single-source shortest path problem for weighted graph $G = (V, E, w)$ with non-negative edge weights, i.e., $w(A, B) \geq 0$ for each edge $(A, B) \in E$ with n nodes and e edges and E is set of edges and V is set of nodes. If the edge $<A,B>$ does not exist, the the weight $w(A,B) = \infty$. $d(x)$ is the distance from source node s to node x. Let S denote the set of nodes that are included in the shortest path, which means, initially S contains only source node s. V-S then denotes the set of nodes that are not included in the shortest path yet and follows [51]:

1. Initialization: Set the source node s, and set $S = S \cup s$.
2. In V-S set, find node i that is adjacent to the source node s. If the weight of the connecting edge from s to i is shortest, then add i to set S.
3. Let i be the new intermediate node. Repeat step 2 to find the next smallest numbered adjacent node j from V-S set. Update the distance between source node s and node j. If $d(j) > d(i) + w(i,j)$, which means if the distance passing through i is shorter than not through it, then modify $d(j)$ to $d(j) = d(i) + w(i,j)$, and then add node j to S.
4. Repeat step 2 and 3 for the n-1 iterations, where n is number of nodes in the graph G. The output then consists of the source node, intermediate nodes and target nodes with the shortest path from the source to the target node.

Here, the weight is distributed equally, positive, and is the distance between two connected nodes. It repeats this process and updates the nodes according to the shortest path found until all the nodes in the graph are added. We define the source and the target nodes and the algorithm finds the shortest path between these nodes.

4.5. Multi-criteria analysis for strategies to handle deadlocks

A suitable combination of a path planning strategy with a deadlock handling strategy is required for a Driverless Transport Vehicle System for effective collision and deadlock-free operations.
Both the deadlock handling strategies, deadlock prevention and avoidance are able to prevent or avoid deadlocks successfully, i.e., no occurrence of deadlocks (NoD) in the Driverless Transport Vehicle System with HFoV. To choose a suitable deadlock handling approach, a multi-criteria analysis is used. The idle waiting time (WT) of the vehicles should be reduced in order for increased throughput of missions by avoiding deadlocks and successfully re-routing to avoid cyclic deadlocks (circular-wait) in order to keep operations running. For example, if there are few vehicles operating in the layout and only one vehicle is allowed to operate in order to avoid deadlocks, the system throughput will be affected. On the other hand, applying too many deadlock-avoiding restrictions can make the system to be computationally expensive. The algorithm designed should be restrictive enough to lower the waiting time but also less complex to make it a computationally faster (Comp) working algorithm. In the case of HFoV, this strategy should be adaptable for the system in case new vehicles are added to existing vehicles. In addition, the algorithm should be operationally effi-

cient in the case of scalability (S) of the system. Another factor to consider is the robustness (R) of the system in case of disturbances dynamic changes or dynamic obstacles in the case of HFoV where the heterogeneous fleet consists of vehicles that are dynamic obstacles navigating through their respective missions. The system is robust if it can act and react efficiently against unexpected events by detecting dynamic obstacles (DDO). In case there are mission paths not yet added by the user, the system should ensure a deadlock-free mission path for planned paths. A global optimality should be achieved by the algorithm control policies, but reducing computational expensiveness does not necessarily mean achieving optimality, hence this factor is considered out of scope and left out in this study. The criteria for this study are computational time, waiting time (WT), no deadlock occurrence (NDO), detection of dynamic obstacles (DDO), robustness (R), scalability (S), and complexity (Cplx) of the solution. A rating scale is provided for each of the criteria; −− worst rating and ++ being the most suitable one. The ratings are based on the scientific research conclusions [64] and represented in table 4.1.

	Comp	WT	NDO	DDO	R	S	Cplx
Detection and recovery	- -	-	- -	+	0	0	0
Prevention by design (Zone controlled) [69] [63]	- -	-	++	0	0	-	0
Prevention with static planning	+	0	++	- -	0	-	+
Prevention with time-window based [48]	-	-	++	+	+	0	-
Avoidance with dynamic resource reservation and wait-for approach [74]	0	-	++	+	0	-	0
Avoidance with wait and proceed [45]	-	-	++	++	0	+	0

Table 4.1: Multi-criteria analysis of deadlock handling strategies

The detection and recovery strategy performs well in detecting dynamic obstacles. However, the complexity and computational expensiveness is higher for large-scale systems.
Prevention with design and zone-controlled traffic management approach for larger zones limits the AGV scalability and for smaller zones creates multiple vehicle pauses and longer vehicle waiting times. Prevention with a time-window-based algorithm outperforms static planning and by design approach in terms of detecting dynamic deadlock strategies time-window-based approach and scalability but performs less in areas of waiting time and computational expensiveness compared to static planning. The prevention with time-based planning to avoid dynamic deadlocks has a computational time complexity of $O(N^2H^2)$ with N being the number of AGVs and H being the time window. In addition, the negotiation for deadlock is FIFO (First-In-First-Out) AGVs, i.e., in case of deadlock, the first AGV gets the chance to move and the next AGV has to wait, increasing AGV idle waiting time.
The alternative approach to zone control is dynamic resource reservation, where the central controller assigns nodes (resources) at a time to the vehicles as they proceed. If two vehicles share a common resource, one of the vehicle's status changes from moving to waiting. The controller compares the residual path (path from current to final node) with other AGVs node-by-node. The computational time is related to the number of AGVs i.e., $O(N)$. This strategy resolves dynamic deadlocks and wait-for to falsify circular wait deadlock conditions, but with increase in the number of vehicles

increases vehicle waiting time and decreases the system throughput.
The upgraded version of dynamic resource reservation is avoidance with *wait and proceed* specifically to tackle cyclic deadlock formation in large-scale systems with up to 80 AGVs. In this approach, deadlock prediction cycles are run frequently issuing new commands to AGVs i.e., new control points (nodes) to occupy one at a time which effectively avoids the occurrence of cyclic deadlocks. The prediction algorithm computational complexity depends on the number of vehicles i.e., $O(N^2)$. An important point to note is how frequently the predictions are made. The sooner the prediction, the better avoidance measures can be taken.

To summarize the analysis, the traffic control policy has two steps; conflict detection and conflict resolution. Collision and deadlock detection are static and dynamic and the resolution strategy is waiting or re-route. Dynamic detection is further categorized as node-by-node detection or frequent deadlock prediction cycles to avoid cyclic deadlocks. For resolution, the waiting strategy involves vehicles instructed to wait to avoid deadlock formation. The waiting can be *First In First Out* based, task-priority based etc. The re-route strategy involves vehicle rerouting only if a deadlock is detected. In order to minimize vehicle waiting time and cyclic deadlock, rerouting is a straightforward approach. For detection, instead of node-by-node detection as the vehicle proceeds, the sooner deadlock prediction, the sooner a deadlock handling strategy can be applied, reducing computational complexity with node-by-node check and increasing the system throughput.

A selected strategy is deadlock detection in prior and avoidance with the implementation of re-routing with Dijkstra path planning to resolve cyclic deadlocks and evaluate if this control strategy can increase operational performance and system throughput.

4.6. Key Performance Indicators for DTVS

In order to quantify the study, Key Performance Indicators (KPIs) are used in research to evalute the operational performance of the developed algorithm. Here, KPIs are measurable values to determine the effectiveness of a proposed scientific approach. These are critical to validate the system's performance qualitatively and quantitatively.

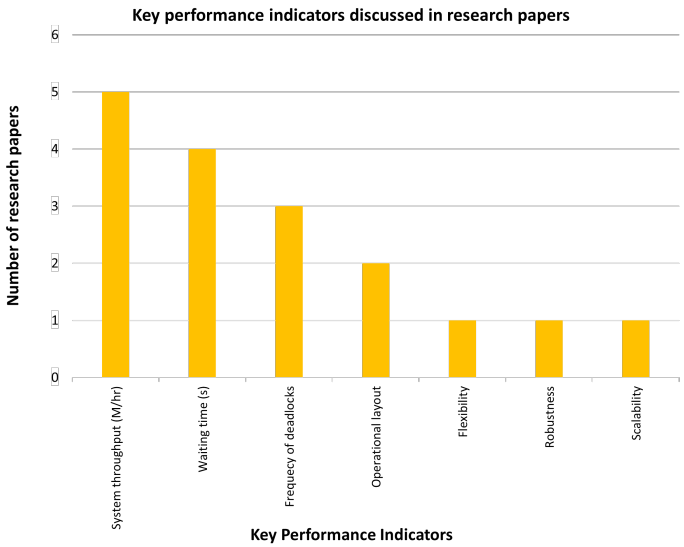


Figure 4.4: Key performance indicators for traffic control strategies used in driverless transport vehicle operations

In an intralogistics industrial environment, time is essence in quantitative measures.

- **System throughput (number of missions completed/hr)**: By employing AGVs and driverless

transport vehicles, the distribution centres, ports and large material handling facilities like Amazon warehouses are reducing the costs and increasing the throughput. However, it is necessary for these infrastructures to invest in safe and reliable vehicle operations, i.e., collision and deadlock-free operations to reduce fatal risks [43]. For a detailed quantitative analysis, a subject of ongoing research is a detailed study on safe vehicle operations, vehicle density and system throughput with different traffic management approaches. According to [15], to improve the operational performance of the system, evaluating factors are enhancing algorithm performance and system throughput. In order the improve the system throughput, optimizing the operational algorithms in multi-AGV systems by feasible traffic coordination is the key [9]. With well co-ordination traffic control strategy, the vehicles travel on assigned paths by the algorithm in order to reduce the travelling time and in turn increase the number of missions completed [3]. Not only for planning but also for completing the missions along with heterogeneous vehicles in size and functionality with developed algorithm (Degene) compared to the traditional algorithm, computational time taken to complete is considered important [70]. The average execution time taken by the number of vehicles to accomplish the missions is considered a performance indicator [4] [41].

- **Waiting time (s)**: Another performance indicator is the waiting time, the time taken for the algorithm to route and schedule the optimal deadlock-free mission path before the mission dispatch to the vehicle. Waiting time is the sum of mission routing and scheduling. For the operational performance check, the vehicles *idle* state is kept as minimum as possible [40] [21] [7] [61] [14].
- **Computational time (s)**: Computational time is the total time taken by the system to successfully complete the defined number of missions. Evidently, the performance of algorithms in an industrial setting is indicated by the measurable quantity and computational time. In the current study, the system throughput plays a vital role both operationally and economically for an industry. Hence, the computational time of the algorithms to solve a complex problem is not a prime key performance indicator in this study.
- **Frequency of deadlocks prevented and avoided**: Another performance indicator is the number of deadlocks prevented and avoided. The occurrence of a number of deadlocks strategically prevented and avoided by different algorithms in the operational time frame to complete all the missions assigned should be studied. This shows the ability of the algorithm to find a new path for the vehicles in order to reduce the total vehicle waiting time [64] [68] [34].

Other performance indicators for the algorithm are to be flexible to easily adapt to different system layouts, robust to be capable of adapting to interruptions during operations and scalable in case of an increase in vehicle or traffic density as discussed in [17].
In this study, the system throughput, and waiting time of vehicles by different algorithms are compared by input parameters number of vehicles (size and functionality) and simulation time. In order to check the operation performance of the warehouse, [49] suggests checking the performance of the algorithms with respect to changing numbers of vehicles, which eventually affects the total waiting and traveling time.

4.7. Summary

According to Coffman, the system is in a deadlock if any of the four conditions come true. The conditions are *Mutual exclusion*, *Hold and wait*, *Non-preemption* and *Circular wait*. In a physical system, the first 3 conditions always come true and with simple rules can be avoided with path planning algorithms. The fourth condition *circular wait* in physical systems causes vehicles to halt

the operation for longer, which increases travel time and decreases system throughput. Hence, this study focuses on developing the deadlock handling algorithm to avoid circular wait as quickly as possible. Deadlock handling strategies are categorized into deadlock detection and resolution, deadlock prevention and deadlock avoidance. *Deadlock detection and resolution* allows deadlocks to occur in order to detect them and then acts to resolve them. It is called *lazy optimistic strategy* which plans route *hoping* that the planned route is deadlock-free. It is time-consuming and increases vehicle waiting time to resolve deadlock.

Deadlock prevention has prior knowledge of the system, missions and vehicles and uses this information to prevent deadlocks. It follows static planning of the routes and fails to prevent deadlock in case of dynamic obstacles. For dynamic obstacle detection, prevention with time-window-based planning and zone control strategy is designed by researchers. With zone control, each vehicle is assigned a fixed or dynamic zone and it is constrained to operate in an allocated zone. Large zone limits the scalability of the system vehicles, and smaller zones cause multiple mission pauses and longer vehicle waiting time to resolve deadlocks and conflicts. Prevention with time-window-based planning avoids dynamic deadlocks with a computational complexity of $O(N^2H^2)$, an increase in the number of vehicles causing increased computation complexity.

To reduce computational complexity and avoid zone allocation, dynamic resource reservation and upgraded version with *wait and proceed* are coined by researchers. In the former, the central controller reserves resources (or nodes) only if it does not lead to deadlock. The vehicles are assigned nodes as they proceed depending on FIFO or mission priority basis. The *winning* vehicle moves while the *loser* vehicle changes its status from moving to waiting. The computational complexity is $O(N)$, and due to the waiting strategy, an increase in the number of vehicles creates an unresolved cyclic deadlock situation. To avoid cyclic deadlock, the latter strategy of *wait and proceed* follows a deadlock prediction cyclic frequency and resolves cyclic deadlocks by waiting strategy. The computational complexity of this approach is $O(N^2)$.

Room for improvement is to reduce computational complexity and replace the waiting strategy with straight forward re-route strategy if deadlock is detected. The prediction of deadlock plays a vital role in controlling cyclic deadlocks and the calling of resolution strategy can be accelerated quickly to re-plan the mission routes aiming for higher system throughput. This deadlock control approach will be modeled in the next chapter and evaluated for its operational performance.

5

Traffic Management Module System Modeling

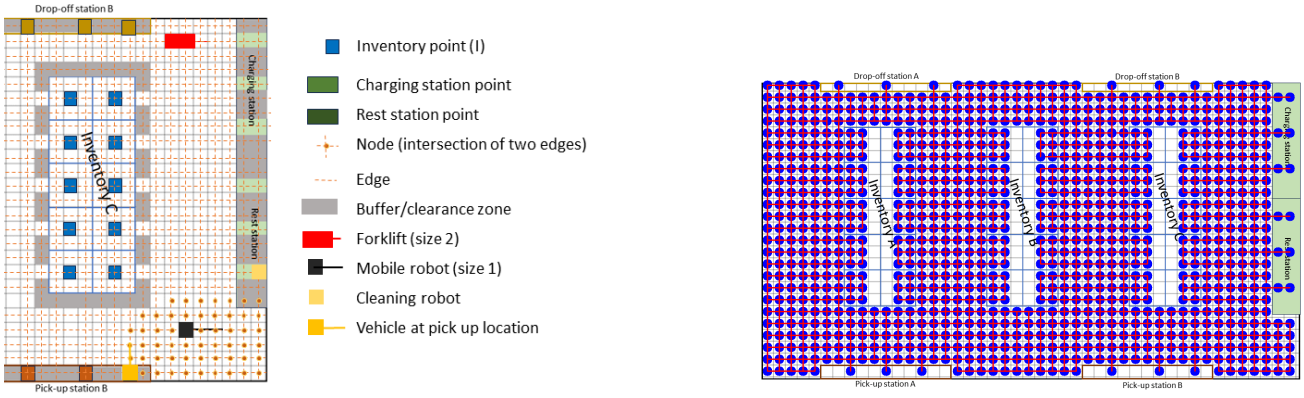
This chapter discusses modeling and developing the novel deadlock control algorithm for the heterogeneous fleet of AGVs in collaborative warehouse operation. First, the front-end modeling to model the operational layout, model for simulation and GUI, and next the back-end modeling that holds developing the strategy and implementing with VDA5050. A deadlock control strategy coined in the previous chapter is modeled and developed in this chapter. The strategy first detects deadlock by developing four check conditions and resolves the deadlock by replanning the mission paths. This strategy achieves the operational research objective drawn in 1.2.1. To achieve the overall objective of this project, technical and operational are brought together later in this chapter, i.e., the operational objective developed is communicated to the vehicles via a communication server with the technical objective of using an industrial standard protocol of VDA5050. The flow of mission and vehicle status with VDA5050 order messages is presented briefly. To simulate the strategy output, a simulation model is built and an input GUI is developed for users with different input options. This chapter answers SRQ 4 about modeling and developing the novel deadlock control strategy.

5.1. Front-end modeling

Before developing the algorithm, the operational environment and components required for the input to the algorithm are to be defined.

5.1.1. Operational layout

The warehouse layout is divided into an operational guide/mission paths connected to via a network of nodes and edges for the vehicles represented in figure 5.1. Each node has an x and y coordinates. These nodes with coordinates are input to the algorithm environment as a .csv file.



(a) Bidirectional layout with nodes and edges

(b) Network of bidirectional layout

Figure 5.1: Bidirectional networked layout used in this study

The transportation area is divided into a tessellated layout with non-overlapping zones called cells. The center of each square-shaped cell is called a node and nodes are connected to each other with

edges. Each cell is connected via edges.
The layout can be a single loop, tandem or conventional as explained earlier in chapter 4. In a single-loop layout, vehicles travel in a unidirectional path, which is computationally less expensive but however not flexible. In this layout, the vehicle has to travel in loops to visit the same node. In case of a vehicle breakdown, that vehicle can block the whole system. Similar to a single loop, the tandem layout has non-overlapping cells with single vehicles operating in tandem but has low-fault tolerance and causes system breakdown similar to a single loop. Hence, to overcome the disadvantages of single-loop and tandem layouts, a bidirectional layout is followed. In this layout, the vehicles travel bi-directionally, which is advantageous to the DTVS operation because vehicles are allowed to take shortcuts in order to reduce travel distance and time. These shortcuts can lead to potential deadlocks and can be resolved with a deadlock control strategy.

5.1.2. Simulation model

To simulate the developed deadlock control strategy for a DTVS with a fleet of heterogeneous AGVs, a simulation model is designed. The simulation model is not only to verify the working of the developed strategy but also to validate the operational performance of the developed DTVS for HFoV.

The layout is divided into 45 X 25 nodes as shown in figure 5.1a. Blue cells are the inventory points, green coded are charging and rest station points. The cells represented in grey are buffers or clearance zones for the vehicle to allow enough space for turns. To introduce heterogeneity in vehicles, three different kinds of vehicles with different sizes, functionalities and manufacturers are introduced. Size 1 occupies one cell at a time, size 2 occupies 2 cells while traversing in a straight line and three in case of turn in order to represent the occupied cells during turn. The tail behind these vehicles is to keep track of the vehicle's movement, also the vehicle's forward movement is in the direction of the head. For heterogeneous functionality and posing as a dynamic obstacle, a cleaning robot in yellow is modeled. The system consists of 3 inventories each with 10 inventory points, 6 pick-up and 6 drop-off locations, 3 charging and 3 rest stations. A topology description of the simulation model is discussed in 5.1a.

- Definition 1 - Charging station point
  These points are defined as nodes designated for charging vehicles in charging stations. If the vehicles reach 20% battery level in between operations, the vehicles are instructed to go to these points and start charging. In this study, it is assumed that vehicles remain fully charged throughout the operation with 100% battery level. In this project, this is the start and end location of the vehicles after the completion of all missions. These are also assumed as rest stations for vehicles in case the rest station is completely occupied.
- Definition 2: Rest station point
  Similar to charging station points, these points in the rest station are designated as the start and end points of the vehicles.
- Definition 3: Source
  Every mission path has a source point. The start node of the path/mission is the source point.
- Definition 4: Target
  Every mission path has a target point. The end node of the path/mission is the target point.
- Definition 5: Inventory point
  Inventory points are the location of the inventory orders in inventory A, B and C.

- State of vehicle
  The vehicle state is either defined as alive (ready for mission execution) or idle (idle after mission completion). These status codes are followed from the VDA5050 manual.
- State of mission
  Mission states are pending, running and completed. Pending is when the mission is awaiting the vehicle allocation, running is operational status and completed in case of mission completion. These status codes are followed from the VDA5050 manual.

The novel deadlock control strategy can be applied to an industry with existing DTVS with a heterogeneous fleet of operations where deadlock can occur by the interaction of two heterogeneous vehicles unknown to each other with the help of VDA5050. For example, a healthcare system where a cleaning robot and service robot to deliver goods are required in the operational layout. In ports or container terminals, two automated container transport vehicles from two different manufacturers work in a shared operational environment. In warehousing or airport baggage handling, automated mobile robots can operate together with automated forklifts in a shared common operational layout.

5.1.3. Graphical User Interface

GUI is designed to pass a set of instructions from the user, i.e., input to the FMS (also called as central controller) and log operational updates on the window. It is divided into three sub-windows, the left window has different tabs for input, the middle window shows the operational layout and vehicle movement during the mission run and the right window shows mission/vehicle/operation states and updates as represented in Figure 5.2.
Tabs on the left window are divided into the selection tab and the output tab. The selection tab consists of algorithm selection, vehicle selection, and input number of missions. The output tab consists of computational time to complete the number of missions input, the cumulative waiting time of the vehicle until all the missions are complete, the deadlocks tracking tab and the progress tab to keep track of the number of missions completed.
The middle window is the configuration space layout. To visualize the mission path, and the vehicle operations if they are working as instructed. It helps in implementing required changes by modifying code for smoother operations. The layout is user-defined input to the FMS from Data Management module.
The right window showcases mission updates as shown. Pick-up mission from allocated node and drop-off to the target node. It updates vehicles available, running missions and completed missions.

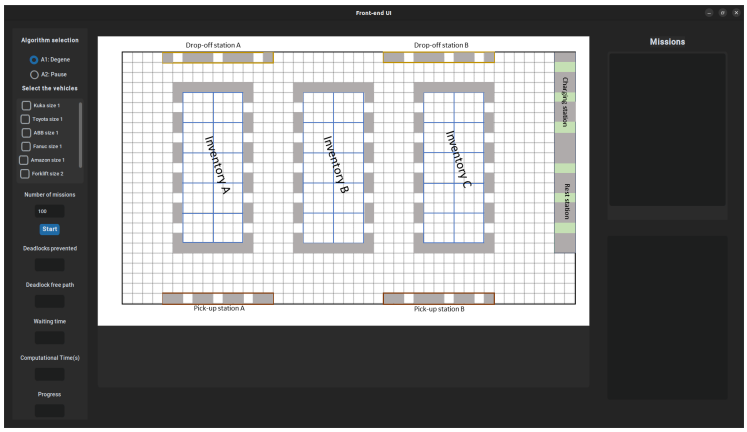


Figure 5.2: Front-end GUI for the Intelligent Transport System designed

5.2. Back-end modeling - Deadlock detection and avoidance algorithm

After defining the necessary resources and operational environment to input into the algorithm, we are now ready to model and develop the deadlock control strategy. A pseudo explanation of the working of the developed algorithm is presented in 5.2.

