

Learning-based path planning for automatic guided vehicles in container terminals

A case study at TBA Group

P. Wijnands

Master Thesis
Date: July, 2022



Learning-based path planning for automatic guided vehicles in container terminals

A case study at TBA Group

by

P. Wijnands

Master Thesis

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

to be defended publicly on Thursday July 23, 2022 at 10:15 AM

Report number: 2022.MME.8666
Student number: 4343840
Thesis committee: Prof. dr. R. R. Negenborn, TU Delft, committee Chair, 3ME Delft
Dr. F. Schulte, TU Delft, 3ME Delft
M. Bokkers MSc, TBA Group

This thesis is confidential and cannot be made public until

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

Before you lies the master thesis 'Learning-based path planning for automatic guided vehicles in container terminals'. This thesis was written to obtain the degree of Master of Science at the Delft University of Technology in Mechanical Engineering with a specialization in Transport Engineering & Logistics. This research was conducted to contribute to the rapidly evolving topic of machine learning. My goal was to see whether this topic could be applied to benefit the path planning in container terminals. I would like to thank TBA Group for providing me with the opportunity for this research. Especially Adriaan ter Mors and Menno Bokkers for their discussions and guidance. I also would like to thank the chair of the thesis committee, Rudy Negenborn, for his supervision over the research and his input during our meetings. Last but not least, I would like to thank Frederik Schulte as daily supervisor for his reviews and discussions.

P. Wijnands
Delft, July 2022

Contents

Acronyms	4
List of symbols	6
1 Introduction	7
1.1 Problem statement	8
1.2 The research objective	8
1.3 Research questions	8
1.4 Research scope	9
1.5 Research structure and methodology	11
2 Fundamentals and practical Case	13
2.1 Research context: TBA Group	13
2.1.1 Implementation	14
2.2 Research context: Machine learning	15
2.2.1 Why would one apply machine learning?	15
2.2.2 Types of machine learning	15
2.3 Research context: Path Planning	16
2.4 Practical case	16
3 Literature Review	17
3.1 Review methodology	17
3.2 Sub question 1: Path planning algorithms.	17
3.2.1 What are Path Planning algorithms?	18
3.2.2 The state of the art in path planning in automated container terminals	19
3.2.3 The challenges in path planning in automated container terminals	21
3.3 Sub question 2: reinforcement learning for path planning	21
3.3.1 Sub question 2a: What is reinforcement learning?	21
3.3.2 Types of reinforcement learning algorithms	23
3.3.3 Sub question 2b: Which reinforcement learning algorithms are applied in path planning?	24
3.3.4 Conclusion: What gap is found in the literature regarding reinforcement learning applied to Path Planning?	26
4 Experimental Setup	27
4.1 The functioning of the container terminal emulator by TBA	27
4.1.1 Choice of highway	27
4.1.2 Order generator	27
4.2 What data from the automated container terminal emulator can be used for the algorithm?	28
4.3 What are the key performance indicators?	28
4.4 How will the scientific academic innovation be determined?	29
4.5 How will the value for TBA be determined?	29
4.5.1 Challenging situations in the container terminals	29
4.5.2 The experiments	30
4.6 What RL methodology suits best for this case?	32
4.6.1 RL algorithms used for path planning	32
4.6.2 Reinforcement Learning aspects	33
4.6.3 This case: learning-based path planning for AGVs in container terminals	33
4.7 Conclusion	34

5	A Markov Decision Process	35
5.1	Markov Decision Process	35
5.2	Environment interaction	36
5.3	Markov Decision Process definition for the case	36
5.3.1	States	36
5.3.2	Actions.	37
5.3.3	Reward.	38
5.3.4	The value of the action: the Q value	38
6	A Reinforcement Learning Based A* Algorithm	39
6.1	Initialize the algorithm	40
6.1.1	Set time and parameters for the Machine Learning	41
6.1.2	How to set the x and y distributions of the cost and position matrix	41
6.1.3	Connection with the emulator	43
6.1.4	Creating empty dictionaries	43
6.2	Availability	43
6.3	Create cost matrix.	43
6.4	Action value calculation.	44
6.5	Observe the state	45
6.6	Select action	45
6.7	Route calculation	46
6.7.1	Settings of the path planning algorithm	47
6.8	Path planning and communication with the emulator	47
6.9	Save states, actions and action values	48
7	Experiments and Results	49
7.1	Scientific academic innovation of the algorithm	49
7.2	Sensitivity analysis	50
7.2.1	Determining the parameter settings during learning.	50
7.2.2	Parameter settings evaluation time	51
7.2.3	Results parameter settings	51
7.2.4	Learning time	52
7.3	Verification	52
7.4	Validation.	52
7.5	Results scenario A: the S-curve	53
7.5.1	Highway distribution	53
7.5.2	KPI 1 Productivity: Total number of completed orders	54
7.5.3	KPI 2.Container flow: The lowest number of completed orders by an AGV.	54
7.6	Results scenario B: the U-curve	55
7.6.1	Highway distribution	55
7.6.2	KPI 1 Productivity: Total number of completed orders	56
7.6.3	KPI 2.Container flow: The lowest number of completed orders by an AGV.	57
7.7	Evaluation of the algorithm	58
7.7.1	Sub-question 5a: How does the algorithm compare to the state of the art machine learning path planning, in terms of the KPIs?	58
7.7.2	Sub-question 5b: How does the algorithm compare to the path planning algorithm of TBA, in terms of the KPIs?.	58
7.7.3	Hypothesis evaluation: How does the algorithm compare to the path planning of the Forced Highway algorithm, in terms of the KPIs?	58
7.7.4	Research question 5: How does the algorithm perform in the experimental setup?	58
8	Managerial insights and discussion	59
8.1	Managerial insights	59
8.1.1	Machine vs human.	59
8.1.2	Innovation by machine learning	59
8.2	Discussion	59