1. Input of data for simulation. It consists of the number and choice of vehicles to the virtual operational environment, the required operational/simulation time, and the pick-up and drop-off locations of these missions. In this study, a random mission pick-up location and drop-off are initialized in order to check the algorithm performance with changing pick-up and drop-off locations in every run.
2. Each mission is divided into two paths; first from vehicle location to pick-up node and then pick-up to drop-off node. For these paths, a source and a target node are defined. The first path is planned with Dijkstra between the first idle vehicle's current location and the pick-up location. The second path is planned also with Dijkstra by finding the shortest route from pick-up to drop-off location.
3. Once the mission is executed, it is stored as the existing path. While the missions that are planned but yet to be executed are named as new paths.
4. Four check conditions are defined. The existing path is checked with new paths for four conditions to avoid deadlocks. These conditions are explained in detail in later sections. In case these two paths have deadlock(s), the new path is instructed to be re-routed by calling Dijkstra function.
5. In order to avoid the cyclic deadlock by assigning the same deadlock(s) node(s) to the new re-route path, the check conditions make sure to remove the edges between deadlock nodes in order to make them unavailable. This prompts the Dijkstra to find a new shorted route to the new path.
6. Once the checked new path is executed, it becomes the existing path
7. The processes 4 to 5 continue until all the missions are executed with deadlock-free paths.

There are four check conditions which check for deadlocks between two missions, detect them and avoid them by re-routing yet-to-be-executed missions. These four conditions; *CheckSingleMultiNodeCollision*, *CheckCommonEdgesInForwardRoute*, *CheckCommonEdgesInReverseRoute*, and *CheckFinalNode* are explained below.

Algorithm 1 Check All Conditions

```
Function CheckAllCondition(new mission path, running mission path)
    ep ← collect running mission path;
    for path ← ep do
        C1 ← CheckSingleMultiNodeCollision;
        C2 ← CheckCommonEdgesInForwardRoute;
        C3 ← CheckCommonEdgesInReverseRoute;
        C4 ← CheckFinalNode;
        if C1 or C2 or C3 or C4 then
            return TRUE;
        return FALSE
```

5.2.1. Check condition 1 - Check for single or multi-node collision and deadlock

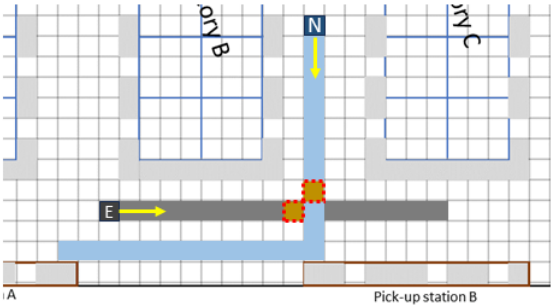


Figure 5.3: Single/Multi-node collision, detection and avoidance of deadlock

In figure 5.3, a black vehicle with a grey path is the existing mission executed and a blue with a blue path is the new mission path planned and yet to be executed. The algorithms check these mission paths for deadlocks as represented in 2. These paths have a deadlock highlighted in brown region with red marking. In order to avoid this single-node deadlock, the blue vehicle is then instructed to re-route.

Algorithm 2 C1 - Check single or multi-collision nodes in between the path and avoid deadlock(s)

Function *C1 (new mission path, existing path, new mission path weights, existing path weights)*
- tolerance ← vehicle size;
- **for** *i* ← *len(min(new mission path,existing path))* **do**
  - n1 ← new mission path(i);
  - e1 ← existing path(i);
  - nw1 ← new mission path weights(i);
  - ew1 ← existing path weights(i);
  - **if** *n1 == e1 & abs(nw1 - ew1) < tolerance* **then**
    - Remove edge (n1, n1-1)
  - **if** *n1 == e1+1 & abs(nw1 - ew1+1) < tolerance* **then**
    - Remove edge (n1, n1-1)
  - **if** *n1 == e1-1 & abs(nw1 - ew1-1) < tolerance* **then**
    - Remove edge (n1, n1-1)
- **If** any edges removed, return TRUE, **else** FALSE

In order to check if the node pair (that is node from the new mission path and the existing mission) are the same and that the vehicle reaches this node at the same time by checking the cumulative weights. This check can avoid collision and deadlock caused at that node.
Tolerance is defined by the size of the vehicles used in the operational layout to check if these vehicles collide when working in proximity. A node in the new mission path and existing mission path is checked if it is the same. If same, it is checked if an absolute difference of cumulative weights of the new mission path and the existing mission path is less than tolerance. If yes, then the current same edge is removed. Similarly, if the node in the new mission path is the same as the previous node of the existing path or the next node of the existing path, then the absolute difference of cumulative weights of the new mission path and the existing mission path is checked. If the difference is less than tolerance, then the current edge is removed and Boolean value *TRUE* is returned. In order to keep a safe buffer zone between two vehicles, the node of the new mission path is checked with the previous and next node of the existing path. This is not necessary but however, recommended.

5.2.2. Check condition 2 - Check for overlapping path segments for vehicles traveling in the same direction and avoid deadlock(s)

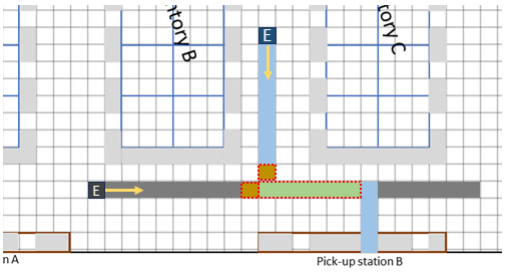


Figure 5.4: Overlapping edges in the same direction traveling vehicles, detection and avoidance of deadlock

In figure 5.4, the black vehicle with the grey path is the existing mission executed and the blue with the blue path is the new mission path planned and yet to be executed. The algorithms check these mission paths for deadlocks as represented in 3. These paths have the longest common sub-sequence and are traveling in the same direction to reach their targets. Check condition 2 is invoked. In order to avoid deadlock, blue is instructed to re-route.

Algorithm 3 C2 - Check common overlapping edges for both the vehicles traveling in the same direction

Function *C2 (new mission path, existing path, new mission path weights, existing path weights)*
- lcs = list(longest common sequence(new mission path, existing path));
- **for** *cl* ← *lcs* **do**
  - **if** *len(cl)* < *2* **then**
    - Skip
  - IndexNew = Get index of cl in new mission path;
  - IndexEp = Get index of cl in existing path;
  - indexNewWeights = new mission path weights(IndexNew);
  - indexEpWeights = existing path weights(IndexEp);
  - **if** *indexNewWeights overlaps indexEpWeights* **then**
    - Remove edges(new mission path(IndexNew))
- **If** any edges removed, return TRUE, **else** FALSE

If the paths have the longest common sub-sequence and the cumulative weight is the same, then the vehicles are traveling in the same direction and arrive at those nodes at the same time which can cause deadlock during their arrival.
First, the longest common sub-sequence (lcs) is taken for both the existing and new mission path, in order to check if two vehicles are traveling in the same direction. If the length of the longest common sub-sequence (lcs) is less than 2, that means it is a single node check, then we ignore this check because check 1 takes care of this. Otherwise, the cumulative weights from the new mission path and existing mission path are checked. If there is an overlap between the cumulative weight of the existing and new mission path, that means the vehicles visit those overlapping nodes at the same time which might cause deadlock just before their arrival at that node. In this case, we remove those overlapping edges and return a boolean value *TRUE*.

5.2.3. Check condition 3 - Check for overlapping path segments for vehicles traveling in opposite directions and avoid deadlock(s)

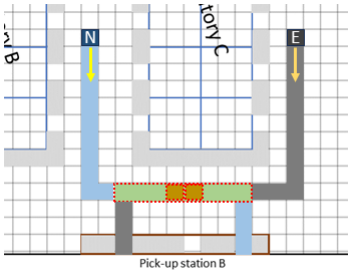


Figure 5.5: Overlapping edges in opposite direction traveling vehicles, detection and avoidance of deadlock

In figure 5.5, the black vehicle with the grey path is the existing mission executed and the blue with the blue path is the new mission path planned and yet to be executed. The algorithms check these mission paths for deadlocks as represented in 4. These paths have the longest common subsequence if checked in reverse for one path and are traveling in opposite directions to reach their targets. Check condition 3 is invoked. In order to avoid deadlock, blue is instructed to re-route.

Algorithm 4 C3 - Check common overlapping edges for both the vehicles traveling in opposite directions

Function *C3 (new mission path, existing path, new mission path weights, existing path weights)*
- reverse lcs = list(longest common sequence(new mission path(reverse), existing path));
- **for** *cl* ← *lcs* **do**
  - **if** *len(cl)* < *2* **then**
    - Skip
  - IndexNew = Get index of cl(reverse) in new mission path;
  - IndexEp = Get index of cl in existing path;
  - indexNewWeights = new mission path weights(IndexNew);
  - indexEpWeights = existing path weights(IndexEp);
  - **if** *indexNewWeights overlaps indexEpWeights* **then**
    - Remove edges(new mission path(IndexNew))
- **If** any edges removed, return TRUE, **Else** FALSE

If the paths have the longest common sub-sequence in reverse and the cumulative weight is the same, then the vehicles are traveling in opposite directions and arrive at those nodes at the same time which can cause deadlock during their arrival.
First, a reverse longest common sub-sequence (lcs) is taken for the existing and longest common sub-sequence (lcs) for the new mission path, in order to check if two vehicles are traveling in the opposite direction. If the length of the longest common sub-sequence (lcs) is less than 2, that means it is a single node check, then we ignore this check because check condition 1 takes care of this. Otherwise, the cumulative weights for the new mission path and the existing mission path are checked. If there is an overlap between the cumulative weight of the existing and new mission path, that means the vehicles visit those overlapping nodes at the same time and might cause deadlock just before their arrival at that node. In this case, we remove those overlapping edges and return a Boolean value *TRUE*

5.2.4. Check condition 4 - For unbalanced weights, the last node check is necessary to avoid deadlocks at drop-off points

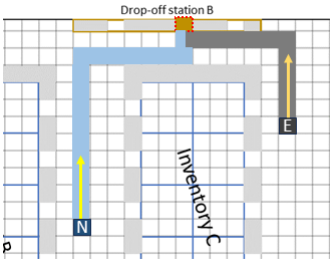


Figure 5.6: Check for unbalanced weight layouts and nodes for deadlock and collision avoidance

In figure 5.6, a black vehicle with a grey path is stored as an existing mission executed, and a blue with a blue path is stored as a new mission path planned and yet to be executed. This mission path information is collected by resource allocation graph and passed to Banker's algorithm to check for *s-state*. The Banker's algorithm checks condition 2 as represented in 5. These paths have a common final node with the same cumulative weights. In order to avoid deadlock, blue is instructed to re-route

Algorithm 5 C4 - In case of unbalanced weights layout, check the final node of two operational vehicle paths

Function *C4 (new mission path, existing path, new mission path weights, existing path weights)*
    tolerance ← vehicle size;
    if *(new mission path(last node) == existing path(last node)) And abs(new mission path weights − existing path weights < tolerance)* **then**
        n1 ← new mission path(i);
        e1 ← existing path(i);
        for *i ← len(min(new mission path,existing path))* **do**
            Remove edge (n1, n1-1)
    If any edges removed, return TRUE, **else** FALSE

In the case of layouts with unbalanced weights (unequal weights) between nodes, the last node of the existing and new mission paths is checked. This is specifically modeled to provide a safe buffer between vehicles, where one is entering the station point and one is leaving the station point. Tolerance is defined by the size of the vehicles used in the operational layout to check if these vehicles collide when working in proximity. If the last node of the existing path and the new mission path are the same, then check if the absolute difference of cumulative weight of the new path and the existing mission path is within the tolerance. If yes, then the new path edges are removed and the Boolean value returned *TRUE*

After the check of all these conditions, a boolean value is returned to check if any edges from any of these conditions are removed. If the value is *TRUE*, a re-route is planned to the new mission path. These conditions are checked until a path with no deadlocks is planned.

5.3. Summary

First, the environment and resources necessary to input in the algorithm are defined. The operational is a bidirectional layout with a network of nodes and edges. The mission paths are a network of these

nodes. The operational layout is represented as a simulation model and user input and visualization of this model is on developed GUI.

The operational objective of this project, modeling and developing a novel deadlock control algorithm follows two steps; deadlock detection in *advance* as a necessary change to the existing works of literature and deadlock resolution by replanning the mission path. For the detection of deadlocks or prediction, four check conditions are developed in space and time by checking the nodes in the path and the time of arrival at the nodes by cumulative weight check. Each executing mission path is checked with new planned paths (yet to be executed) by four check conditions to detect if the paths have any occurrence of deadlocks. Check Condition 1 to detect deadlock between mission vehicles if they share a single node and arrive at the same time. Check Condition 2 to detect deadlock between mission vehicles if they travel in the same direction share multiple nodes and arrive at the same time. Check Condition 3 to detect head-on deadlock between mission vehicles if they travel in the opposite direction share multiple nodes and arrive at the same time. Check Condition 4 to detect deadlock just outside the pick-up/drop-off/inventory station between two mission vehicles where one is leaving the pick-up/drop-off/inventory station and the other is entering that station. These check conditions return a boolean value; *TRUE* if deadlock detected or else *FALSE*. If *TRUE*, a mission path replanning with the Dijkstra algorithm is called to re-route the new planned path with the next shortest route in order to avoid the deadlock. If *FALSE*, the mission follows the initially planned path and no replanning is initiated.

6

Implementation, Experiments and Results

The purpose of this chapter is to evaluate the operational performance of the developed deadlock control algorithm and answer final SRQ 5. For evaluation, the algorithm is first implemented, followed by verification and validation, simulation experiments are conducted and results are analyzed.

6.1. Implementation

Implementation of the modified deadlock avoidance algorithm is carried out by Python programming with IDE Visual Studio Code.

6.1.1. Technical specifications

Python version	3.8
Ubuntu version	Ubuntu 22.04.2 LTS
ROS version	ROS 2
Processor	AMD@ Ryzen 5 4600hs
RAM	8.0 GiB
Graphics model	Radeon Graphics X 12
Graphics card	NVIDIA GeForce GTX

Table 6.1: PC specifications

6.1.2. Mission route network and behavior tree

In this study, a Python package called NetworkX is used for the analysis of complex network graphs [22]. The NetworkX functionality is based on an understanding of graphs. A graph is a mathematical structure used to model processes in different domains like physical, biological and information systems. The entities of the system are called nodes and these entities are connected by edges. These graphical networks have a function to navigate through nodes and edges and working with these graphs helps to optimize paths in complex networks. As the aim of this study does not focus on path optimization, this tool is used to study the structure of the graph and used as a standard programming interface and building a graphical network for this study application. With its ability to operate on large graphs with around 100 million nodes and connected edges, NetworkX with the Dijkstra algorithm for path planning complements the developed deadlock control algorithm.

The basic structure of a mission with orders is a network graph of nodes and edges. The vehicle is instructed to transverse the nodes and edges in the instruction to complete a mission. The full configuration space with the networked graph is imported by FMS from the Data Management Module.
A mission is a series of tasks to be completed by an assigned vehicle. This mission tree with a list of task nodes is represented as a behavior tree. For vehicles to react to all sorts of situations, it is advantageous to use a mission tree instead of an array of task nodes with steps. Here, each mission

tree has a higher/start parent node termed as *root* and children node.
This study follows the mission behavior tree from [27] as a reference. Each mission tree node has four states: IDLE, RUNNING, SUCCESS and FAILURE. There are four mission tree nodes: route, action, sequence and selector. Nodes of the behavior tree are mentioned and explained below in 6.1.2.

- *name*: Each node is assigned a unique name in a mission .JSON format. If not specified, it is automatically set a name.
- *parent*: The parent of the node. If not specified, then it is *root* node.
- *sequence*: In charge of executing children nodes in order. If all child nodes are complete with SUCCESS, then the sequence node changes state to SUCCESS. If a child node is running and completes with SUCCESS, then the next node is started, or else the sequence node states FAILURE.
- *selector*: In charge of executing children nodes in order. If all child nodes are complete with SUCCESS, then the selector node changes state to SUCCESS. If a child node is running and completes with SUCCESS, then the next node is started, or else the selector node states FAILURE.
- *action*: Performs generic and named action on vehicle to execute.
- route: Instructions from FMS that said vehicle to transverse the given path (waypoints). Once the final waypoint is executed, it returns either SUCCESS or else FAILURE

The description of sequence and selector node sounds the same, however, a sequence node attempts to run all the child nodes as long as they return SUCCESS, and will instantly return FAILURE if one of the child nodes fails. In the case of a selector, it will attempt to get only one SUCCESS and upon failure, it will push child nodes to either get SUCCESS or exhausts all child node. For example, if there are ten nodes, in case the fifth node fails, the sequence node will ignore the rest of the nodes and will return FAILURE for the complete branch. But in the case of a selector, it will still run the rest of the nodes for SUCCESS until those nodes are exhausted. A note that all mission tree nodes are mutually exclusive i.e., a selector node can not also be a sequence node and a route node can not also be an action node.

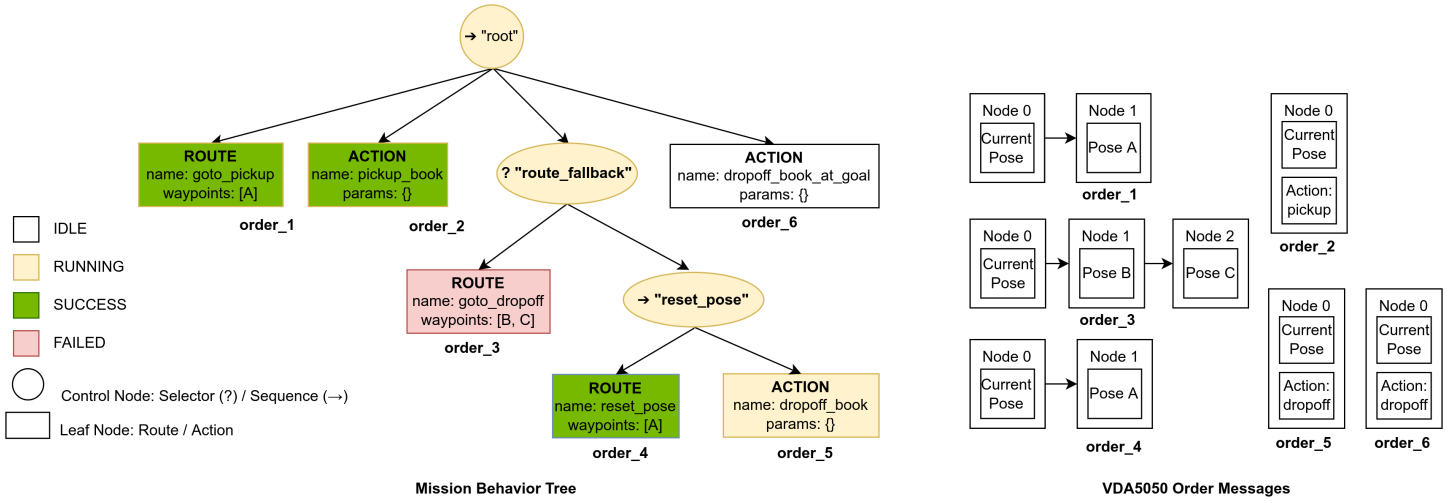


Figure 6.1: Working of mission behavior tree [27]

An example is presented in figure 6.1. On the left is a representation of the behavior tree and on the

right are corresponding VDA5050 order messages. For each route or action sent by FMS, the mission tree node is translated into separate VDA5050 order messages. VDA5050 order JSON schemas are represented in appendix 9. *waypoints* mentioned in *route* node is an order node. These are appended together to form an order message for the path planned. In the action mission tree node, the action to be executed is attached to the first node of order, the one with the current pose of the vehicle. Based on the progression of the behavior tree, the order messages are sent sequentially [27].

For example in figure 6.1, a vehicle is asked to travel to the pick-up location and execute the action of picking up the book. After the successful execution, the vehicle jumps to selector node $route fallback$, where if the vehicle fails to go to a drop-off location, then it is asked to go back to the pick-up point and drop off the book there, or else it drops off the book at target/goal location.

Once the mission paths are input and checked with all the check conditions for deadlock control, the deadlock-free mission is converted into a .JSON file. The mission .JSON file consists of the name of the vehicle assigned, the mission consisting of transverse nodes each with a unique name, a parent, a route with x and y coordinates, actions if any, timeout, and deadline as shown in the example figure 6.2. Here, *Amazon* is transversing from node 150 to 103, with the waypoints. The other waypoints are not mentioned in this example .JSON file.


```
{
  "robot": "Amazon",
  "mission_tree": [
    {
      "name": "Service_robot_150_node_103_0",
      "parent": "root",
      "route": {
        "waypoints": [
          {
            "x": 0.0866,
            "y": 0.0407,
            "theta": 0,
            "map_id": "node_103"
          }
        ]
      }
    },
    {
      "name": "pickup_order",
      "parent": "root",
      "action": {
        "action_type": "dummy_action",
        "action_parameters": {
          "should_fail": 0,
          "time": 1
        }
      }
    }
  ],
  "timeout": 100,
  "deadline": "2023-09-01 13:22:49.704036",
  "needs_canceled": false,
  "name": "Service_mission_id_Amazon_699f010d-1bf5-1_chunk_0"
}
```


Figure 6.2: Mission .JSON format for vehicle *Amazon*

Input parameters for simulation experiments:

- Algorithm selection
- Vehicle density: Number of vehicles
- Vehicle size and functionality
- Simulation time

6.2. Verification and Validation

For model verification and validation, there are various techniques and tests used. In this study, a documentation of the tests and techniques applicable for this study are derived from [54].

6.2.1. Verification and Validation Techniques

- *Animation*: The operational behavior of the model is represented graphically as it moves through time. Here, the visualization of the model output acts as *animation* verification and validation technique.
- *Face Validity*: If the model and its behavior are reasonable, several logic questions are posed if the conceptual model is correct and if the input-output relationships are reasonable. Here, input-output behavior questions if the increase in the number of vehicles causes an increase in business in the layout and does that affect the system throughput.
- *Operational Graphics*: Visual display of performance indicators and their dynamic behavior with time. For instance, the dynamic behavior of performance indicators like waiting time and system throughput with dynamic changes in the number of vehicles, size, and functionality as the simulation model runs through a given simulation time to ensure the right behavior of the system.
- *Parameter Variability - Sensitivity Analysis*: As the name says, this technique employs changes in the input parameters to determine its effect on the behavior and output of the model. The same input-output relation should occur in reality. The sensitive parameters that cause significant changes in the output behavior of the model are to be set sufficiently accurately before the simulation runs of the model. For instance, the same set of missions to the respective vehicle in all the simulations run in different models.

As this study focuses on the operational behavior of the system, *operational validity* is conducted for this project. The purpose of operational validity is to determine if the output behavior of the simulation model has the accuracy required for the intended purpose of the model across the intended applicable domain. Here, output behavior is the simulation throughput with an increase in the number of AGVs, sizes, and functionality and the model's intended purpose is to avoid deadlocks efficiently to reduce travel costs and increase system throughput in heterogeneous warehouse environments. The techniques discussed in 6.2.1 are used for operational validation. These techniques can be used subjectively and objectively.

	Observable System
Subjective Approach	1. Comparison Using Graphical Displays 2. Explore Model behavior
Objective Approach	Comparison Using Statistical Tests and Procedures

Table 6.2: Operational validation classification

The term *comparison* compares the simulation model output to another model output using graphical displays in case of subjective and statistical tests like confidence interval in case of an objective approach. For *explore model behavior*, examination of the output behavior of the model is carried out using techniques mentioned in 6.2.1. For a high degree of confidence in the simulation model and its results, several different sets of experimental conditions are required to compare the output behavior of different models. For this, an observable system that generates data to compare is used. In this study, the developed model is validated with the other model subjectively with graphical

displays and model behavior as mentioned in table 6.2. In addition, a performance validation of the simulation is conducted to validate the developed model.

6.2.2. Verification

Three case studies with HFoV operations are considered to verify the working of the developed algorithm. The operational layout is bi-directional and square-base and rectangular-base AGVs are used. The term 'base' refers to the shape of the vehicle base. Square bases are shown in figure 6.3a and rectangular bases are shown in figure 6.3b.

(a) Square-based **DTV!** **(b)** Rectangular-based **DTV!**

Figure 6.3: Heterogeneous sized vehicles used in case study

6.2.3. Case study 1 - Square base service vehicles from different manufacturers

In this case, the vehicles are of the same size and from different manufacturers. The simplest case study is to check if the algorithm successfully detects deadlocks in advance and initiates vehicle re-route or changes the route to avoid those deadlocks. Visual verification of case study 1 is represented in figure 6.4.

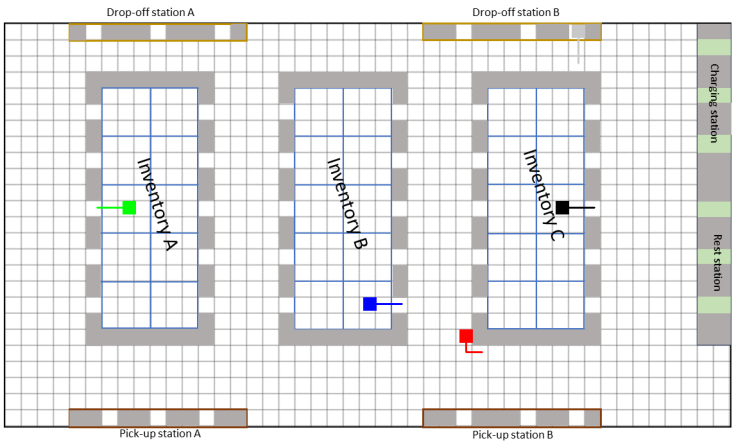


Figure 6.4: Case study 1 visualization

6.2.4. Case study 2 - Square and rectangular base service vehicles from different manufacturers

In this case, the vehicles are square and rectangular-based (AGV forklifts and container trucks) are to be tested. Increasing the level of difficulty for the deadlock algorithm by simulating different sizes of vehicles and heterogeneous fleets, i.e., square occupies one node at a time, but rectangular occupies two nodes while traversing in a straight line and 3 while in turn. A sufficient buffer or

tolerance for rectangular-size vehicles is coded in the algorithm keeping in mind the nodes covered during turn. To check if these lines of instruction are followed, a visualization is done with case study 2. Visual verification of case study 2 is represented in figure 6.5.

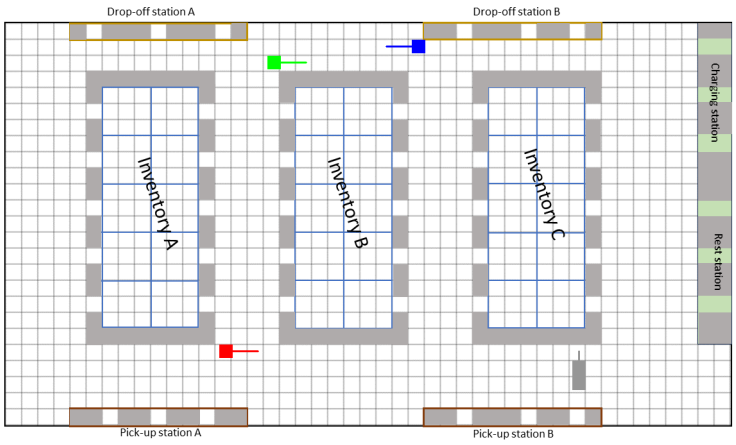


Figure 6.5: Case study 2 visualization

6.2.5. Case study 3 - Square, rectangular base service and square base cleaning vehicle from different manufacturers

In this case, a cleaning robot (square-base) with a cleaning function and a service vehicle (square-base and rectangular base) that does the duty of pick-up and drop-off are tested. In a healthcare environment, cleaning bots are used more often than in warehousing services. The cleaning bot cleans one node at a time and can pose a dynamic obstacle for other service vehicles. The algorithm is also modeled to control these dynamic obstacles. To verify this, the case study 3 is drawn. Visual verification of case study 3 is represented in figure 6.6.

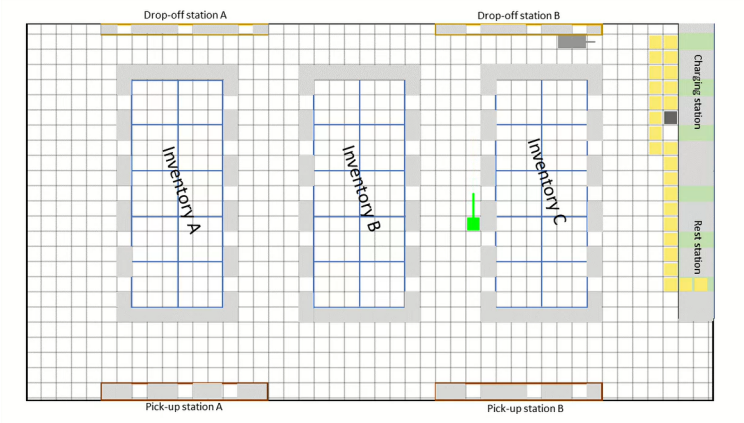


Figure 6.6: Case study 3 visualization

As the operational layout is bidirectional with a well-connected circuit, there is always an alternate path available in case of mission re-route. Hence, the algorithm is successful in detecting deadlocks and avoiding them. As a result, with the verification case studies, no deadlocks were noticed in the visualization window.

6.2.6. Validation

A subjective validation technique with *explore model behavior* is conducted to conceptually validate the model first. The validation output is represented in table 6.3. For the validation of results objectively, there is a lack of real-world data to validate the outcomes of this algorithm. Furthermore, the environment setting and algorithm structure differ from research to research, hence limiting the validation of this algorithm with other scientific papers for objective validation. Nevertheless, conceptual behavior and performance validation are carried out to validate the simulation model.

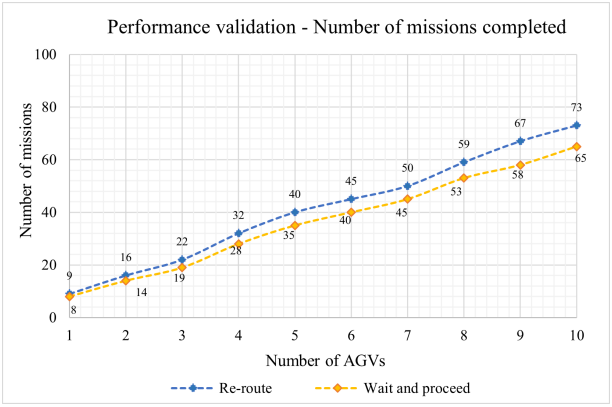
Category	Techniques used	Justification for technique used	Resuslt/ Conclusion	Confidence in result
Deadlock detection and avoidance by vehicle re-routing	Animation	To check if vehicles re-route to avoid deadlocks and complete the mission successfully through time	No deadlocks witnessed in visualization	High
	Face validity	Logical model behavior changes with change in input. For example, with an increase in the number of vehicles, the busyness increases given the same layout and eventually, the availability of routes decreases, hence less system throughput	Logical behavior of the algorithm with change in input parameters	High
	Operational Graphics	Similar to animation, in addition to visualize the change in the system throughput with an increase in the number of vehicles and heterogeneity.	Operations factors like system throughput varies with varying input	High
	Parameter Variability - Sensitivity analysis	The sensitive parameter is the the sequence of the mission assigned to the vehicle in each run. If the the sequence differs, for example, the vehicle gets a mission to source and target with less distance compared to the previous run, then the system throughput varies as the the vehicle has more time to complete more missions due to the smaller mission duration.	Randomess of the mission sequence affected the vehicle travel distance and the system throughput	High

Table 6.3: Conceptual model validity evaluation table

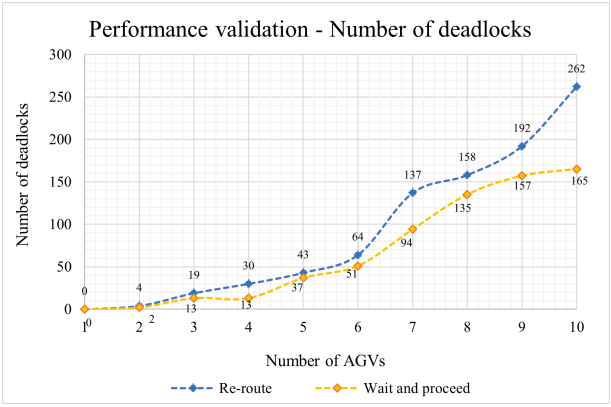
For performance validation in the simulation experiment, the vehicle speed is set to $1m/s$ and the distance from node to node is considered $1m$. Simulation time is set to 600 secs. In an area of $25mX45m$, the maximum distance traveled by AGVs did not exceed 3600m.

With more vehicles, more missions are executed at the same time, leading to decreases in the total distance traveled by vehicles. If less number of vehicles are operating with the same set of simulation times, less number of missions are executed at the same time, leading to increased traveling distances of vehicles. The algorithm shows the same behavior as shown in figure 6.7a and thus validates the model. If re-routing is employed, then logically more deadlocks will occur with more vehicles due

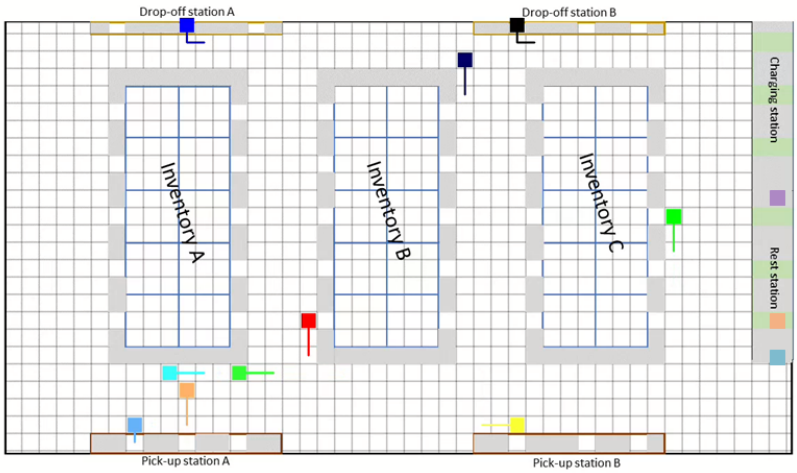
to traffic congestion caused by vehicle re-routing than wait and proceed strategy. This performance behavior of the algorithm is represented in figure 6.7b and thus validated. If the number of missions to be completed is less than the number of vehicles, only the required number of vehicles is assigned missions to complete and the remaining vehicles stay idle. This is visually simulated and represented in figure 6.7c. The number of missions to execute is set to 10 and the number of vehicles input is 13. As the number of missions to execute is 10, only 10 vehicles are given commands to execute, and the rest 3 are idle at the rest/charging stations, and only the required number of vehicles are executing missions.



(a) Number of missions with number of vehicles



(b) Occurrence of deadlocks with an increase traffic and congestion



(c) Only required number of AGVs to execute missions

Figure 6.7: Performance validation of developed deadlock control algorithm

A few aspects of this model are taken into assumption and do not replicate real-world situations. For instance, vehicle batteries are always full and do not require a break in between the operations for charging. No vehicle downtime due to structural and functional breakdowns is considered. Inventories, pick-up, and drop-off locations have infinite space, and orders flow continuously so there is no waiting time at these locations. The simulations hence result in higher throughput than the realistic actual production site. In case the developed model performs better than the other in simulation, it will also follow the same behavior in reality. An evaluation table for model validity is represented in table 6.3. Therefore, this model with a developed deadlock control algorithm is feasible for this thesis study.

6.3. Simulation Experiments and Results

The developed algorithm is compared with *wait and proceed* approach researched in paper [45] because both follow the deadlock prediction. The difference is the frequency of prediction of deadlock and deadlock avoidance approach. For deadlock avoidance, the waiting strategy is implemented in *wait and proceed* and re-route in the novel deadlock control approach developed in this study.

The performance indicators considered are:

- System throughput (number of missions completed/hr): The total number of orders or missions completed by the system with each algorithm per hour. This also gives insight into the optimal number of vehicles required for the missions.
- Waiting time (s): The time taken by the algorithm to find a deadlock-free mission path and dispatch it to the vehicle. Waiting time never exceeds computational time. It is the sum of time taken to route planning (rp) and schedule the mission to vehicle(sm). In addition waiting time in *wait and proceed* also accounts for the time the vehicle waits (vw) to avoid deadlock. In brief,

$$\begin{aligned} T_{DDC} &= T_{rp} + T_{sm} \\ T_{WP} &= T_{rp} + T_{sm} + T_{vw} \end{aligned} \tag{6.1}$$

  where DDC is the Developed Deadlock Control of this project and WP is Wait and Proceed.

  In this study, the detection of deadlock is by prediction approach for both *wait and proceed* and *re-route*, as prediction is required to falsify circular wait deadlock condition. To evaluate the operational performance of the selected strategy, the performance comparison is conducted between *wait and proceed* and *re-route*. As the waiting strategy involves vehicle waiting and the re-route strategy involves more driving time (loaded or unloaded), a fair comparison to evaluate the operational performance considering waiting time is with traffic management efficiency. The vehicle moving time and waiting time are both evaluated to check the efficiency of the algorithm in managing traffic and vehicle fleet coordination as defined in [48]. It is expressed as $\eta = T_{moving}/(T_{moving} + T_{waiting})$. T_{moving} is the moving time of vehicles to complete missions. η for WP and DDC is expressed in equation 6.2. For the developed deadlock control re-route strategy, time consumed by the vehicle in re-route is also taken into account. $T_{waiting}$ is T_{DDC} and T_{WP} as defined in equation 6.1.

$$\begin{aligned} T_{movingDDC} &= T_{moving} + T_{re-routing} \\ T_{movingWP} &= T_{moving} \\ \eta_{DDC} &= T_{movingDDC}/(T_{movingDDC} + T_{DDC}) \\ \eta_{WP} &= T_{movingWP}/(T_{movingWP} + T_{WP}) \end{aligned} \tag{6.2}$$

6.3.1. Operational performance - system throughput

System throughput is defined as the number of missions/orders completed by the system in a given time, here (per hour). Experiments 1,2 and 3 (similar to case studies 1,2 and 3 set up) are simulated for system throughput.

Experiment 1 is only with mobile AGVs, experiment 2 is with mobile and forklift AGVs and Experiment 3 is with mobile, forklift, and cleaning AGVs. In experiment 1, the output showcased is from 7 AGVs to 19 AGVs, as this range showcased insightful output to compare. In experiment 2, cluster A of 10 mobile AGVs with 2,3,4 and 5 forklift AGVs, cluster B of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs, cluster C of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs and cluster D of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs. Similar to clusters in experiment 3, experiment 4 follows the same clustering with a cleaning robot added to clusters A, B, C, and D. These clusters are chosen as they represent an insightful range of output from experiments.

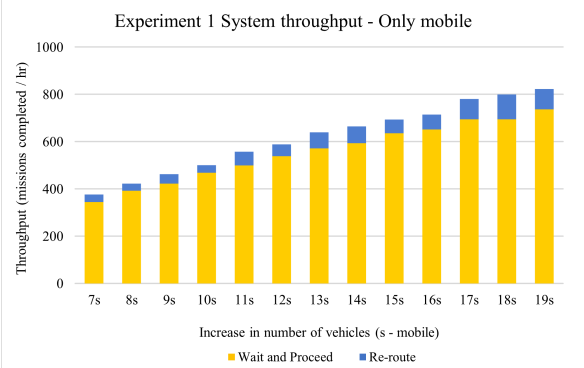


Figure 6.8: Experiment 1 - System throughput (missions completed/hr) performance evaluation

Experiment 1 output presented in figure 6.8, the system throughput of developed deadlock control with a *re-route* approach is increased by an average of 10.46% compared to the *wait and proceed* approach. That means a developed approach completes 10.46% more missions in an hour than *wait and proceed*. To notice is almost stagnant behavior of throughput after 17 AGVs in experiment 1 with a *re-route* approach.

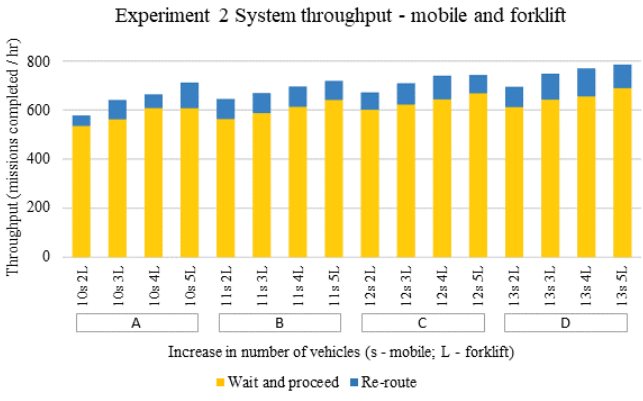


Figure 6.9: Experiment 2 - System throughput (missions completed/hr) performance evaluation

Experiment 2 output presented in figure 6.9, the *re-route* approach shows increased system throughput by 12.15% in cluster A, 13.56% in cluster B, 13.02% in cluster C, and 15.39% in cluster D compared to *wait and proceed*. On average, *re-route* system throughput outperforms *wait and proceed* by completing 13.53% more missions in an hour in experiment 2 with different size AGVs. In

experiment 2 for *re-route*, stagnant behavior in throughput starts to begin from clusters B, C, and D with 4 forklifts.

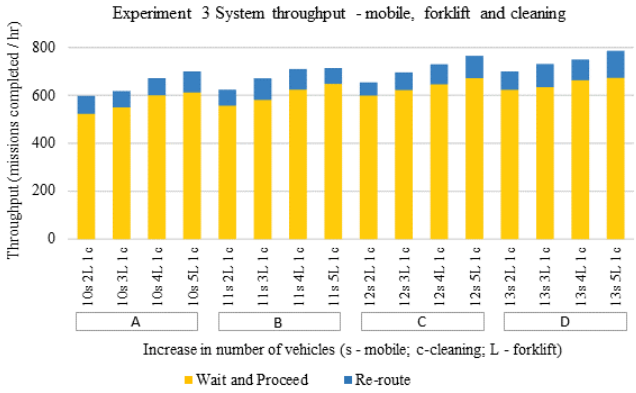
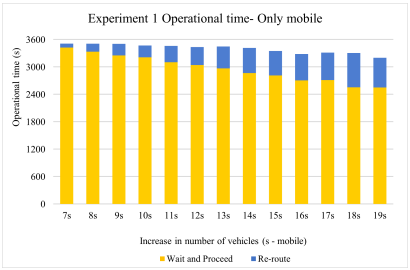


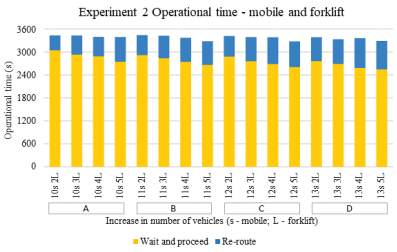
Figure 6.10: Experiment 3 - System throughput (missions completed/hr) performance evaluation

Experiment 3 output presented in figure 6.10, the *re-route* approach shows increased system throughput by 13.27% in cluster A, 12.87% in cluster B, 11.98% in cluster C, and 14.39% in cluster D compared to *wait and proceed*. On average, *re-route* system throughput outperforms *wait and proceed* by completing 13.12% more missions in an hour in experiment 3 with different sizes and functionalities of AGVs. In experiment 3, cluster B with 4 forklifts shows stagnant behavior in throughput.

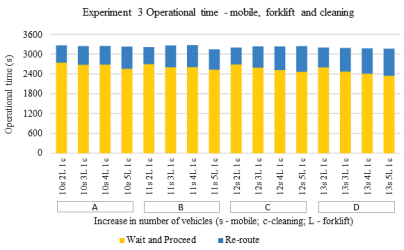
The system throughput *re-route* approach of experiment 3 slightly decreases compared to experiment 2 because experiment 3 introduces a cleaning AGV that traverses node by node with a cleaning mission acting as a dynamic obstacle for other service AGVs causing increased *re-route* and less operational time.



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 6.11: Operational time (s) evaluation with *re-route* and *wait and proceed* approach

An explanation behind the increase in the number of missions by re-route compared to wait and proceed is explained by efficient operational time represented in figure 6.11.
Operational time is the total time spent by vehicles to complete missions. This time is $T_{movingDDC}$ for *re-route* and $T_{movingWP}$ for the *wait and proceed* approach and is calculated following 6.2. The total operational time of wait and proceed in experiments 1,2 and 3 is less than *re-route* as AGVs' time spent to avoid deadlocks by *wait and proceed* is higher compared to the re-routing time strategy, thus yielding more operational time to complete missions with *re-route*.
Even though operational time decreases with an increase in AGVs, but increase in AGVs positively affects the increase in system throughput. Furthermore, the *re-route* algorithm replans the paths to avoid deadlock efficiently in order to increase the operational time for vehicles to complete more missions, thus avoiding infeasible re-routing and inefficient utilization of resources. To prove this point of efficiency of resource utilization, it is explained in section 6.3.2 with outputs represented.

6.3.2. Operational performance - traffic management efficiency

In this project, to evaluate the performance of the developed deadlock control algorithm, its operational performance is evaluated against an existing deadlock prediction approach by [45]. The author developed a scientific deadlock control approach for AGVs in ports and terminals with deadlock prediction and avoiding by *wait and proceed* approach. Traffic management efficiency accounts for vehicle moving time and waiting time and both of these factors are evaluated to check the efficiency of the algorithm in managing traffic as defined in [48], and is used in industrial settings as well.

A common trend in experiments is that the efficiency slightly decreases with a large fleet of AGVs with more congestion and hence traffic control and management complexity increase.

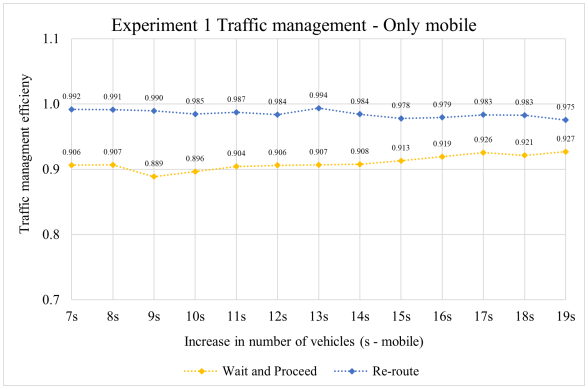


Figure 6.12: Experiment 1- Traffic management efficiency performance evaluation

In experiment 1 presented in figure 6.12, the efficiency of the re-route approach averaged is 0.985, and the wait and proceed is 0.91.

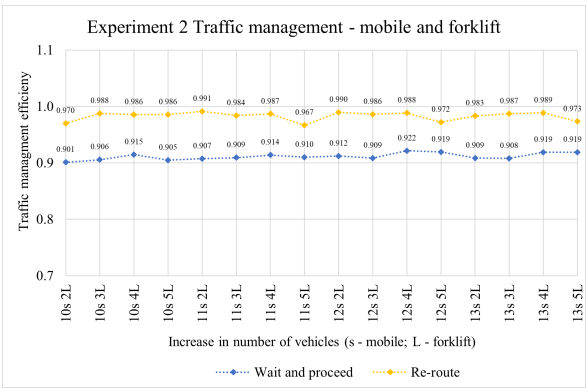


Figure 6.13: Experiment 2- Traffic management efficiency performance evaluation

In experiment 2 presented in figure 6.13, *re-route* efficiency averaged in cluster A is 0.982, cluster B 0.982, cluster C 0.984, and cluster D 0.983, and *wait and proceed* cluster A is 0.907, cluster B 0.91, cluster C 0.915 and cluster D 0.914.

In experiment 3 presented in figure 6.14, *re-route* efficiency averaged in cluster A is 0.988, cluster B 0.983, cluster C 0.99, and cluster D 0.982, and *wait and proceed* cluster A is 0.908, cluster B 0.914, cluster C 0.919 and cluster D 0.92.

Re-route shows increased efficiency of 8.24% in experiment 1, 7.81% in experiment 2, and 7.70% in experiment 3 compared to *wait and proceed*. Within re-route, the efficiency slightly decreases with

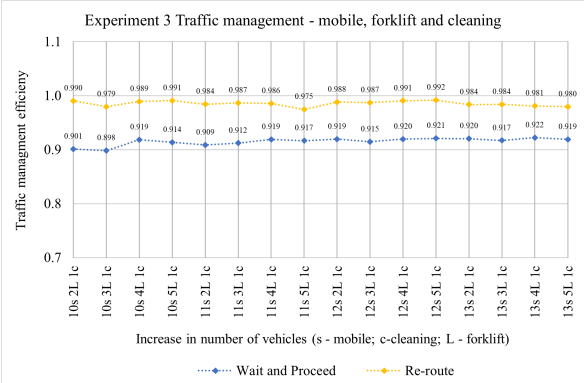


Figure 6.14: Experiment 3- Traffic management efficiency performance evaluation

experiments, as experiments introduce size change and functionality parameters, hence creating more deadlocks and increased traffic management complexity.

6.4. Summary

The developed deadlock control algorithm modeled in the previous chapter is technically deployed or implemented in this chapter to analyze and evaluate its operational performance. The system specifications are; Python 3.8 programming version runs on Ubuntu 22.04.2 LTS system listed in table 6.1.
After the missions are run through deadlock detection check conditions, and re-planned to avoid deadlocks, the final deadlock-free mission path is constructed as a behavior tree to execute actions. The behavior tree forms a route network with elements called *Nodes* and is defined as *parent* and *children*. To distinguish missions from each other, each mission has a unique name and each waypoint also has a unique name.
These mission trees to execute are in .JSON format and shared with HFoV as VDA5050 order messages following VDA5050 JSON schemas via a communication server embedded with VDA5050 protocol interface. The schemas used in this study can be referred to in appendix 9 and for more schemas in the VDA5050 manual.

After the technical implementation, the developed deadlock control algorithm is first verified and validated to check if the model is right and if this is the right model respectively. Three case studies are designed to keep a note of heterogeneity in size and functionality and to check if the algorithm resolves dynamic deadlocks and obstacles if introduced. Case study 1 with similar size vehicles from different manufacturers as simplest to see if the algorithm follows as instructed, case study 2 with different size vehicles from different manufacturers challenging further the deadlock prediction and avoidance ability, and case study 3 with different sizes, functionality vehicles from different manufacturers challenging the algorithm to detect dynamic disturbances introduced by as cleaning robot middling service vehicles time to time. Verification follows techniques like *Animation*, *Face Validity*, *Operational Graphics*, and *Parameter Variability and Sensitivity* checks of the three case studies. Validation is operational and performance. Operational validation is done with a subjective approach with exploring model behavior and graphical displays. Performance validation is carried out by changing the input parameters like the number of vehicles, types, and speed and validating distance traveled. In addition, changing the number of vehicles and traffic congestion, a number of deadlocks detected and avoided is also validated.

After verification and validation, simulation experiments are conducted to evaluate the operational performance of the developed deadlock control algorithm. The developed deadlock control approach

follows deadlock prediction and avoidance by *re-route* approach. To evaluate its performance, [45] which also follows deadlock prediction and employs *wait and proceed* for deadlock avoidance is used as a comparison strategy.

Experiments coined for evaluation are the same as case studies. Evaluation factors or key performance indicators are system throughput to evaluate the number of missions completed in a given time, and waiting time (s) in terms of traffic management efficiency. The *wait and proceed* employs vehicle waiting and the developed approach modeled in this study employs *re-route* to avoid deadlocks. To compare the effective operational time and resource utilization for these two different approaches, the vehicle moving time and waiting time are both evaluated to check the efficiency of the algorithm in managing traffic and vehicle fleet coordination as done in industries [48]. Simulation experiments are run and the results of the operational performance are as follows:

System throughput with the re-route approach is increased by 10.46% in experiment 1, 13.53% in experiment 2, and 13.12% in experiment 3 compared to wait and proceed. Re-route completes 12.37% more missions in an hour than wait and proceed. This is due to the effective operational time. Operational time is the total vehicle moving time. For re-route, it is total vehicle moving time + total waiting time + total re-routing time. For wait and proceed, it is total vehicle moving time + total vehicle waiting time. These both equations are presented in 6.1 and 6.2. Wait and proceed, due to less operational time, has decreased system throughput than re-route.

The operational time is affected by how efficiently the traffic is managed and vehicles are coordinated. This is evaluated with traffic management efficiency which signifies effective deadlock detection, avoidance, and obstacle resolution efficiency of algorithms to complete missions efficiently. The developed deadlock control algorithm's traffic management efficiency is 8.24% higher in experiment 1, 7.81% in experiment 2, and 7.7% in experiment 3 than wait and proceed.

7

Conclusions and Recommendations

The main research question for this thesis is: *How to improve the deadlock control algorithm for efficient traffic management and operational performance of a heterogeneous fleet of AGVs in collaborative intralogistics operation?*

A developed novel deadlock control algorithm with a new approach was presented that is able to employ modified traffic control strategies for a heterogeneous fleet of driverless transport vehicles operating in shared operational space. A vehicle traffic coordination and control policy with deadlock prediction and efficient deadlock avoidance was presented for a centralized traffic management module in an intelligent transport system. This transport system is intelligent by establishing a standard communication interface between the fleet and the traffic management module, and the modules are interconnected with cloud-based services and Internet of Things capability. To study the impact of improved traffic control algorithms compared to existing algorithms, a conceptual simulation model was programmed and experiments were conducted. In this final chapter, the conclusions are drawn by answering sub-research questions and the main conclusion is drawn by answering the main research question of this thesis. Following this are limitations of the work and future recommendations for future research are discussed.

7.1. Sub conclusions

The sub-conclusions are drawn based on answering sub-research questions thoroughly.

7.1.1. Sub research question 1

What is the ongoing area of research in traffic control management for HFoV in collaborative intralogistics operations?

The majority of studies in traffic control management of AGVs in warehouses address the detection of deadlocks edge-by-edge or node-by-node resource checks, and how to trigger the deadlock handling technique to either avoid or prevent them. However, one thing to notice is the detection of deadlocks in the early stages, just after the path plan but before vehicle movement execution. Prior deadlock detection can improve system performance and aid in time management by preventing cyclic deadlock scenarios and controlling deadlocks. This work identified a possible research gap of deadlock detection in advance for the path rather than edge-by-edge and avoidance by re-routing the vehicle path.

7.1.2. Sub research question 2

What is the framework of an Intelligent Driverless Transport System in a warehouse with HFoV and Why is standard communication interface VDA5050 required for warehouses with heterogeneous fleets of AGVs?

Driverless Transport System framework consists of three modules, *Warehouse Management Module*

with embedded *Data Management module*, *Central Controller* with embedded *traffic management module* and *Communication server* which is in direct communication with the homogeneous fleet of vehicles. In order to introduce a heterogeneous fleet of vehicles, the driverless transport system is upgraded to an intelligent transport system. The intelligent transport system has two added features, *cloud-based* services with mission database and dispatch in the cloud, and communication server embeds MQTT broker with VDA5050 industrial standard communication interface for a heterogeneous fleet of vehicles.
For faster processing, an upgrade of the driverless transport system to *intelligent* driverless transport system is necessary. Industry 4.0 technologies combination; cloud-based and IoT called as *CloudIoT* is implemented in the framework. This Intelligent Transport System follows a centralized control architecture for this study. An intelligent system with a cloud-based infrastructure is built to mitigate the computational expense of processing information in a centralized architecture. Additionally, an interconnected network of devices is established in an intelligent system to provide real-time mission and vehicle position updates.
Furthermore, an intelligent system offers a standard communication interface, VDA5050, as the heterogeneous fleet of AGVs exchange data in formats distinct to each manufacturer. Every vehicle from various manufacturers has a special Warehouse Management System installed. By removing the need for each new vehicle to be individually integrated in order to install its management system and lowering integration-related costs, VDA5050 assists in resolving redundancies in the integration of new AGVs into the current Warehouse Management System. When AGVs from various manufacturers are driven inside a warehouse, their situational awareness is compromised, leading to crashes. As a result, each of these vehicles has its own defined functioning region. A heterogeneous fleet of vehicles can co-work in shared space with VDA5050's ability to manage a heterogeneous fleet under a single Warehouse Management System. This allows communication of each operational vehicle status in the system to be shared up to date with one Warehouse Management System. If a specific operation calls for a diverse fleet of vehicles, this can be accomplished with VDA5050 standard communication without requiring the operator to manage the fleet manually and be present on-site.

7.1.3. Sub research question 3

What are deadlock conditions defined in the traffic control policy of warehouse operations and how are these deadlocks handled?

According to Coffman, the system is in a deadlock if any of the four conditions come true. The conditions are *Mutual exclusion*, *Hold and wait*, *Non-preemption*, and *Circular wait*. In a physical system, the first 3 conditions always come true, and with simple path-planning rules, these can be avoided easily. Vehicles stop operating for extended periods of time in physical systems due to the fourth condition *circular wait*, which lengthens trip times and reduces system throughput. In order to prevent circular wait, the development of the deadlock handling method is the main goal of this work. Three categories apply to deadlock handling strategies: deadlock avoidance, deadlock prevention, and deadlock detection and resolution.

Deadlock detection and resolution lets deadlocks occur before taking action to break them. The path is designed with the *lazy optimistic strategy*, *hoping* that there won't be a deadlock. In order to resolve deadlock, it takes time and lengthens the waiting period for vehicles.

Deadlock prevention has prior knowledge of the system, missions, and vehicles and uses this information to prevent deadlocks. It follows static planning of the routes and fails to prevent deadlock in case of dynamic change in mission paths. Researchers create a zone control approach and time window-based planning for prevention in order to detect obstacles dynamically. Zone control limits a vehicle to operate inside a designated zone, which is either fixed or dynamic. Large zones restrict

the system vehicles' capacity to scale, whereas smaller zones result in more mission pauses and longer vehicle wait times to clear conflicts and deadlocks. Prevention with time window-based planning avoids dynamic deadlocks with a computational complexity of $O(N^2H^2)$, an increase in the number of vehicles (N) causing increased computation complexity with time-window (H).

To reduce computational complexity and avoid zone allocation, dynamic resource reservation and upgraded version with *wait and proceed* are coined by researchers. In the former, the central controller reserves resources (or nodes) only if it does not lead to deadlock. The vehicles are assigned nodes as they proceed depending on FIFO (First In First Out) or mission priority basis. The *winning* vehicle moves while the *loser* vehicle changes its status from moving to waiting. The computational complexity is $O(N)$, and due to the waiting strategy, an increase in the number of vehicles creates an unresolved cyclic deadlock situation. To avoid cyclic deadlock, the latter strategy of *wait and proceed* follows a deadlock prediction cyclic frequency and resolves cyclic deadlocks by waiting strategy. The computational complexity of this approach is $O(N^2)$.

Room for improvement is to reduce computational complexity and replace the waiting strategy with straight forward re-route strategy if deadlock is detected. In order to re-plan the mission routes in an effort to increase system throughput, the resolution strategy can be called sooner with the help of the critical role that deadlock prediction plays in controlling cyclic deadlocks. This deadlock control method's operational performance is modeled and assessed.

7.1.4. Sub research question 4

How to design and develop a deadlock control algorithm? What necessary changes are required in developing the novel approach for this study?

Firstly, the resources and surroundings required for algorithm input are specified. The operational system has a bidirectional network with nodes and edges. These nodes form a network known as the mission pathways. The operational arrangement is modeled as a simulation, and a designed GUI is used for user input and model viewing.

The project's operational goal was to model and build a novel deadlock control technique by following two steps; deadlock detection in *advance* as a necessary change to the existing works of literature and deadlock resolution by replanning the mission path.
Four check conditions are modeled in space and time to identify deadlocks by evaluating the nodes along the path and the arrival time at each node using a cumulative weight check. To find out if there are any deadlocks in the mission routes, four check conditions are applied to each executing mission path and new planned paths that have not yet been implemented. Check Condition 1 to detect deadlock between mission vehicles if they share a single node and arrive at the same time. Check Condition 2 to detect deadlock between mission vehicles if they travel in the same direction share multiple nodes and arrive at the same time. Check Condition 3 to detect head-on deadlock between mission vehicles if they travel in the opposite direction share multiple nodes and arrive at the same time. Check Condition 4 to detect deadlock just outside the pick-up/drop-off/inventory station between two mission vehicles where one is exiting the pick-up/drop-off/inventory station and the other is entering that station. These check conditions return a Boolean value; *TRUE* if deadlock is detected or else *FALSE*. If *TRUE*, a mission path replanning with the Dijkstra algorithm is called to re-route the new planned path with the next shortest route in order to avoid the deadlock. If *FALSE*, the mission follows the initially planned path by Dijkstra, and no replanning is initiated.

7.1.5. Sub research question 5

How can the operational performance of the developed deadlock control algorithm of a HFoV's intelligent traffic management module be evaluated?

After the missions are run through deadlock detection check conditions, and are re-planned to avoid deadlocks, the final deadlock-free mission path is constructed as a behavior tree to execute actions. The behavior tree forms a route network with elements called *Nodes* and is defined as *parent* and *children*. To distinguish missions from each other, each mission has a unique name, and each waypoint also has a unique name. These mission trees to execute are in .JSON format and shared with HFoV as VDA5050 order messages following VDA5050 JSON schemas via a communication server embedded with VDA5050 protocol interface. The schemas used in this study can be referred to in appendix 9 and for more schemas in the VDA5050 manual.

After the technical implementation, the developed deadlock control algorithm is first verified and validated to check if the model is right and if this is the right model respectively. Three case studies are designed to keep a note of heterogeneity in size and functionality and to check if the algorithm resolves dynamic deadlocks and obstacles if introduced. Case study 1 with similar size vehicles from different manufacturers as simplest to see if the algorithm follows as instructed, case study 2 with different size vehicles from different manufacturers challenging further the deadlock prediction and avoidance ability, and case study 3 with different sizes, functionality vehicles from different manufacturers challenging the algorithm to detect dynamic disturbances introduced by as cleaning robot middling service vehicles time to time. Verification followed techniques like *animation*, *Face validity*, *operational graphics*, and *Parameter variability and sensitivity* checks of the three case studies. Validation is operational and performance. Operational validation is done with a subjective approach of exploring model behavior and graphical displays. In order to validate performance, various input parameters are changed, such as the number, kind, and speed of the vehicles, and the distance traveled is verified. Additionally, the number of deadlocks detected and avoided is validated, with respect to changes in the number of vehicles and traffic congestion.

Following verification and validation, simulation studies are carried out to assess the created deadlock control algorithm's operational performance. The *re-route* technique is used in deadlock prediction and avoidance in the proposed deadlock control strategy. [45], which uses *wait and proceed* for deadlock avoidance and likewise follows deadlock prediction, is employed as a comparative approach to assess its performance.
Case studies and experiments are developed for assessment. Evaluation criteria, or KPIs, included system throughput, which assessed the number of missions finished in a specific amount of time and waiting time (s) as a measure of the effectiveness of traffic management. To prevent deadlocks, the proposed technique modeled in this study uses *re-route* and the *wait and proceed* uses vehicle waiting. The vehicle movement and waiting times are assessed in order to determine the algorithm's effectiveness in controlling traffic and vehicle fleet coordination, as is done in other sectors, and to compare the effective operating time and resource usage for these two distinct techniques.

Simulation studies are conducted and the results of the operational performance are as follows:
In comparison to wait and proceed, the developed re-route technique increases system throughput by 10.46% in experiment 1, 13.53% in experiment 2, and 13.12% in experiment 3. Compared to wait and proceed, the developed approach completes 12.37% more missions in an hour. The effective operational time was the reason. Operational time is the total vehicle moving time. For re-route, it is total vehicle moving time + total waiting time + total rerouting time. For wait and proceed, it is total vehicle moving time + total vehicle waiting time. These both equations are presented in 6.1 and 6.2. Wait and proceed, due to less operational time, has decreased system throughput than

re-route.
The effectiveness of traffic management and vehicle coordination has an impact on the operating time. This is assessed using traffic management efficiency, which denotes the ability of algorithms to detect deadlocks, avoid them, and resolve obstacles effectively in order to carry out missions efficiently. The developed deadlock control algorithm's traffic management efficiency is 8.24% higher in experiment 1, 7.81% in experiment 2, and 7.7% in experiment 3 than wait and proceed.

7.2. Main conclusion

The main research question
How to improve the deadlock control algorithm for efficient traffic management and operational performance of a heterogeneous fleet of AGVs in collaborative intralogistics operation?

First, a heterogeneous fleet of AGVs working in a collaborative operational layout is modeled for an Intelligent Transport System. By using the industry standard communication protocol VDA5050 to interact with a diverse fleet of AGVs, this Intelligent Transport System fulfills the study's technical research goal.

Secondly, a novel deadlock control method that uses check criteria for deadlock detection in a dynamic environment and commences a re-route strategy to successfully avoid deadlocks is built into a centralized traffic management module. For operational assessment, this strategy is simulated, used in three tests, and compared with the existing wait and proceed method for deadlock avoidance. The evaluation factors used are system throughput (number of missions completed in a given time) and traffic management efficiency to evaluate the vehicle moving time and waiting time and check the efficiency of the traffic management and vehicle coordination by algorithms.
The results of the experiments show the system can complete 12.37% more missions per hour with less number of vehicles with the novel deadlock control algorithm developed in this project. In addition, the traffic management efficiency of the developed algorithm is higher by 7.91% than the wait and proceed strategy. With this operational objective is achieved.

To answer the main research question, the industries can improve the deadlock control algorithm by implementing deadlock prediction in advance and employing re-routing of vehicles to avoid and falsify circular wait deadlocks. This can be employed in an operational layout with a heterogeneous fleet of vehicles for increased interoperability with VDA5050 in complement with a novel deadlock control algorithm for an increase in the operational output of the system.

7.3. Limitations and recommendations

7.3.1. Layout network

The layout used in this thesis is a bidirectional networked layout. The route network in the layout is strong, which makes more routes available at once to avoid deadlock and route re-planning. This influences the operational performance outcome like system throughput. Many industries use non-tessellated guided paths like unidirectional, running parallel to each other. For this approach to showcase its impact more closely in a real-world setting, it is recommended to study this approach on different layouts and evaluate its performance.

For path planning, a heuristic simple Dijkstra algorithm is used in this study. Advanced algorithms that solve complex networks faster are recommended for future research.

7.3.2. Effective mission distribution

In this investigation, the central controller is unable to assign missions efficiently when there are several vehicles sharing a pick-up station. Here, in order to prevent arriving at the pick-up point simultaneously, the controller directs the vehicles to finish the first in-queue by re-routing. It is possible to prevent this needless re-routing and vehicle movement by properly arranging the missions. In this instance, it is advised to distribute the missions wisely by assigning the next mission to the vehicle to complete first and returning to the first mission once the pick-up location is empty. This allows the vehicles to complete tasks more quickly by completing them in an ad hoc manner.

7.3.3. Information exchange

The information exchange system model is created with the assumption that, while a mission is being carried out, the AGVs' sensors for local mapping and obstacle detection are turned off. FMS transmits mission data to the fleet of vehicles for execution. The developed deadlock control algorithm analyzes traffic control policies and arranges paths so as to identify and avoid obstacles and deadlocks that are already a part of the system. For instance, if an operator whose information and vehicle path are unknown and not fed in the system, then the path remains unknown to the system and is not checked by the algorithm.
The system has to be linked to sensors on board AGVs in order to identify vehicle movement insights that are not currently recorded in the system. This work may be extended by processing data from the onboard sensors, which can be used to identify and guide vehicles locally to avoid unknown obstacles based on visible information.

7.3.4. System architecture

A centralized architecture for the flow of information is employed in this study to achieve global optimization. However, to improve the processing time and decrease computational time, the study of other architectures like distributed or non-centralized is recommended.

7.3.5. Assumptions

A few presumptions were made in order to carry out the experiments. It is assumed that order availability on a constant basis eliminates the time needed for the vehicle to load at pick-up and inventory sites. Additionally, assuming infinite space and no conflicts, it is expected that these sites may accommodate more than two vehicles entering at the same pick-up or drop-off time, but not simultaneously. Additionally, it is assumed that vehicles always have enough battery power, therefore no consideration is given to recharging times. During re-routing, these vehicles consume more energy than when wait and proceed situations. When vehicle charging time is taken into account, fewer vehicles may be available. It may be necessary to add more vehicles in order to boost the system throughput. Future research endeavors should take this crucial study point into account. Overall, by removing these presumptions, this study may be expanded for more accurate results.

8

Appendix A - Research paper

A novel deadlock control algorithm for a heterogeneous fleet of autonomous transport vehicles in a collaborative intralogistics environment

Ashwini Rathi, Ir. Mark Duinkerken, Ing. Marloes Hengeveld

***Abstract*— For high efficiency and flexibility, a fleet of Automated guided vehicles (AGVs) both homogeneous and heterogeneous are widely used automation products for material handling in warehouses and automated production lines. Given the layout capacity, the AGVs interact with each other, which provokes challenges in Driverless transport vehicle system (DTVS) traffic management in dynamic environments. One of the main challenges is how to avoid collision and deadlock between heterogeneous fleets of AGVs in a bidirectional layout. This research study proposes a deadlock detection and avoidance algorithm that follows the deadlock prediction and uses a dynamic re-routing strategy with Dijkstra to avoid deadlocks. The mission paths are checked in space and time for overlapping edges by four check conditions and cumulative weights and return a boolean value to avoid cyclic deadlocks. For the heterogeneous fleet of AGVs, a standard communication protocol VDA5050 is used to maintain a standard communication interface between vehicles and the traffic management module with cloud-based microservices for increased processing time and interoperability within the warehouse. This communication interface is used to communicate the novel deadlock control algorithm to a heterogeneous fleet of AGVs. The proposed algorithm not only improves the throughput by increasing vehicle operational time but also successfully avoids congestion and deadlocks with high traffic management efficiency in the logistic transportation system.**

Keywords: Traffic control, heterogeneous AGVs, interoperability, deadlock resolution, warehouse

I. INTRODUCTION

With the increase in the global automation market CAGR by 14%, the intralogistics industrial products, AGV's (Automated Guided Vehicles) CAGR is forecasted to grow by 6.06 % by 2025. Industries implementing these products aim for efficient and faster material handling operations. These vehicles interact with each other often, so it is necessary to implement traffic control policies in order to avoid congestion and deadlocks for risk-free operations. Due to an increase in production costs and automation, industries aim for logical yet feasible operational performance requirements such as higher system throughput, and traffic management efficiency.

This paper focuses on developing a novel deadlock detection and avoidance algorithm for a heterogeneous fleet of AGVs in a driverless transport system. A wide array of decisions have to be made in order to control traffic in the system, for instance, the operational layout design, traffic control policies, number of vehicles, type and functionality of vehicles.

For smooth, continuous operations in the transport system, traffic control policies are drawn to avoid collisions and handle deadlocks correctly and efficiently [4]. The selection of these policies that affect the performance of the driverless transport system is critical and is used for mission routing, scheduling, collision, and deadlock avoidance during mission operations. Since one or more mission paths have an overlap, and if the vehicles arrive at those overlapping nodes at the same time it causes multiple vehicles to permanently block the node and the path, and is defined as a deadlock situation. This causes operation halt and hence it is necessary to develop more efficient traffic control strategies to ensure continuous operation of the transport system [3]. Therefore, the operational objective of the paper is to develop a novel deadlock control algorithm for efficient mission execution for a fleet of AGVs.

The traffic control strategy embedded in *Traffic Management Module* for a heterogeneous fleet of vehicles has both operational and technical challenges. The heterogeneous fleet of vehicles is distinguished by structural heterogeneity and functional heterogeneity. The vehicles are considered structurally heterogeneous if they differ in design and dynamics, for example, bulk body, aerodynamic body, vehicle speed, payload capacity, fuel consumption, etc. On the other hand, functionally heterogeneous vehicles if not all vehicles are executing the same field of operation. For instance, a cleaning vehicle equipped with cleaning instruction software functionally works differently from a service vehicle whose software controls are for service pick-up and drop-off. These vehicles are assigned to visit the source point and target point while cleaning visits all the points in the configuration space. In a heterogeneous fleet, vehicles, due to under-knowledge of the vehicles in the system, see each other as dynamic obstacles and can cause deadlock. This prompts a technical challenge and leads to the technical research objective of implementing the deadlock control algorithm developed under operational objective on a *heterogeneous fleet of AGVs* operating collaboratively in the warehouse. The contributions of this work are:

- Intelligent transport system for a heterogeneous fleet of vehicles: For warehouses with heterogeneous vehicles, the traffic management module should possess interconnectivity between and among the fleet of AGVs for ease in interoperability, collaborative operations between fleets and ease in the integration of new vehicles to the existing warehouse system. This is a technical

research objective respective to the heterogeneous fleet of AGVs for integrated and collaborative warehouse operations.

- A novel deadlock control algorithm: An algorithm with check conditions for deadlock detection, strategy to avoid deadlocks and execute missions effectively. This is the operational research objective for efficient mission execution by the *heterogeneous* fleet of AGVs with an aim of efficient traffic management and increased system output.

Before physical implementation, it is critical to check the feasibility of these decisions in a virtual environment, which also takes into account the complexity and dynamics of the transport system. For this, simulation models are typical approaches [1] used in this paper, hence mapping the correct representation of the decisions for near approximate to the exact representation of the planned system.

II. RELATED WORK

According to [6], the material handling process accounts for up to 70% of total manufacturing costs, which eventually leads many industries to shift to highly automated solutions for higher turnover. One of the systems under Automated material handling operation is AGVs. The performance of the AGV system highly depends on the layout of the system (both cyber and physical layout) and the strategies used for controlling the vehicles. Control strategies for such AGV systems should at least perform the following three functions: path planning, task assignment, and traffic control [7]. The main challenge with the heterogeneous fleet of vehicles lies in the area of navigation, deadlocks created in case two different vehicles fail to recognize each other as vehicles but navigational blocks and halt themselves, failing to complete the missions successfully. In addition, the vehicles in the fleet are from different manufacturers or with different functionalities and come with their own central controller. In this case, it is a challenging task to operate these heterogeneous vehicles with different central controllers. It, thus, turns out to be a non-sustainable approach in case of the factory floor expansion which demands for increase in the vehicle types. Therefore, a one-in-all central controller with vehicles communicating bidirectionally with standard protocols is the next research area in *factory of future*.

Deadlocks, explained in this article discusses three ways to resolve deadlocks; deadlock detection and recovery, deadlock avoidance, and deadlock prevention.

Deadlock detection and recovery handles deadlocks in two steps; deadlock detection and recovery and is termed as *lazy optimistic strategy* [20]. It reserves or schedules a route if available *hoping* that the planned route will be deadlock-free and works successfully for the systems with occasional deadlocks. A major disadvantage is that this strategy is unable to predict deadlock even after certain information on deadlock occurrence in the future is available in the system. Deadlock prevention is an offline strategy, where the system schedules paths in such a way that is deadlock-free in prior. For bidirectional layout and higher throughput, a prevention strategy with faster network routing is beneficial in terms of cost reduction with a reduction in used areas by vehicles [10], but however limits in preventing deadlocks in case of dynamic disturbances in the operational layout. With time-based route planning, certain segments/cells in the route of a vehicle are allocated and the rest of the segments in the route are utilized by other vehicles. In the paper [11], each node in a cell of the operational layout has a list of time windows reserved by vehicles and a list of free time window that is available for the vehicles to reserve. In this way, the algorithm plans the vehicle route with free time windows in the proposed time-window graph instead of physical cell nodes of the operational flow path. The computation time is then $o(v^4n^2)$ where v is the number of vehicles and n is the number of nodes. This means, that with an increase in the number of vehicles, the computational complexity increases, which makes it suitable for only small transport systems. Factors such as acceleration, deceleration, and external obstacles in dynamic environments make it difficult to calculate time windows precisely in case of delays and can lead to unpredictable obstacles.

Similar to the prevention strategy, deadlock avoidance avoids the occurrence of the deadlocks. This policy plans the mission operations dynamically depending on the system state such that it remains deadlock-free. Resource allocation graphs are useful for detecting deadlocks. Dynamic resource allocation can affect the resource/zone utilization and system throughput, and eventually increase the lead time and vehicle travel time [14]. In terms of transportation systems, a Banker's algorithm is a resource allocation and deadlock avoidance strategy that predetermines if the system will remain in *safe state* or not by first simulating the allocation of node(s)/cell(s) to the mission paths. It then makes a *safe state* check before actually allocating the node(s)/cell(s) to the mission to navigate through them in order to avoid deadlock and implements *wait-for* approach to avoid the unsafe state navigation by vehicle waiting strategy. In article [7], the central controller calls deadlock avoidance every time to check if a particular vehicle is allowed to *reserve* a set of cells/tiles required for its navigation. The algorithm determines if the reservation is allowed only if the order of movements of AGVs exists such that the system remains deadlock-free. The combination of steps of different vehicles is checked by the algorithm to evaluate the system state, which causes unnecessary deterring of AGVs and longer calculation time. A modified Banker's algorithm strategy proposed by [15] provides a solution to solve the low utilization of vehicles and longer waiting times. Under special circumstances, the vehicles are allowed to transverse unsafe states (deadlock existing states) to decrease the vehicle waiting times and mission execution times. This extended set of states is allowed only if all the missions can be executed safely, or else the vehicles wait until the state is safe. An extension of this modification is proposed by [16] for an improved near-optimal deadlock avoidance strategy. It consists of two stages, an offline which preprocesses guide-path, and an online which combines preprocessed guide-

path results with the vehicle status to evaluate the safety of the system dynamically. In order to avoid the vehicles waiting until the overlapped edges of two paths are released, alternative shortest paths are obtained. Before a path from alternative paths is allocated to a vehicle, a safety assessment of the system is carried out. If all these alternative paths lead to an unsafe state, the vehicle is instructed to pause and wait for the next scheduled check. This questions the trade-off between travel distance in case of alternative path allotment or waiting time until the system gets back to *safe* state.
In dynamic environments, the dynamic reservation of node(s) or resources(s) is the avoidance of cyclic deadlock formation. In [13], before the dynamic reservation of node(s), a *wait and proceed* deadlock prediction cycle is run frequently issuing new commands to AGVs i.e., new control points (nodes) to occupy one at a time which effectively avoids the occurrence of cyclic deadlocks. The prediction algorithm computational complexity depends on the number of vehicles (N) i.e., $O(N^2)$. An important point to note is how frequently the predictions are made. The sooner the prediction, the better avoidance measures can be taken.
Most of these papers discuss detecting deadlock resource-by-resource or node-by-node, then prompting the deadlock handling strategy to either prevent or avoid them. However, detecting deadlocks in prior just after path planning and before vehicle movement execution is a factor to consider. Deadlock detection in prior can help in time management to control deadlocks, avoiding cyclic deadlock situations, and can positively affect the system throughput as noticed [13]. Therefore, the potential research gap noticed in literature is deadlock detection in *advance* for the path and not edge-by-edge, and hence will be studied in this paper, and its operational performance will be evaluated.

III. METHODS

DTVS (Driverless Transport Vehicle System) is an authorization-based system where vehicles need permission from DTVS for every action to be performed. The system consists of three modules, a warehouse management module, a fleet management system (central controller), and a communication server (wired/wireless) as represented in figure 1.

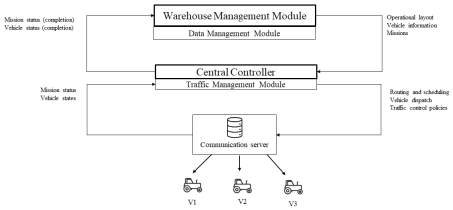


Fig. 1: Driverless Transport Vehicle System modules

- Warehouse management module: The data management module consists of information about each vehicle in the fleet, mission requests, and operational space information. This data is input into the fleet management system.
- Fleet management system (central controller): It is considered as *high-level fleet management* which supervises the fleet of vehicles by planning, coordinating, and controlling the fleet. It provides necessary instructions to the vehicles to accomplish the order pick-up/drop. The Traffic Management module which holds traffic control policies is sub-embedded in the Fleet Management System.
- Communication server: Communication servers are used to send/receive mission information to and from a fleet of vehicles.

Each heterogeneous fleet of vehicles has a communication format unique to its manufacturer. To establish communication of data from the fleet management system or traffic management module to and from the fleet of vehicles, it is necessary to implement a standard interface between vehicles and *traffic management module* for an adaptable, robust system.
With Industry 4.0 technologies such as *Internet of Things* and *Cloud computing*, interconnectivity among different modules and machines becomes easier. This system is termed as *Intelligent Transport System* [5] which is an upgrade in DTVS.

A. Framework of the intelligent transport system

A representation of this Intelligent transportation system is in figure 2.

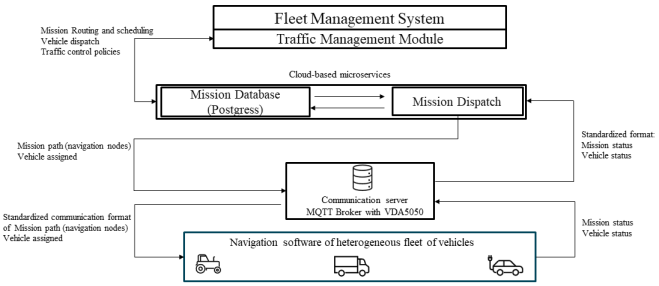


Fig. 2: Intelligent Transportation System with *CloudIoT*

1) Fleet Management System

Transportation system management (Fleet Management System) includes vehicle dispatching, routing, and scheduling, traffic control for collision and deadlock avoidance, and maintenance strategies.

- Mission Routing and scheduling: Mission orders are received from the Warehouse Management system, and the mission route is first planned and assigned to the vehicle. The scheduling takes into account the departure and arrival time from the pick-up location to the delivery location, by the cumulative weights (distance between two connected nodes) from the pick-up to the drop-off location. This cumulative weight calculation is used for collision and deadlock-free scheduling.
- Vehicle dispatching: Dispatching is when an order is assigned to the vehicle or vehicle to the order. Dispatching

methods can vary from time constraints, priorities, or the nearest idle vehicle available. The simple dispatching rule is assigning an order to the idle or next available vehicle with the shortest or nearest travel distance/time from the order location. This rule is followed in the current study.

- Traffic control policies: A planned route is feasible for vehicle operation with appropriate traffic control policies to account for collisions and deadlocks. These control strategies are closely related to dispatching, routing, and scheduling.

2) Cloud-based low-level Fleet Management System

It is 'low-level' fleet management that runs in the cloud or edge. The microservice module manages communication with the software stacks of the vehicles (navigational stack here). In addition, it is responsible for executing the mission node-by-node in the path, keeping track and collecting feedback information such as position, status, and error from the vehicles and feeding it to the FMS.

A centralized architecture meets the objectives of this research that are suitable for modern vehicle fleet planning and coordination. However, centrally processing large amounts of planning and coordination data sounds inefficient, however, this is also a good approach to obtain globally optimal solutions in case of task planning, allocation, and execution, thus providing an effective decision-support tool. Another problem with centralized architecture is computational expensiveness. This is solved by outsourcing cloud-based infrastructure for computationally heavy processes, In addition, these systems come with sophisticated cooling systems that tick green with energy-efficient system goals.
In case of network loss or disconnectivity, the mission database is a cloud-based microservice that can rebuild the states and continue from the information stored in the database, so the mission is not lost. It can be believed that the system, instead of leading to downtime, will ensure that it maintains operability, thus cloud-based low-level fleet management is used in this framework.

3) Communication server - MQTT broker

MQTT broker is used for communication between the cloud-based low-level fleet management and the navigation stack of the vehicle. MQTT is a lightweight, text-based message exchange protocol (message broker) with IoT publish/subscribe model and consists of clients that are divided into subscriber and publisher. The subscriber is a message-receiving client who is registered with the broker and notifies it to receive specific types of messages. The publisher is a message-sending client who sends a message to the subscriber when asked through the broker. For instance, low-level fleet management is subscribed to receive state and factsheets from vehicle software to keep an update on the vehicle status and publishes mission *order*, *Actions*, and *instantActions* to vehicle software to execute missions with required actions. On another side, vehicle software is subscribed to *order*, *Actions*, and *instantActions* published by low-level management. Here, Mosquitto is used as an MQTT broker.
MQTT allows the distribution of messages to sub-channels termed as topics. Clients (here mission dispatch module and software on the vehicle) subscribe to these topics to receive the required information that interests them. Topics concerning the current study are *Order* (Communication of driving orders like nodes to navigate from FMS to vehicle), *Action* (Action to be executed sent from FMS to vehicle), *instantActions* (any immediate actions to be executed), *state* (vehicle state) and *factsheet* (vehicle setup). These topics can be referred from the VDA5050 manual. Mosquito is a lightweight MQTT broker with the capability to exchange large amounts of data over low network overhead, with limited network bandwidth and interrupted communication. Thus, it can be implemented on low-power devices like microcontrollers used in remote IoT sensors.

4) Communication server - Industrial Standard Protocol VDA5050

The communication server acts as a bridge between the cloud/edge microservices and the vehicle in order to exchange topics (order, status) between the FMS and software on vehicles. This interface has some requirements such as it has to be agnostic with different software stacks such as ROS or Isaac SDK that run on vehicles. In addition, the interface should facilitate a simplified connection strategy of new vehicles into an existing FMS, and enable parallel operation between vehicles from different manufacturers and inventory systems in the same operational environment. It should be cloud-friendly and scalable to large numbers of vehicles for smooth integration into the existing system. One such industrial standard protocol was developed by
VDA5050 and was established in 2022 between the Verband der Automobilindustrie e.V. (German abbreviation VDA) and Verband Deutscher Maschinen-und Anlagenbau e.V. (German abbreviation VDMA) with an aim to create universally applicable interface tool. It defines a messaging structure and uses the MQTT network protocol to publish/subscribe the message structures.

VDA5050 helps to solve the redundancy in the integration of new vehicles into the existing warehouse management system, hence eliminating the individual integration of each new vehicle to install its management system and reducing integration-related costs. If vehicles from different manufacturers are operating in the warehouse, they lack situational awareness and cause collisions. Due to this, these vehicles are allocated separate bounded operational areas for each. By managing a heterogeneous fleet under one warehouse management system, VDA5050 allows communication of each operating vehicle's status in the system to be shared to date with one warehouse management system, hence allowing a heterogeneous fleet of vehicles co-working in shared space. In case a particular mission requires a heterogeneous fleet of vehicles, this can be possible with VDA5050 standard communication and also without the need for the operator to manually control the fleet.

The technical research objectives coined in this paper are achieved by *Intelligent Transport System* framework. Now, to execute missions with efficient traffic management of heterogeneous fleets in intelligent transport systems, a novel deadlock control algorithm is modeled and analyzed hereafter.

B. Traffic Management Module Analysis

An embedded module in the Fleet Management System *Traffic Management Module* has the functionality of traffic control strategies with rules and algorithms for conflict-free and deadlock-free operations, hence, being one of the evaluating factors of the operational performance of AGVs. One fundamental problem with a heterogeneous fleet of AGVs is the multi-vehicle traffic control strategy for collision-free paths or deadlock-free path planning of each vehicle in the network [17]. Mishandling of multi-vehicle traffic leads unresolvable traffic congestion and higher operations costs because of the halt of missions due to deadlock occurrence and risk to human operators on-site. Hence, collision and deadlock handling strategies for multi-vehicle, especially heterogeneous systems have a critical study discussion in this research.

For traffic control strategy, the transportation system is divided into *Transportation System Design*, *Vehicle Management* and *Transportation System Control*.

1) Transportation System Design

Transportation system design discusses the structure of flow-path layouts, such as *unidirectional*, *bidirectional*, and *multi-lane*. With a unidirectional path, no opposite traffic is allowed, hence requiring simple controls. One-way traffic has less layout utilization and higher vehicle traveling distance with less number of missions completed. Bidirectional flow allows two-way traffic, facilitates transportation cost reduction, and minimizes the area used by increasing the transportation network and hence higher throughput. A bidirectional path requires a smaller number of vehicles and is more advantageous than unidirectional.
Due to bidirectional traffic, congestion increases and system complexity increases. As vehicles travel two-way, it is then necessary to check the accessible route available for vehicles in order to avoid collisions or deadlocks when traveling in the opposite direction. That means mission scheduling and vehicle dispatching are closely related to vehicle routing. This then requires an adequate traffic control algorithm for the bidirectional flow of vehicles.

2) Transportation System Control

Bidirectional system control can be classified as centralized, decentralized, and distributed. In the case of decentralized, the decisions are made based on local information. Here, the vehicles are aware of their state and the state of neighboring vehicles. The traffic management is handled and communicated between these vehicles and negotiated by themselves. Decentralized control has low computation and a simpler solution approach but has low efficiency. A distributed control, algorithm uses a combination of global and local information. In the case of centralized control, all the information about the vehicles, positions, planning, and coordination of transportation systems are stored and computed in one place. It is highly efficient but requires longer computation time. For computational expansiveness of centralized control, outsourcing cloud-based infrastructure for computationally heavy processes can be employed.

3) Vehicle Management

Vehicle management involves deploying traffic control algorithms in order to avoid collisions and deadlocks and managing vehicle coordination. There are four deadlock conditions that, if true, can pose traffic management challenges and can halt vehicle movement. The deadlock conditions, handling strategies, and necessary changes required to select a strategy to model for this study is discussed hereafter

4) Deadlock categorization

For a situation to be posed as deadlock, Coffman coined four conditions as explained below [18]:

- Mutual exclusion: This condition states that one cell can be occupied by one vehicle at a time. It ensures that if two vehicles share a common path, then one cell can be reserved by one vehicle at a time.
- Hold and wait: This condition holds true when one vehicle waits for a cell to be free which is currently occupied by another vehicle.
- Non-preemption: This condition is met when the cell is released only when the vehicle (V1) occupying that cell has left it completely. It is infeasible to remove the vehicle (V1) occupying the cell until the completion of the assigned order at that particular cell.
- Circular wait: This condition holds true when a circular chain of vehicles waiting for each other to move to the cell that is currently occupied by the next vehicle in the chain

In order to prevent the occurrence of deadlocks, at least one of the four above-mentioned conditions should be broken. In DTVS in the warehouse, the first three conditions are always true in physical systems and are solved by efficient path planning. In the context of the problem of this study, the breaking of Coffman's fourth condition *circular wait* requires more than path planning to avoid the occurrence of cyclic deadlocks.

5) Deadlock handling strategies

The deadlock handling strategies are classified into three; deadlock detection and recovery policy, deadlock prevention policy, and deadlock avoidance policy.

Deadlock detection and recovery handles active deadlocks. That means this policy allows the occurrence of deadlocks. These deadlocks are detected and recovered by another algorithm that re-plans the path of at least one vehicle that is in a deadlock situation. This approach does not prevent deadlocks in advance. While deadlock prevention and avoidance handle

passive deadlocks, i.e., resolving deadlocks in advance.
Deadlock prevention is an off-line traffic control policy that aims at the complete avoidance of any situation that may lead to a deadlock by pre-planning 100% deadlock-free paths prior to the mission execution. It follows the static planning approach and computes routes that do not require change and are robust against disturbances.
Deadlock avoidance is an online control policy in real-time that avoids the occurrence of deadlocks in the next event by dynamically allocating the vehicle's mission cell based on the information of the current state of the system. This approach is used for resolving dynamic obstacles that can create a deadlock situation.
Detection of deadlock sooner by running a prediction cycle has efficient time management, implements an avoidance strategy sooner, and therefore avoids the occurrence of cyclic deadlocks. A thorough explanation is carried out in section II.

Questions like; in case of non-re-routing, is it possible that vehicles wait indefinitely? In some scenarios, this is not the best possible solution. The waiting strategy can cause persistent cyclic deadlocks (single and multiple) as vehicles involved in this scene are instructed to wait and rely on each other for movement as approached by most of the algorithms reviewed in the literature. An alternate solution is to compare paths and replan or re-route the vehicles. This further raises some questions. If a re-route is initiated, is there always an alternate path available? If two vehicles have the same source point, does one vehicle have to wait for a very long time until the zone is deadlock-free or is it possible to assign a different mission with a different source point?
In case for the system to be deadlock-free, Coffman's fourth condition Circular wait should be eliminated. The algorithms for avoidance focus on waiting until a mission path has a safe state. The straightforward way to implement a deadlock avoidance strategy is re-routing instead of waiting. Re-routing by path planning also helps to avoid Coffman condition 4 of Circular wait by allowing the mission to break from the cyclic deadlock and to re-route and continue operation. In paper [19], authors propose a dynamic routing algorithm that avoids deadlocks efficiently. The dynamic routing calls the Dijkstra algorithm for active path reservation and considers waiting time in the cost, making it possible to choose between waiting until no overlap or detouring. However, the performance of this method is limited to the layout used in that study. From research by [21], a basic principle of deadlock-detecting prediction cycle can be derived to detect deadlocks in advance, and instead of the *wait and proceed* strategy used, an alternative approach to study and model is to prompt vehicle re-routing to avoid deadlocks.

6) Mission path planning

Proposed by Edsger Dijkstra in 1959, the Dijkstra algorithm is easily implementable and adaptable to topology or configuration space change and dynamic environments for path planning. The configuration graph is divided into nodes and connected to each other by edges. A graph can be undirected, directed, or weighted. Each weight can either be a distance or time between the two connected nodes. In this study, the layout flow path is bidirectional.

It is a famous solution for the shortest path problem was given by Dijkstra. It is a greedy algorithm that solves the single-source shortest path problem for weighted graph $G = (V, E, w)$ with non-negative edge weights, i.e., $w(A, B) \geq 0$ for each edge $(A, B) \in E$ with n nodes and e edges and E is set of edges and V is set of nodes. If the edge $<A,B>$ does not exist, the the weight $w(A,B) = \infty$. $d(x)$ is the distance from source node s to node x. Let S denote the set of nodes that are included in the shortest path, which means, initially S contains only source node s. V-S then denotes the set of nodes that are not included in the shortest path yet and follows [**?**]:

1) Initialization: Set the source node s, and set $S = S\ U\ s$.
2) In V-S set, find node i that is adjacent to the source node s. If the weight of the connecting edge from s to i is shortest, then add i to set S.
3) Let i be the new intermediate node. Repeat step 2 to find the next smallest numbered adjacent node j from V-S set. Update the distance between source node s and node j. If $d(j) > d(i) + w(i,j)$, which means if the distance passing through i is shorter than not through it, then modify $d(j)$ to $d(j) = d(i) + w(i,j)$, and then add node j to S.
4) Repeat step 2 and 3 for the n-1 iterations, where n is number of nodes in the graph G. The output then consists of the source node, intermediate nodes, and target nodes with the shortest path from the source to the target node.

In this study, to handle dynamic disturbances and obstacles, a potential research gap in detecting deadlocks sooner will be modeled. To avoid vehicles getting into circular-wait deadlock by waiting for strategy until the deadlock is cleared, a straightforward approach of re-routing the mission using the Dijkstra algorithm to avoid deadlocks will be modeled as avoidance strategy in IV.

IV. MODELLING

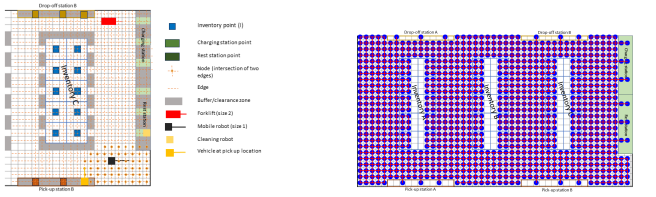
A. Selected strategy model objective

The objective is to model and develop a two-step deadlock control algorithm to effectively falsify *circular-wait* deadlocks; deadlock detection by check conditions before the mission execution or vehicle movement and avoidance of detected deadlocks by re-planning the mission path by re-routing.

1) Operational layout and simulation model

Before developing the algorithm, the operational environment and components required for the input to the algorithm are to be defined. The warehouse layout is divided into operational guide/mission paths with bidirectional flow paths connected via a network of nodes and edges for the vehicles

represented in figure 3. Each node has an x and y coordinates. These nodes with coordinates are input to the algorithm environment as a .csv file.



(a) Bidirectional layout with nodes and edges

(b) Network of bidirectional layout

Fig. 3: Bidirectional networked layout used in this study

In this layout, the vehicles travel bidirectional, which is advantageous to the driverless transport system operation because vehicles are allowed to take shortcuts in order to reduce the travel distance and time. These shortcuts can lead to potential deadlocks and can be resolved with a deadlock control strategy.

The simulation model of the layout consists of 3 inventories each with 10 inventory points, 6 pick-up and 6 drop-off locations, 3 charging, and 3 rest stations.

A Graphical User Interface (GUI) is designed to pass a set of instructions from the user, i.e., input to the Fleet Management System (also called as central controller) and log operational updates on the window. It is divided into three sub-windows, the left window has different tabs for input, the middle window shows the operational layout and vehicle movement during the mission run and the right window shows mission/vehicle/operation states and updates as represented in Figure 4.

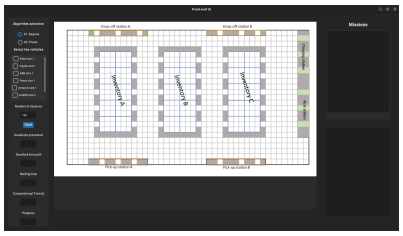


Fig. 4: Graphical User Interface for the Intelligent Transport System designed

2) Novel Deadlock Control Algorithm

A pseudo explanation of the working of the newly selected strategy is presented in IV-A.2.

1) Input of data for simulation. It consists of the number and choice of vehicles in the virtual operational environment, operation/simulation time, and pick-up and drop-off locations of these missions. In this study, a random mission pick-up location and drop-off are initialized in order to check the algorithm performance with changing pick-up and drop-off locations in every run.
2) Each mission is divided into two paths; first from vehicle location to pick-up node and then pick-up to drop-off node. For these paths, a source and a target node are defined. The first path is planned between the first idle vehicle's current location and the pick-up location. The second path is planned with Dijkstra by finding the shortest route from pick-up to drop-off location.
3) Once the mission is executed, it is named as the existing path. While the missions that are planned but yet to be executed are named as new paths.
4) Four check conditions are defined. The existing path is checked with new paths for four conditions to avoid deadlocks. These conditions are explained in detail in later sections. In case these two paths have deadlock(s), the new path is instructed to be re-routed by calling Dijkstra function.
5) In order to avoid the cyclic deadlock by assigning the same deadlock(s) node(s) to the new re-route path, the check conditions make sure to remove the edges between deadlock nodes in order to make them unavailable. This prompts the Dijkstra to find a new shortest route to the new path.
6) Once the checked new path is executed, it becomes the existing path
7) The processes 4 to 5 continue until all the missions are executed with deadlock-free paths.

For deadlock detection, four check conditions; C1, C2, C3, and C4 are developed.

Algorithm 1 Check All Conditions

Function *CheckAllCondition(new mission path, running mission path)*

```
ep ← collect running mission path;
for path ← ep do
    C1 ← CheckSingleMultiNodeCollision;
    C2 ← CheckCommonEdgesInForwardRoute;
    C3 ← CheckCommonEdgesInReverseRoute;
    C4 ← CheckFinalNode;
    if C1 or C2 or C3 or C4 then
        return TRUE;
    return FALSE
```

B. Check condition 1 - Check for single node collision and deadlock

In figure 5, a black vehicle with the grey path is the existing mission executed and blue with the blue path is the new mission path planned and yet to be executed. The algorithm checks this mission path for deadlock detection using check condition 1 2. These paths have a deadlock highlighted in brown region with red marking. In order to avoid this single-node deadlock, a blue vehicle is then instructed to re-route.

In order to check if the node pair (that is node from the new mission path and the existing mission) are same and that

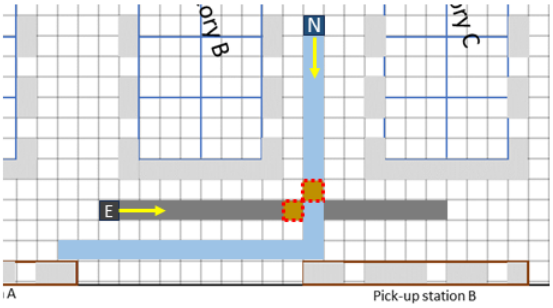


Fig. 5: Single node collision, detection and avoidance of deadlock

Algorithm 2 C1 - Check single node deadlock and avoid deadlock(s)

Function *C1 (new mission path, existing path, new mission path weights, existing path weights)*

```
tolerance ← vehicle size;
for i ← len(min(new mission path,existing path)) do
    n1 ← new mission path(i);
    e1 ← existing path(i);
    nw1 ← new mission path weights(i);
    ew1 ← existing path weights(i);
    if n1 == e1 & abs(nw1 - ew1) < tolerance then
        Remove edge (n1, n1-1)
    if n1 == e1+1 & abs(nw1 - ew1+1) < tolerance then
        Remove edge (n1, n1-1)
    if n1 == e1-1 & abs(nw1 - ew1-1) < tolerance then
        Remove edge (n1, n1-1)
If any edges removed, return TRUE, else FALSE
```

the vehicle reaches this node at the same time by checking the cumulative weights, this check can avoid a collision at that node and a deadlock caused just before that node. Tolerance is defined by the size of the vehicles used in the operational layout to check if these vehicles collide when working in proximity. A node in the new mission path and existing mission path is checked if it is the same.

C. Check condition 2 - Check for overlapping path segments for vehicles traveling in the same direction and avoid deadlock(s)

In figure 6, a black vehicle with the grey path is the existing mission executed, and blue with the blue path is the new mission path planned and yet to be executed. The algorithm checks these mission paths for deadlock detection with check condition 2 3. These paths have the longest common sub-sequence and are traveling in the same direction to reach their targets. Check condition 2 is invoked. In order to avoid deadlock, blue is instructed to re-route.

If the paths have the longest common subsequence and the cumulative weight is the same, then the vehicles are traveling in the same direction and arrive at those

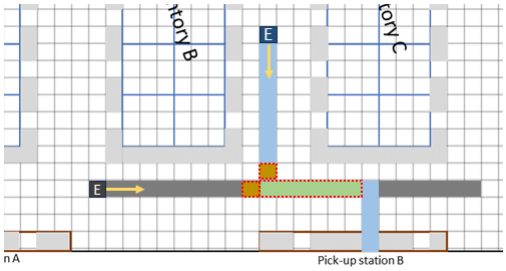


Fig. 6: Overlapping edges in the same direction traveling vehicles, detection and avoidance of deadlock

Algorithm 3 C2 - Check common overlapping edges for both the vehicles traveling in the same direction

Function *C2 (new mission path, existing path, new mission path weights, existing path weights)*

```
lcs = list(longest common sequence(new mission path, existing path));
for cl ← lcs do
    if len(cl) < 2 then
        Skip
    IndexNew = Get index of cl in new mission path;
    IndexEp = Get index of cl in existing path;
    indexNewWeights = new mission path weights(IndexNew);
    indexEpWeights = existing path weights(IndexEp);
    if indexNewWeights overlaps indexEpWeights then
        Remove edges(new mission path(IndexNew))
If any edges removed, return TRUE, else FALSE
```

nodes at the same time which can cause deadlock during their arrival. In this case, we remove those overlapping edges.

D. Check condition 3 - Check for overlapping path segments for vehicles traveling in opposite directions and avoid deadlock(s)

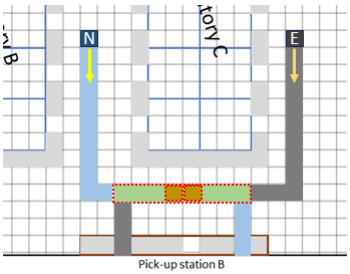


Fig. 7: Overlapping edges in opposite direction traveling vehicles, detection and avoidance of deadlock

In figure 7, a black vehicle with the grey path is the existing mission executed, and blue with a blue path is the new mission path planned and yet to be executed. The algorithms check mission paths for deadlock detection

by checking condition 3 4. These paths have the longest common sub-sequence if checked in reverse for one path and are traveling in opposite directions to reach their targets. Check condition 3 is invoked. In order to avoid deadlock, blue is instructed to re-route.

Algorithm 4 C3 - Check common overlapping edges for both the vehicles traveling in opposite directions

Function *C3 (new mission path, existing path, new mission path weights, existing path weights)*

```
reverse lcs = list(longest common sequence(new mission path(reverse), existing path));
for cl ← lcs do
    if len(cl) < 2 then
        Skip
    IndexNew = Get index of cl(reverse) in new mission path;
    IndexEp = Get index of cl in existing path;
    indexNewWeights = new mission path weights(IndexNew);
    indexEpWeights = existing path weights(IndexEp);
    if indexNewWeights overlaps indexEpWeights then
        Remove edges(new mission path(IndexNew))
If any edges removed, return TRUE, Else FALSE
```

If the paths have the longest common sub-sequence in reverse and the cumulative weight is the same, then the vehicles are traveling in opposite directions and arrive at those nodes at the same time which can cause deadlock during their arrival. In this case, we remove those overlapping edges.

E. Check condition 4 - For unbalanced weights, the last node check is necessary to avoid deadlocks at drop-off points

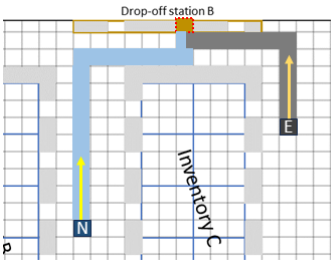


Fig. 8: Check for unbalanced weight layouts and nodes for deadlock and collision avoidance

In figure 8, a black vehicle with the grey path is the existing mission executed, and blue with the blue path is the new mission path planned and yet to be executed. The algorithm checks mission paths for deadlock detection following 5. These paths have a common final node with the same cumulative weights. In order to avoid deadlock, blue is instructed to re-route

Algorithm 5 C4 - In case of unbalanced weights layout, check the final node of two operational vehicle paths

Function *C4 (new mission path, existing path, new mission path weights, existing path weights)*

```
tolerance ← vehicle size;
if (new mission path(last node) == existing path(last node)) And abs(new mission path weights − existing path weights < tolerance) then
    n1 ← new mission path(i);
    e1 ← existing path(i);
    for i ← len(min(new mission path,existing path)) do
        Remove edge (n1, n1-1)
If any edges removed, return TRUE, else FALSE
```

In the case of layouts with unbalanced weights (unequal weights) between nodes, the last node of the existing and new mission paths is checked. This is specifically modeled to provide a safe buffer between vehicles, where one is entering the station point and one is leaving the station point.

After the check of all these conditions, a Boolean value is returned to check if any edges from any of these conditions are removed. If the value is *TRUE*, a re-route is planned to the new mission path. These conditions are checked until a path with no deadlocks is planned.

V. EXPERIMENTS AND RESULTS

A. Implementation

Once the mission paths are input and checked with all the check conditions for deadlock control, the deadlock-free mission is converted into. a JSON file. The mission. JSON file consists of the name of the vehicle assigned, the mission consisting of transverse nodes each with a unique name, a parent, a route with x and y coordinates, actions if any, timeout, and deadline as VDA5050 schemas. This schematic JSON structure is communicated via a communication server to and from a fleet of vehicles and a fleet management system.

B. Model Verification and Validation

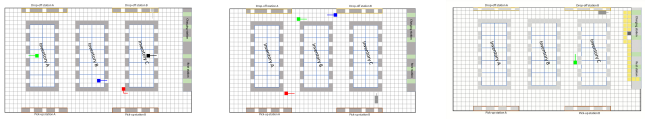
Three case studies with a heterogeneous fleet of AGV operations are considered to verify the working of the developed algorithm. The operational layout is bidirectional. The term 'base' refers to the shape of the vehicle base.

Case study 1 - Square-based service vehicles from different manufacturers. In this case, the vehicles are of the same size and from different manufacturers. The simplest case study is to check if the algorithm successfully detects deadlocks in advance and initiates vehicle re-route or changes the route to avoid those deadlocks. Represented in figure 9a.

Case study 2 - Square and rectangular base service vehicles from different manufacturers. In this case, the vehicles are square AGVs, and rectangular-based (AGV forklifts and container trucks) are to be tested. Increasing the level of difficulty for the deadlock algorithm by simulating different sizes

of vehicles and heterogeneous fleets, i.e., square occupies one node at a time, but rectangular occupies two nodes while traversing in a straight line and 3 while in turn. A sufficient buffer or tolerance for rectangular-size vehicles is coded in the algorithm keeping in mind the nodes covered during turn. To check if these lines of instruction are followed, a visualization is done with case study 2. Represented in figure 9b.

Case study 3 - Square, rectangular base service, and square base cleaning vehicle from different manufacturers. In this case, a cleaning robot (square-base) with a cleaning function and a service vehicle (square-base and rectangular base) that does the duty of pick-up and drop are tested. In a healthcare environment, cleaning bots are used more often than in warehousing services. The cleaning bot cleans one node at a time and can pose a dynamic obstacle for other service vehicles. The algorithm is also modeled to control these dynamic obstacles. To verify this, the case study 3 is drawn. Represented in figure 9c



(a) Case study 1 visualization (b) Case study 2 visualization (c) Case study 3 visualization

Fig. 9: Model Verification with case studies

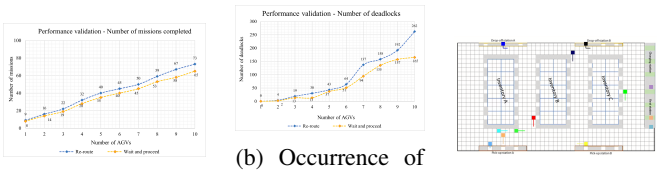
As the operational layout is bidirectional with a well-connected circuit, there is always an alternate path available in case of mission re-route. Hence, the algorithm is successful in detecting deadlocks and avoiding them. As a result, with the verification case studies, no deadlocks were noticed in the visualization window.

1) Validation

For the validation of results objectively, there is a lack of real-world data to validate the outcomes of this algorithm. Furthermore, the environment setting and algorithm structure differ from paper to paper, hence limiting the validation of this algorithm with other scientific papers for objective validation. Nevertheless, conceptual behavior and performance validation are carried out to validate the simulation model.

For performance validation in the simulation experiment, the vehicle speed is set to $1m/s$ and the distance from node to node is considered $1m$. Simulation time is set to 600 secs. In an area of $25mX45m$, the maximum distance traveled by AGVs did not exceed 3600m.

With more vehicles, more missions are executed at the same time, leading to a total distance traveled by vehicles. If less number of vehicles are operating with the same set of simulation times, less number of missions are executed at the same time, leading to increased traveling distances of vehicles. The algorithm shows the same behavior as shown in 10a and thus validates the model. If rerouting is employed, then logically more deadlocks will occur with more vehicles due to traffic congestion caused by vehicle rerouting than wait and proceed strategy. This performance behavior of the algorithm is represented in figure 10b and thus validated. If the number of missions to be completed is less than the number of vehicles, only the required number of vehicles is assigned missions to complete and the remaining vehicles stay idle. This is visually simulated and represented in figure 10c. The number of missions to execute is set to 10 and the number of vehicles input is 13. As the number of missions to execute is 10, only 10 vehicles are given commands to execute, and the rest 3 are idle at the rest/charging stations, and only the required number of vehicles are executing missions.



(a) Number of missions with number of vehicles (b) Occurrence of deadlocks with an increase traffic and congestion (c) Only required number of AGVs to execute missions

Fig. 10: Performance validation of developed deadlock control algorithm

A few aspects of this model are taken into assumption and do not replicate real-world situations. For instance, vehicle batteries are always full and do not require a break in between the operations for charging. No vehicle downtime due to structural and functional breakdowns is considered. Inventories, pick and drop locations have infinite space, and orders flow continuously so there is no waiting time at these locations. The simulations hence result in higher throughput than the realistic. In case the developed model performs better than the other in simulation, it will also follow the same behavior in reality. Therefore, this model with a developed deadlock control algorithm is feasible for this thesis study.

C. Simulation experiments

The developed algorithm is compared with *Wait and proceed* researched in paper [21] approach because both follow the deadlock prediction approach. The difference is the frequency of prediction of deadlock and deadlock avoidance approach. For deadlock avoidance, the waiting strategy is implemented in *wait and proceed* and re-route in the novel deadlock control approach developed in this study.

The performance indicators considered are:

- System throughput (number of missions completed/hr): The total number of orders or missions completed by the system with each algorithm per hour. This also gives insight into the optimal number of vehicles required for the missions.
- Waiting time (s): The time taken by the algorithm to find a deadlock-free mission path and dispatch it to the

vehicle. Waiting time never exceeds computational time. It is the sum of time taken to route planning (rp) and schedule the mission to vehicle(sm). In addition waiting time in *wait and proceed* also accounts for the time the vehicle waits (vw) to avoid deadlock. In brief,

$$\begin{aligned} T_{DDC} &= T_{rp} + T_{sm} \\ T_{WP} &= T_{rp} + T_{sm} + T_{vw} \end{aligned} \tag{1}$$

where DDC is the Developed Deadlock Control of this project and WP is wait and Proceed.

In this study, the detection of deadlock is by prediction approach for both *wait and proceed* and *re-route*, as prediction is required to falsify circular wait deadlock condition. To evaluate the operational performance of the selected strategy, the performance comparison is conducted between *wait and proceed* and *re-route*. As the waiting strategy involves vehicle waiting and the re-route strategy involves more driving time (loaded or unloaded), a fair comparison to evaluate the operational performance considering waiting time is with traffic management efficiency. The vehicle moving time and waiting time are both evaluated to check the efficiency of the algorithm in managing traffic and vehicle fleet coordination as defined in [22]. It is expressed as $\eta = T_{moving}/(T_{moving} + T_{waiting})$. T_{moving} is the moving time of vehicles to complete missions. η for *WP* and *DDC* is expressed in equation 2. For the developed deadlock control re-route strategy, time consumed by the vehicle in re-route is also taken into account. $T_{waiting}$ is T_{DDC} and T_{WP} as defined in equation 1.

$$\begin{aligned} T_{movingDDC} &= T_{moving} + T_{rerouting} \\ T_{movingWP} &= T_{moving} \\ \eta_{DDC} &= T_{movingDDC}/(T_{movingDDC} + T_{DDC}) \\ \eta_{WP} &= T_{movingWP}/(T_{movingWP} + T_{WP}) \end{aligned} \tag{2}$$

D. Operational performance - system throughput

System throughput is defined as the number of missions/orders completed by the system in a given time, here (per hour).

Experiments 1,2 and 3 (similar to case studies 1,2 and 3 set up) are simulated for system throughput. Experiment 1 is only with mobile AGVs, experiment 2 is with mobile and forklift AGVs and Experiment 3 is with mobile, forklift, and cleaning AGVs.

In experiment 1, the output showcased is from 7 AGVs to 19 AGVs, as this range showcased insightful output to compare. In experiment 1 figure 11, the system throughput of developed deadlock control with a re-route approach is increased by an average of 10.46% compared to the wait and proceed approach. That means a developed approach completes 10.46% more missions in an hour than wait and proceed. To notice is almost stagnant behavior of throughput after 17 AGVs in experiment 1 with a re-route approach.

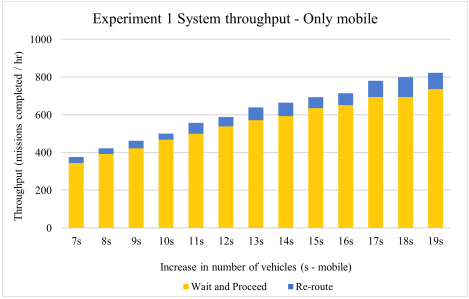


Fig. 11: Experiment 1 - System throughput (missions completed/hr) performance evaluation

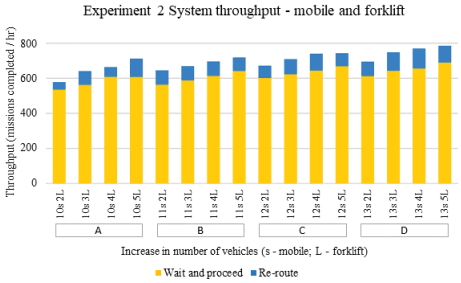


Fig. 12: Experiment 2 - System throughput (missions completed/hr) performance evaluation

In experiment 2, cluster A of 10 mobile AGVs with 2,3,4 and 5 forklift AGVs, cluster B of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs, cluster C of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs, and cluster D of 11 mobile AGVs with 2,3,4 and 5 forklift AGVs. In experiment 2 figure 12, the re-route approach shows increased system throughput by 12.15% in cluster A, 13.56% in cluster B, 13.02% in cluster C, and 15.39% in cluster D compared to wait and proceed. On average, re-route system throughput outperforms wait and proceed by completing 13.53% more missions in an hour in experiment 2 with different size AGVs. In experiment 2 for re-route, stagnant behavior in throughput starts to begin from clusters B, C, and D with 4 forklifts.

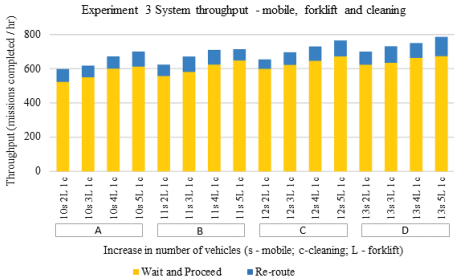
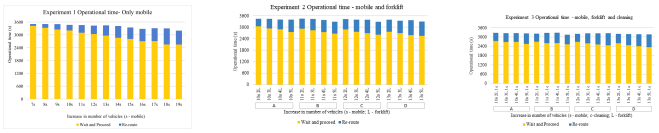


Fig. 13: Experiment 3 - System throughput (missions completed/hr) performance evaluation

Similar to clusters in experiment 3, experiment 4 follows the same clustering with a cleaning robot added to clusters A, B, C, and D. These clusters are chosen as they represent an insightful range of output from experiments. In experiment

3 figure 13, the re-route approach shows increased system throughput by 13.27% in cluster A, 12.87% in cluster B, 11.98% in cluster C, and 14.39% in cluster D compared to wait and proceed. On average, re-route system throughput outperforms wait and proceed by completing 13.12% more missions in an hour in experiment 3 with different sizes and functionalities of AGVs. In experiment 3, cluster B with 4 forklifts shows stagnant behavior in throughput.

The system throughput re-route approach of experiment 3 slightly decreases compared to experiment 2 because experiment 3 introduces a cleaning AGV which traverses node by node with a cleaning mission acting as a dynamic obstacle for other service AGVs causing increased re-route and less operational time.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Fig. 14: Operational time (s) evaluation with re-route and wait and proceed approach

An explanation behind the increase in the number of missions by re-route compared to wait and proceed is explained by efficient operational time represented in figure 14.

Operational time is the total time spent by vehicles to complete missions. This time is $T_{movingDDC}$ for re-route and $T_{movingWP}$ for the wait and proceed approach and is calculated following 2. The total operational time of wait and proceed in experiments 1,2 and 3 is less than re-route as AGVs' time spent to avoid deadlocks by wait and proceed is higher compared to the re-routing time strategy, thus yielding more operational time to complete missions with re-route.

Even though operational time decreases with an increase in AGVs, but increase in AGVs positively affects the increase in system throughput. Furthermore, the re-route algorithm replans the paths to avoid deadlock efficiently in order to increase the operational time for vehicles to complete more missions, thus avoiding infeasible re-routing and inefficient utilization of resources. To prove this point of efficiency of resource utilization, it is explained in section V-E with outputs represented.

E. Operational performance - Traffic management efficiency

In this project, to evaluate the performance of the developed deadlock control algorithm, its operational performance is evaluated against an existing deadlock prediction approach by [21]. The author developed a scientific deadlock control approach for AGVs in ports and terminals with deadlock prediction and avoiding by wait and proceed approach. Traffic management efficiency accounts for vehicle moving time and waiting time and both of these factors are evaluated to check the efficiency of the algorithm in managing traffic as defined in [22], and is used in industrial settings as well.

A common trend in experiments is that the efficiency slightly decreases with a large fleet of AGVs with more congestion and hence traffic control and management complexity increase.

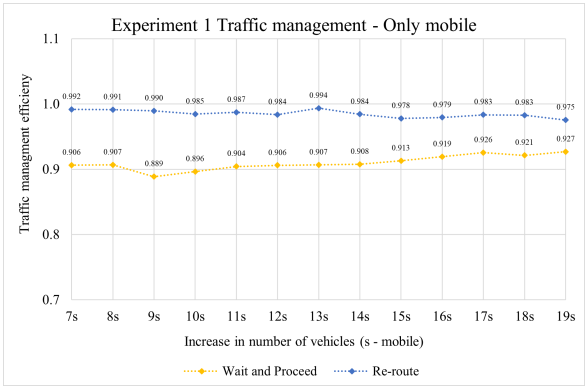


Fig. 15: Experiment 1 - Traffic management efficiency performance evaluation

In experiment 1 output presented in figure 15, the efficiency of the re-route approach averaged is 0.985, and the wait and proceed is 0.91.

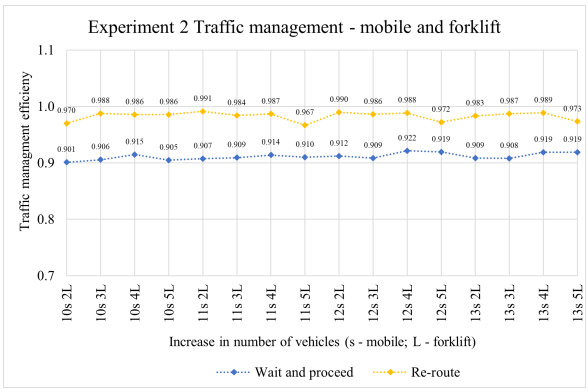


Fig. 16: Experiment 2 - Traffic management efficiency performance evaluation

In experiment 2 output presented in figure 16, re-route efficiency averaged in cluster A is 0.982, cluster B 0.982, cluster C 0.984, and cluster D 0.983, and wait and proceed cluster A is 0.907, cluster B 0.91, cluster C 0.915 and cluster D 0.914.

In experiment 3 output presented in figure 17, re-route efficiency averaged in cluster A is 0.988, cluster B 0.983, cluster C 0.99, and cluster D 0.982, and wait and proceed cluster A is 0.908, cluster B 0.914, cluster C 0.919 and cluster D 0.92.

Re-route shows increased efficiency of 8.24% in experiment 1, 7.81% in experiment 2, and 7.70% in experiment 3 compared to wait and proceed. Within re-route, the efficiency slightly decreases with experiments, as experiments

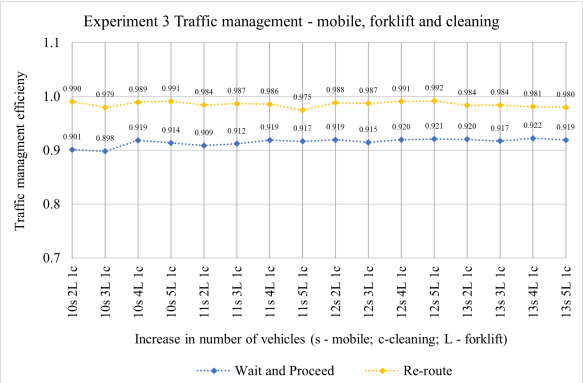


Fig. 17: Experiment 3 - Traffic management efficiency performance evaluation

introduce size change and functionality parameters, hence creating more deadlocks and increased traffic management complexity.

VI. Conclusion and Recommendations

In this research, two research objectives are achieved. The technical research objective is to implement an intelligent transport system with industrial standard communication protocol VDA5050 for a heterogeneous fleet of AGVs. An operational research objective of developing a novel deadlock control algorithm for the intelligent transport system of a heterogeneous fleet of AGVs operating in a collaborative intralogistics environment. This algorithm is communicated to the heterogeneous fleet of AGVs via a communication server with VDA5050. The novel deadlock control detects dynamic deadlocks (cyclic and non-cyclic) using four conditions and employs a re-routing approach to avoid those deadlocks. The modified re-route approach outperforms the wait-and-proceed strategy by increasing the system throughput by 10.46% in experiment 1, 13.53% in experiment 2, and 13.12% in experiment 3. Overall, the results of the experiments show the system can complete 12.37% more missions per hour with less number of vehicles with the developed deadlock control approach. In addition, the traffic management efficiency of the developed algorithm is higher by 7.91% than the wait-and-proceed strategy, which signifies effective deadlock detection, avoidance, and obstacle resolution efficiency of algorithms to complete missions efficiently. With this operational objective is achieved.

For future recommendations, firstly, it is interesting to study the developed approach to different warehouse layouts like a beehive or parallel guide paths, and with different vehicle sizes greater than 2 as used in the study. Secondly, an important point to consider is energy consumption and the time taken for the vehicles to recharge during the operations. Third, collecting and analyzing information of data from sensors to enable local mapping and navigation along with global is recommended. Lastly, to decrease the computational time, it is advised that for future research a more versatile control architecture like distributed or non-centralized should be studied for this approach.

References

[1] Albrecht, T., and G. Ullrich. "Fahrerlose Transportsysteme, Eine Fibel mit Praxisanwendungen zur Technik für die Planung." (2019).

[2] Paweł Tadejko. “Application of Internet of Things in logistics–current challenges”. In: Ekono- mia i Zarządzanie 7.4 (2015), pp. 54–64.

[3] Pan, Fei, and Qiyuan Sun. "A traffic control strategy of the heavy-duty AGVS in a square topology." 2019 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2019.

[4] Müller, Marcel, et al. "Comparison of deadlock handling strategies for different warehouse layouts with an AGVS." 2020 Winter Simulation Conference (WSC). IEEE, 2020.

[5] Sun Chuanyu. “Analysis on the Interaction of both Computer Network and Modern Logistics”. In: E-Busness Journal (2009), pp. 44–45

[6] Mauro Gamberi, Riccardo Manzini, and Alberto Regattieri. “An new approach for the au- tomatic analysis and control of material handling systems: integrated layout flow analy- sis (ILFA)”. In: The international journal of advanced manufacturing technology 41 (2009), pp. 156–167.

[7] KJC Fransen, MA Reniers, and JAWM Van Eekelen. “Deadlock avoidance algorithm for AGVs on a tessellated layout”. In: 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE). IEEE. 2022, pp. 1163–1169.

[8] Emmanuel A. Oyekanlu et al. “A Review of Recent Advances in Automated Guided Vehicle Technologies: Integration Challenges and Research Areas for 5G-Based Smart Manufacturing Applications”. In: IEEE Access 8 (2020), pp. 202312–202353. doi: 10.1109/ACCESS.2020. 3035729.

[9] Walter Quadrini, Elisa Negri, and Luca Fumagalli. “Open interfaces for connecting automated guided vehicles to a fleet management system”. In: Procedia Manufacturing 42 (2020), pp. 406–413.

[10] Ewgenij Gawrilow et al. “Conflict-free vehicle routing: Load balancing and deadlock prevention”. In: EURO journal on transportation and logistics 1.1-2 (2012), pp. 87–111.

[11] Chang W Kim and Jose MA Tanchoco. “Conflict-free shortest-time bidirectional AGV route- ing”. In: The International Journal of Production Research 29.12 (1991), pp. 2377–2391.

[12] Nenad Smolic-Rocak et al. “Time Windows Based Dynamic Routing in Multi-AGV Systems”. In: IEEE Transactions on Automation Science and Engineering 7.1 (2010), pp. 151–155. doi: 10.1109/TASE.2009.2016350.

[13] Yunlong Zhao et al. “Dynamic resource reservation based collision and deadlock prevention for multi-AGVs”. In: IEEE Access 8 (2020), pp. 82120–82130.

[14] Jung-woon Yoo et al. “An algorithm for deadlock avoidance in an AGV System”. In: The international journal of advanced manufacturing technology 26 (2005), pp. 659–668.

[15] Luka Kalinovcic et al. “Modified Banker’s algorithm for scheduling in multi-AGV systems”. In: 2011 IEEE International Conference on Automation Science and Engineering. 2011, pp. 351– 356. doi: 10.1109/CASE.2011.6042433

[16] Daiyu He et al. “Deadlock Avoidance in Closed Guide-Path Based MultiAGV Systems”. In: IEEE Transactions on Automation Science and Engineering (2022).

[17] Ennio Anthonius Thijssen et al. “MQTT based Communication Framework for AGVs in a Digital Twin”. In: Manufacturing Systems (2021).

[18] Keisuke Okumura and Sébastien Tixeuil. “Fault-Tolerant Offline Multi-Agent Path Planning”. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. 10. 2023, pp. 11647–11654.

[19] Kang Min Kim, Chang Hyun Chung, and Young Jae Jang. “Deadlock Avoidance Dynamic Routing Algorithm for a Massive Bidirectional Automated Guided Vehicle System”. In: 2022 Winter Simulation Conference (WSC). IEEE. 2022, pp. 1–12.

[20] Knapp, Edgar. "Deadlock detection in distributed databases." ACM Computing Surveys (CSUR) 19.4 (1987): 303-328.

[21] Moorthy, Rajeeva Lochana, et al. "Cyclic deadlock prediction and avoidance for zone-controlled AGV system." International Journal of Production Economics 83.3 (2003): 309-324.

[22] Pratissoli, Federico, et al. "Hierarchical Traffic Management of Multi-AGV Systems With Deadlock Prevention Applied to Industrial Environments." IEEE Transactions on Automation Science and Engineering (2023).

9
Appendix B

9.0.1. MQTT connection, security and Quality of Service(QoS!) in networking

MQTT provides an option of setting a last comment/message for client in case if the client disconnects unexpectedly due to network issues. This last comment/message is distribute by broker to other subscribes. This feature comes under topic *connection*. In case if vehicle disconnects from broker, it stores all the order information and executes its task till the target node.
For protocol security, the broker configuration can be modified. **QoS!** is use of technology (here MQTT) to ensure the performance of application critical in nature (here, topics and sub-topics) with limited network bandwidth/capacity. MQTT QoS level 0 (best effort) for critical topics like *order*, *state*, *factsheet*. For topic like *connection*, MQTT QoS level 1 (at least once) can be used. This study assumes no loss of connection during operation.

9.0.2. Subtopics for communication

Subtopic name	**Published by**	**Subscribed by**	**Used for**	**Implementation**	**Schema**
order	Mission dispatch module	Robot client	Communication of executing mission orders from FMS to vehicle	mandatory	order.schema
instantActions	Mission dispatch module	Robot client	Communication of actions that are needed to be immediately executed	mandatory (not considered)	instantActions.schema
state	Robot client	Mission dispatch module	Communication of state of vehicle	mandatory	state.schema
visualization (not considered)	Robot client	visualization system module	High frequency of position for visualization only	optional	visualization.schema
connection	MQTT broker/ robot client	Mission dispatch module	Indicates connection loss of vehicle. Added for MQTT protocol connection check	mandatory (not considered)	connection.schema
factsheet	Robot client	Mission dispatch module	Vehicle setup in FMS module	madatory	factsheet.schema

2023.MME.8870

9.0.3. Topic *order*

FMS sends a JSON encapsulated order to the vehicle via MQTT broker as topic.

9.0.3.1. Maps

In order to ensure a consistent navigation amount heterogeneous fleet of vehicles, the position is specified with reference to local map coordinate system as represented in figure 9.1. The map co-ordinate system is right-handed, with z-axis pointing upwards in sky. A positive rotation is counterclockwise. Vehicle co-ordinate is right-handed, with x-axis pointing forward direction. The x, y and z co-ordinates are in metres, orientation in radians and within $+pi$ to $-pi$. These are set in accordance with standard DIN ISO 8855 standard [28].

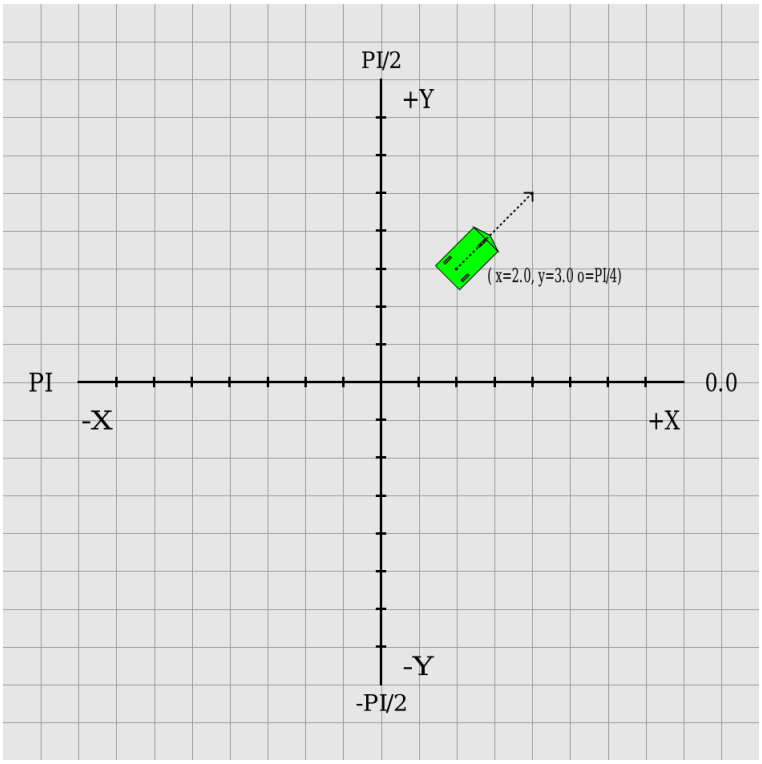


Figure 9.1: Coordination system with sample vehicle according to standard VDA5050

9.0.4. Topic *order*: implementation of order message

Table 9.1 showcases object structure used in this study of order message. For more structures, one can refer to VDA5050 manual.

Object structure	**Unit**	**Data-type**	**Description**
hearderId		uint32	HeaderId of the order message
timestamp		string	Time stamp (ISO 8601, UTC) YYYY-MM-DD THH-mm-ss.ssZ
version		string	Version of the protocol
manufacturer		string	Manufacturer of the vehicle
serial number		string	Vehicle serial number
orderId		string	Order identification. This is used to identify multiple order messages that belong to the same orderId. In case vehicle losses order message, it tracks back the orderId
node position{		JSON-object	Defines location of the nodes on the configuration space/map. All maps have same specific global origin
x	m	float64	x-position on the configuration space/map
y	m	float64	y-position on the configuration space/map
theta	rad	float64	Absolute orientation on the node
mapId}		string	Unique identification of the maps. For ex, map of operational floor, map of upper floor, etc
action{		JSON-object	Decribes action that vehicle is instructed to execute
actionType}		string	Name of the action

Table 9.1: Topic *order* JSON encapsulated message contents

9.0.5. Actions

Vehicle supporting actions other than driving, these actions are via the *action* field with either node or edge, or sent via instantActions sub-topic. If actions are instructed to be executed on edge, then those actions must run only when vehicle is on the edge. In case of actions on node, they can run as long as they need to run and should be self-terminating (ex pick action, drop action, etc). The thesis do not order any specific actions. There are predefined actions that reader can refer from VDA5050 manual for more complex implementation, also instantActions.

9.0.6. Topic *state*

The states of the vehicle are transferred as topic *state*, which is only one topic. Instead of separate messages like orders, battery-state or any errors, having one topic is beneficial to reduce workload of broker and FMS for handling messages and also keeping the vehicle state information synchronized. Vehicle state is published whenever subscribed by the client, or every 30s via broker to FMS. state message are transmitted during events:

- Receiving an order
- Order update
- Errors or warnings
- Traversing over nodes assigned

Whenever a vehicle transverses through those nodes and edges, the mission states are updated with *nodeStates* and *edgeStates* of those corresponding edges and nodes. Topic *state* exchange of messages with JSON format is explained table 9.2.

Field	Data-type	Unit	Description
headerID	uint32		Header Id of the message
timestamp	string		Timestamp (ISO 8601, UTC) YYYY-MM-DD THH:mm:ss.ssZ
version	string		Protocol version
manufacturer	string		Vehicle manufacturer
serialNumber	string		Vehicle serial number
orderId	string		Unique identification number of the current order or completed order. It is kept until a new orderId is received by the vehicle
agvPosition	JSON-object		Current position of the vehicle on map
actionStates [actionState]	array		List of actions both current and yet to be completed The action is state is preserved until vehicle receives new order
operatingMode	string		Enum{AUTOMATIC, SEMIAUTOMATIC, MANUAL, SERVICE,TEACHIN} explained in table
agvPosition{	JSON-object		Defines position or location on a map or configuration space. Each floor has its own space/map representation
x	float64	m	x-position on the configuration space/map with respect to map co-ordiante system
y	float64	m	y-position on the configuration space/map with respect to map co-ordiante system
theta	float64	rad	Vehicle orientation Range: [-Pi..+Pi]
mapId	string		Unique id for map/configuration space. Differs from floor to floor
batteryState{	JSON-object		
batteryCharge	float64	%	Vehicle battery state of charge
charging }	boolean		true: Charging in progress false: Vehicle is currently not charging
error{	JSON-object		
errorLevel}	string		Enum [WARNING, FATAL] WARNING: Maintenance required or Field violation. No operator intervention required FATAL: Vehicle enable to run. Operator intervention is asked. For ex, sensor malfunction, deadlock situation

Table 9.2: Topic *state* JSON encapsulated message contents

9.0.6.1. Errors or warnings

There are two levels of errors: WARNING and FATAL. WARNING is self-resolving error (operational field violation) while FATAL requires human intervention, for ex if vehicles are in deadlock

situation.

9.0.6.2. operatingMode Enum description

Identifier	Description
AUTOMATIC (Used in this study)	Vehicle is under complete control of the FMS. Vehicle executes the order messages instructed by FMS
SEMIAUTOMATIC	Vehicle is under complete control of the FMS. Vehicle executes the order messages instructed by FMS. It is different from above identifier in a way where Human-Machine interface controls the driving speed
MANUAL	Vehicle is NOT under control of the FMS. FMS does not send driving order or actions to the vehicle. Human-Machine interface (HMI) is used to steer, control and vehicle handling.
SERVICE	Vehicle is NOT under control of the FMS. FMS does not send driving order or actions to the vehicle. Authorized operator can reconfigure the vehicle
TEACHIN	Vehicle is NOT under control of the FMS. FMS does not send driving order or actions to the vehicle. Vehicle is in learning phase

Table 9.3: Description of *operatingMode* under topic *state*

9.0.6.3. actionStates

actionState is used to represent the action whenever a vehicle receives it. *actionStates* are in array format. *actionState* uses *actionStatus* to represent the stage of the lifecycle of that action. The table is represented in 9.4

actionStatus	**Description**
WAITING	The vehicle has received the action, but not the node location where it is supposed to trigger the action
INITIALIZING	Action is trigerred
RUNNING	The action is running
PAUSED	The action is paused, either due to pause by instantAction topic or by external trigger like pause button
FINISHED	The action is finished and result is reported via resultDescription
FAILED	Action could not be completed

Table 9.4: actionStatus for the lifecycle stage check of actionStates

9.0.7. Topic *connection*

A last will message or comment can be set by the client on each vehicle. This message with topic *connection* is published by broker in case of disconnection with vehicle. In this way, FMS discovers the *connection* status by subscribing to the topic via broker. The interval to exchange this message is configurable in broker and default set to 15 seconds. The disconnection is detected via a heartbreak exchanged between broker and pub/sub client. The **QoS!** is level 1 (at least once). In case if connection has been ended by user by initiating a MQTT disconnection command, the last will message is not shared. This last will message is in JSON encapsulated message with fields represented in table 9.5

Field	**Data-type**	**Description**
headerID	uint32	Header Id of the message
timestamp	string	Timestamp (ISO 8601, UTC) YYYY-MM-DD THH:mm:ss.ssZ
version	string	Protocol version
manufacturer	string	Vehicle manufacturer
serialNumber	string	Vehicle serial number
connectionState	string	Enum {ONLINE, OFFLINE, CONNECTIONBROKEN} ONLINE: Connection between vehicle and broker is active OFFLINE: Connection between vehicle and broker is offline in coordinated way. CONNECTIONBROKEN: Connection between vehicle and broker has unexpectedly broken and ended

Table 9.5: Topic *connection* JSON encapsulated message contents

9.0.8. Topic *factsheet*

As the name says, it provides specific (technical and functional) information about the vehicles in the fleet. This *factsheet* can be used to optimize the vehicle operations, simulation and planning as the simulation can check the performance of different vehicles. In case of heterogeneous vehicles, the *factsheet* include the type of communication interfaces of these vehicles which is then used to integrate the vehicle type series into the VDA5050-compliant FMS.
factsheet is in JSON format in order for it to be both human-readable and for machine processing. The *factsheet* is requeseted by FMS also through instant action message *factsheetRequest*

9.0.8.1. *factsheet* JSON structure

Field	Data-type	Description
headerID	uint32	Header Id of the message
timestamp	string	Timestamp (ISO 8601, UTC) YYYY-MM-DD THH:mm:ss.ssZ
version	string	Protocol version
manufacturer	string	Vehicle manufacturer
serialNumber	string	Vehicle serial number
typeSpecification	JSON-object	Defines specific class and the vehicle capabilities
physicalParameters	JSON-object	Defines basic physical properties of teh vehicle
protocolLimits	JSON-object	Limits defined for the length of identifiers, arrays, strings and similar in MQTT connection
protocolFeatures	JSON-object	Features of VDA5050 protocol that are supported
agvGeometry	JSON-object	In depth definition of geometry of the vehicle
loadSpecification	JSON-object	Specification of load capabilities of vehicle
localizationParameters	JSON-object	In depth specification of localization

Table 9.6: Topic *factsheet* JSON encapsulated message contents

9.0.8.2. physicalParameters JSON-object

This JSON-object defines physical properties of the vehicle explain in table 9.7.

Field	Data-type	Description
speedMin	float64	[m/s] Min controlled speed (continuous) of the vehicle
speedMax	float64	[m/s] Max speed of the vehicle
accelerationMax	float64	[m/sq.s] Max acceleration defined at max load of the vehicle
decelerationMax	float64	[m/sq.s] Max deceleration defined at max load of the vehicle
heightMin	float64	[m] Minimum height of the vehicle
heightMax	float64	[m] Maximum height og the vehicle
width	float64	[m] Width of the vehicle
length	float64	[m] Length of the vehicle

Table 9.7: JSON-object structure for physical properties of the vehicle

9.0.8.3. typeSpecification JSON-object

This JSON-object defines general properties of the vehicle explain in table 9.8

Field	Data-type	Description
seriesName	string	Generalized series name as specified by vehicle manufacturer
seriesDescription	string	Human readable vehicle type series description in free text format
agvKinematic	string	Simplified vehicle kinematics-type description [OMNI, DIFF, THREEWHEEL] OMNI: Omni-directional vehicle DIFF: Differential drive THREEWHEEL: three-wheel driven vehicle or with similar kinematics
agvClass	string	Simplified description of functional vehicle class [FORKLIFT, CONVEYOR, TUGGER, CARRIER] FORKLIFT: forklift functionality CONVEYOR: Vehicle with conveyors on it TUGGER: tugger functionality CARRIER: Load carrier either with or without any lifting unit
maxLoadMass	float64	[kg], max load on vehicle
localizationTypes	array of string	Localization type description [NATURAL, REFLECTOR, RFID, DMC, SPOT, GRID] NATURAL: Localization with natural landmarks REFLECTOR: Laser reflector RFID: RFID-based tags DMC: Data matrix code (QR-code or bar code) SPOT: Magnetic tapes/ spots GRID: Magnetic grid
navigationTypes	array of string	Path planning types supported by vehicle. GUIDED is the path sent by FMS PHYSICAL_LINE_GUIDED: No path planning, vehicle follows installed physical path VIRTUAL_LINE_GUIDED: Vehicle navigates on fixed (virtual) paths AUTONOMOUS: Used in local path planning, vehicle plans autonomous path with onboard equipment

Table 9.8: JSON-object structure for general properties of the vehicle

Each JSON-object in table 9.6 are discussed in tables below. The ones applicable/used in this thesis are presented. Remaining JSON-object of *factsheet* can be referred from VDA5050 manual for more integrated vehicle communication system.

Scientific references

[1] Matei Alexandru, Circa Dragoș, and Zamfirescu Bălă-Constantin. "Digital Twin for automated guided vehicles fleet management". In: *Procedia Computer Science* 199 (2022), pp. 1363–1369.

[2] Sreenatha G Anavatti, Sobers LX Francis, and Matthew Garratt. "Path-planning modules for Autonomous Vehicles: Current status and challenges". In: *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*. 2015, pp. 205–214. DOI: 10.1109/ICAMIMIA.2015.7508033.

[3] Ly Gia Bao, Truong Giang Dang, and Nguyen Duy Anh. "Storage assignment policy and route planning of agvs in warehouse optimization". In: *2019 International Conference on System Science and Engineering (ICSSE)*. IEEE. 2019, pp. 599–604.

[4] Nikolaos Baras and Minas Dasygenis. "An algorithm for routing heterogeneous vehicles in robotized warehouses". In: *2019 Panhellenic Conference on Electronics & Telecommunications (PACET)*. IEEE. 2019, pp. 1–4.

[5] Muhammad Bashir et al. "Optimal enforcement of liveness to flexible manufacturing systems modeled with Petri nets via transition-based controllers". In: *Advances in Mechanical Engineering* 10.1 (2018), p. 1687814017750707.

[6] Maximilian Berndt et al. "Centralized Robotic Fleet Coordination and Control". In: *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*. 2021, pp. 1–8.

[7] Maximilian Berndt et al. "Deterministic Planning for Flexible Intralogistics". In: *2021 26th International Conference on Automation and Computing (ICAC)*. IEEE. 2021, pp. 1–6.

[8] Alessio Botta et al. "Integration of cloud computing and internet of things: a survey". In: *Future generation computer systems* 56 (2016), pp. 684–700.

[9] Xinyu Chen et al. "An ETCEN-Based Motion Coordination Strategy Avoiding Active and Passive Deadlocks for Multi-AGV System". In: *IEEE Transactions on Automation Science and Engineering* 20.2 (2022), pp. 1364–1377.

[10] Sun Chuanyu. "Analysis on the Interaction of both Computer Network and Modern Logistics". In: *E-Busness Journal* (2009), pp. 44–45.

[11] Edward G Coffman, Melanie Elphick, and Arie Shoshani. "System deadlocks". In: *ACM Computing Surveys (CSUR)* 3.2 (1971), pp. 67–78.

[12] Matthias De Ryck, Mark Versteyhe, and Frederik Debrouwere. "Automated guided vehicle systems, state-of-the-art control algorithms and techniques". In: *Journal of Manufacturing Systems* 54 (2020), pp. 152–173.

[13] Kshipra Dixit and Ajay Khuteta. "A Dynamic and Improved Implementation of Banker's Algorithm". In: *International Journal on Recent and Innovation Trends in Computing and Communication* 5.8 (2017), pp. 45–49.

[14] Mariagrazia Dotoli and Maria Pia Fanti. "Deadlock Detection and Avoidance Strategies for Automated Storage and Retrieval Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.4 (2007), pp. 541–552. DOI: 10.1109/TSMCC.2007.897690.

[15] Ivica Draganjac et al. “Decentralized control of multi-AGV systems in autonomous warehousing applications”. In: *IEEE Transactions on Automation Science and Engineering* 13.4 (2016), pp. 1433–1447.

[16] KJC Fransen, MA Reniers, and JAWM Van Eekelen. “Deadlock avoidance algorithm for AGVs on a tessellated layout”. In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2022, pp. 1163–1169.

[17] KJC Fransen et al. “A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems”. In: *Computers & Operations Research* 123 (2020), p. 105046.

[18] Mauro Gamberi, Riccardo Manzini, and Alberto Regattieri. “An new approach for the automatic analysis and control of material handling systems: integrated layout flow analysis (ILFA)”. In: *The international journal of advanced manufacturing technology* 41 (2009), pp. 156–167.

[19] Ewgenij Gawrilow et al. “Conflict-free vehicle routing: Load balancing and deadlock prevention”. In: *EURO journal on transportation and logistics* 1.1-2 (2012), pp. 87–111.

[20] Lixiao Guo, Qiang Yang, and Wenjun Yan. “Intelligent path planning for automated guided vehicles system based on topological map”. In: *2012 IEEE Conference on Control, Systems Industrial Informatics*. 2012, pp. 69–74. DOI: 10.1109/CCSII.2012.6470476.

[21] Zhengang Guo et al. “A timed colored petri net simulation-based self-adaptive collaboration method for production-logistics systems”. In: *Applied Sciences* 7.3 (2017), p. 235.

[22] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[23] James W. Havender. “Avoiding deadlock in multitasking systems”. In: *IBM systems journal* 7.2 (1968), pp. 74–84.

[24] Daiyu He et al. “Deadlock Avoidance in Closed Guide-Path Based MultiAGV Systems”. In: *IEEE Transactions on Automation Science and Engineering* (2022).

[25] Gaston C Hillar. *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[26] Kitae Hwang, Jae Moon Lee, and In Hwan Jung. “Performance monitoring of MQTT-based messaging server and system”. In: *Journal of Logistics, Informatics and Service Science* 9.1 (2022), pp. 85–96.

[27] *Isaac Mission Dispatch*. 2022. URL: https://github.com/NVIDIA-ISAAC/isaac_mission_dispatch#isaac-mission-dispatch/.

[28] *ISO 8855:2011: Road vehicles — Vehicle dynamics and road-holding ability*. 2018. URL: https://www.iso.org/standard/51180.html/.

[29] Luka Kalinovcic et al. “Modified Banker’s algorithm for scheduling in multi-AGV systems”. In: *2011 IEEE International Conference on Automation Science and Engineering*. 2011, pp. 351–356. DOI: 10.1109/CASE.2011.6042433.

[30] Karthik Karur et al. “A survey of path planning algorithms for mobile robots”. In: *Vehicles* 3.3 (2021), pp. 448–468.

[31] Lea Kaven et al. “Digital Twin Pipeline zur VDA-5050-Integration: Ein standardisierter Ansatz für die automatisierte Implementierung der flexiblen Produktion”. In: *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 117.5 (2022), pp. 327–331.

[32] Chang W Kim and Jose MA Tanchoco. "Conflict-free shortest-time bidirectional AGV routeing". In: *The International Journal of Production Research* 29.12 (1991), pp. 2377–2391.

[33] Kang Min Kim, Chang Hyun Chung, and Young Jae Jang. "Deadlock Avoidance Dynamic Routing Algorithm for a Massive Bidirectional Automated Guided Vehicle System". In: *2022 Winter Simulation Conference (WSC)*. IEEE. 2022, pp. 1–12.

[34] Kap Hwan Kim, Su Min Jeon, and Kwang Ryel Ryu. "Deadlock prevention for automated guided vehicles in automated container terminals". In: *Container Terminals and Cargo Systems: Design, Operations Management, and Logistics Control Issues* (2007), pp. 243–263.

[35] Edgar Knapp. "Deadlock detection in distributed databases". In: *ACM Computing Surveys (CSUR)* 19.4 (1987), pp. 303–328.

[36] Jean-Paul Laumond et al. *Robot motion planning and control*. Vol. 229. Springer, 1998.

[37] Ying Tat Leung and Gwo-Ji Sheen. "Resolving deadlocks in flexible manufacturing cells". In: *Journal of manufacturing systems* 12.4 (1993), pp. 291–304.

[38] Qing Rebecca Lia, Zachary Dydek, and Daniel Theobald. "Why interoperability is critical to the warehouse of the future". In: *ISR Europe 2022; 54th International Symposium on Robotics*. VDE. 2022, pp. 1–7.

[39] Shiwei Lin et al. "A review of path-planning approaches for multiple mobile robots". In: *Machines* 10.9 (2022), p. 773.

[40] Gaiyun Liu, Yuting Liu, and Zhiwu Li. "Robust liveness-enforcing supervisor for Petri nets with unreliable resources based on mixed integer programming". In: *Soft Computing* 26.8 (2022), pp. 4019–4032.

[41] Waldemar Małopolski. "A sustainable and conflict-free operation of AGVs in a square topology". In: *Computers & Industrial Engineering* 126 (2018), pp. 472–481.

[42] Julius B Mathews et al. "Industrial applications of a modular software architecture for line-less assembly systems based on interoperable digital twins". In: *Frontiers in Mechanical Engineering* 9 (2023), p. 1113933.

[43] Damjan Miklić et al. "A modular control system for warehouse automation-algorithms and simulations in USARSim". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3449–3454.

[44] László Monostori. "Cyber-physical production systems: Roots, expectations and R&D challenges". In: *Procedia Cirp* 17 (2014), pp. 9–13.

[45] Rajeeva Lochana Moorthy et al. "Cyclic deadlock prediction and avoidance for zone-controlled AGV system". In: *International Journal of Production Economics* 83.3 (2003), pp. 309–324.

[46] Keisuke Okumura and Sébastien Tixeuil. "Fault-Tolerant Offline Multi-Agent Path Planning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 10. 2023, pp. 11647–11654.

[47] Emmanuel A. Oyekanlu et al. "A Review of Recent Advances in Automated Guided Vehicle Technologies: Integration Challenges and Research Areas for 5G-Based Smart Manufacturing Applications". In: *IEEE Access* 8 (2020), pp. 202312–202353. DOI: 10.1109/ACCESS.2020.3035729.

[48] Federico Pratissoli et al. "Hierarchical Traffic Management of Multi-AGV Systems With Deadlock Prevention Applied to Industrial Environments". In: *IEEE Transactions on Automation Science and Engineering* (2023).

[49] Mingyao Qi et al. “On the evaluation of AGVS-based warehouse operation performance”. In: *Simulation Modelling Practice and Theory* 87 (2018), pp. 379–394.

[50] Guo Qing, Zhang Zheng, and Xu Yue. “Path-planning of automated guided vehicle based on improved Dijkstra algorithm”. In: *2017 29th Chinese control and decision conference (CCDC)*. IEEE. 2017, pp. 7138–7143.

[51] Guo Qing, Zhang Zheng, and Xu Yue. “Path-planning of automated guided vehicle based on improved Dijkstra algorithm”. In: *2017 29th Chinese Control And Decision Conference (CCDC)*. 2017, pp. 7138–7143. DOI: 10.1109/CCDC.2017.7978471.

[52] Walter Quadrini, Elisa Negri, and Luca Fumagalli. “Open interfaces for connecting automated guided vehicles to a fleet management system”. In: *Procedia Manufacturing* 42 (2020), pp. 406–413.

[53] Nadim Rana et al. “Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments”. In: *Neural Computing and Applications* 32 (2020), pp. 16245–16277.

[54] Robert G. Sargent. “Verification and validation of simulation models”. In: *Proceedings of the 2010 Winter Simulation Conference*. 2010, pp. 166–183. DOI: 10.1109/WSC.2010.5679166.

[55] Marija Seder et al. “Open platform based mobile robot control for automation in manufacturing logistics”. In: *IFAC-PapersOnLine* 52.22 (2019), pp. 95–100.

[56] Arie Shoshani and EG Coffman. “Sequencing tasks in multiprocess systems to avoid deadlocks”. In: *11th Annual Symposium on Switching and Automata Theory (swat 1970)*. IEEE. 1970, pp. 225–235.

[57] Nenad Smolic-Rocak et al. “Time Windows Based Dynamic Routing in Multi-AGV Systems”. In: *IEEE Transactions on Automation Science and Engineering* 7.1 (2010), pp. 151–155. DOI: 10.1109/TASE.2009.2016350.

[58] Kaarthik Sundar, Saravanan Venkatachalam, and Satyanarayana G. Manyam. “Path planning for multiple heterogeneous Unmanned Vehicles with uncertain service times”. In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 480–487. DOI: 10.1109/ICUAS.2017.7991336.

[59] Paweł Tadejko. “Application of Internet of Things in logistics–current challenges”. In: *Ekonomia i Zarządzanie* 7.4 (2015), pp. 54–64.

[60] Ennio Anthonius Thijssen et al. “MQTT based Communication Framework for AGVs in a Digital Twin”. In: *Manufacturing Systems* (2021).

[61] Emeka E Ugwuanyi, Muddesar Iqbal, and Tasos Dagiuklas. “A comparative analysis of deadlock avoidance and prevention algorithms for resource provisioning in intelligent autonomous transport systems over 6G infrastructure”. In: *IEEE Transactions on Intelligent Transportation Systems* (2022).

[62] *VDA 5050 explained*. 2021. URL: https://ifr.org/post/vda-5050-explained/.

[63] Iris FA Vis. “Survey of research in the design and control of automated guided vehicle systems”. In: *European journal of operational research* 170.3 (2006), pp. 677–709.

[64] Mirte van Weert. “Deadlock avoidance and detection for the grid-based AGV-sorter system”. PhD thesis. Master’s thesis, Eindhoven University of Technology, 2019.

[65] Irwin Welber. “Factory of the Future”. In: *IEEE Control Systems Magazine* 7.2 (1987), pp. 20–22.

[66] *What is VDA 5050?* 2023. URL: https://navitecsystems.com/blog-post/what-is-vda-5050/.

[67] *Why IoT changes everything.* 2022. URL: https://www.ericsson.com/en/internet-of-things.

[68] Haining Xiao et al. "A Collision and Deadlock Prevention Method With Traffic Sequence Optimization Strategy for UGN-Based AGVS". In: *IEEE Access* 8 (2020), pp. 209452–209470. DOI: 10.1109/ACCESS.2020.3039515.

[69] Haining Xiao et al. "A collision and deadlock prevention method with traffic sequence optimization strategy for UGN-based AGVs". In: *IEEE Access* 8 (2020), pp. 209452–209470.

[70] Fengjia Yao et al. "Improving just-in-time delivery performance of IoT-enabled flexible manufacturing systems with AGV based material transportation". In: *Sensors* 20.21 (2020), p. 6333.

[71] Xianfeng Ye et al. "An Agent-based System Architecture for Automated Guided Vehicles in Cloud-Edge Computing Environments". In: *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 2023, pp. 23–28. DOI: 10.1109/CSCWD57460.2023.10152739.

[72] Jung-woon Yoo et al. "An algorithm for deadlock avoidance in an AGV System". In: *The international journal of advanced manufacturing technology* 26 (2005), pp. 659–668.

[73] Vasileios Zeimpekis, George M. Giaglis, and Ioannis Minis. "Development and Evaluation of an Intelligent Fleet Management System for City Logistics". In: *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. 2008, pp. 72–72. DOI: 10.1109/HICSS.2008.121.

[74] Yunlong Zhao et al. "Dynamic resource reservation based collision and deadlock prevention for multi-AGVs". In: *IEEE Access* 8 (2020), pp. 82120–82130.

Technological news articles and blogs

[27] *Isaac Mission Dispatch*. 2022. URL: https://github.com/NVIDIA-ISAAC/isaac_mission_dispatch#isaac-mission-dispatch/.

[28] *ISO 8855:2011: Road vehicles — Vehicle dynamics and road-holding ability*. 2018. URL: https://www.iso.org/standard/51180.html/.

[62] *VDA 5050 explained*. 2021. URL: https://ifr.org/post/vda-5050-explained/.

[66] *What is VDA 5050?* 2023. URL: https://navitecsystems.com/blog-post/what-is-vda-5050/.

[67] *Why IoT changes everything*. 2022. URL: https://www.ericsson.com/en/internet-of-things.

Books only

[25] Gaston C Hillar. *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[36] Jean-Paul Laumond et al. *Robot motion planning and control*. Vol. 229. Springer, 1998.