9	Conclusion	61
9.1	Main research question	61
	Bibliography	63
A	Flowchart of the Learning-based path planning for container terminals algorithm	67
B	Scientific paper	69

Acronyms

AGV	Automated Guided Vehicle
AGVs	Automated Guided Vehicles
BA	Buffer Area
CQL	Coordinating Q-learning
CT	Container Terminal
DL	Deep Learning
DNN	Deep Neural Network
DQL	Deep Q-Learning
GC	Gantry Crane Crane
GCs	Gantry Crane Cranes
HW	Highway
KPI	Key Performance Indicator
KPIs	Key Performance Indicators
MA	Multi Agent
MARL	Multi Agent Reinforcement Learning
MBRL	Model Based Reinforcement Learning
MC	Monte-Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
ML	Machine Learning
MLPP	Machine Learned Path Planning
MLPP	Machine Learning Path Planning
MV	Multi Vehicle
NN	Neural Network
OPABRL	Optimized Path Algorithm Based on Reinforcement Learning
PP	Path Planning
PPO	Proximal Policy Optimization
QC	Quay Crane
QCs	Quay Cranes
QL	Q-Learning
RL	Reinforcement Learning
RLPPA	Reinforcement Learning Path Planning Algorithm
SA	Stacking Area

SL Supervised Learning

SP Stacking Position

SQL Structured Query Language

SVRLPPA Single Vehicle Reinforcement Learning Path Planning Algorithm

TBA Technisch Bestuurskundige Adviesgroep

TD Temporal-Difference

TEAMS Terminal Equipment Automated Management System

TEU Twenty-foot Equivalent Unit

TOS Terminal Operating System

TSN Time Space Network

UL Unsupervised Learning

YC Yard Crane

VARLPPA Vehicle Aware Reinforcement Learning Path Planning Algorithm

List of symbols

Symbol	Definition
T	model
$R()$	reward function
π	policy
s	state
a	action
\hat{T}	estimate of the model
\hat{R}	estimate of reward function
$V(s)$	Value of a given state
$R(s, a)$	reward of the optimum action of the current state plus
γ	discount factor

1

Introduction

As the world economy keeps growing, the container cargo shipping demand will keep increasing, particularly in developing countries [32]. Graph 1.1 shows an overview of container throughput in million Twenty-foot Equivalent Unit (TEU) per year in the last decade. The increasing amount of flow in goods demands terminals to be able to cope with large deep sea container ships [16]. With the growth of the demand for TEU transport, the demand for smarter and faster container terminals increases with it.

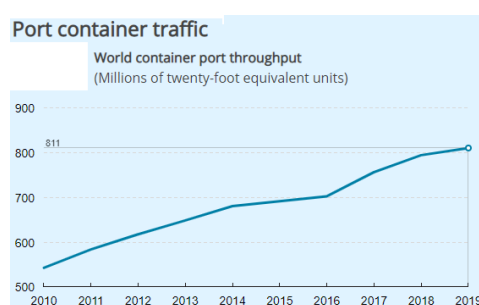


Figure 1.1: World container traffic in millions of TEU per year [35].

Figure 1.2 shows an overview of the continuous increasing growth of container ships. To be able to cope with these situations and the limited port space around large cities, existing ports have to evolve to retain their market position. Simultaneously, the need for larger and more efficient ports increases. Consequently, the demand for know-how increases. Modern technologies make it possible to gather data, model situations and better process them [27]. Training machines to cope with data or situations becomes a common solution in the increasing demand for efficiency.



Figure 1.2: Fifty years of container ship growth

1.1. Problem statement

The demand for efficient and smart automated container terminals increases. Their layouts differ among one other because of different surroundings, equipment, degree of automation etc. This causes different challenges for each terminal. In automated container terminals, the Automated Guided Vehicles (AGVs) are guided by a path planning algorithm. In some cases, the layout leads to exceptional situations where case-specific solutions have to be found by the engineers to optimize the behaviour of the AGVs in the terminals. Tuning or even building new software for each single situation requires a lot of manpower. A solution for the path planning software to learn how to cope with exceptional situations in (new) terminals can be Machine Learning. Machine Learning is software which uses data to draw conclusions or choose an action. An example would be to use information of the terminal to choose the most efficient path for the vehicles. In doing so, Machine Learning software can cope with any new situation. Conclusions of the Machine learning algorithm can be integrated into the existing software so this is improved. Therefore there is potential in the integration of machine learning in software for the path planning systems in automated container terminals.

1.2. The research objective

By applying Machine Learning (ML) to path planning in container terminals, this master thesis contributes to the improvement and scientific academic innovation focused on the path planning software of container terminals. The current research has multiple objectives, as described below.

- Find a gap in the literature, by reviewing literature describing the state of the art path planning, as well as path planning combined with machine learning algorithms.
- Design an innovative algorithm capable of path planning in container terminals.
- Realize the innovative algorithm capable of path planning in container terminals.
- Evaluate the algorithm based on testing a practical case.

1.3. Research questions

The research objectives described in 1.2 and the problem statement described in 1.1 lead to the following main question:

How can Machine Learning increase the efficiency of path planning for automated guided vehicles at a container terminal?

In order to get a clear answer to this main question, several sub-questions need to be answered:

1. What are the challenges in AGV Path Planning in automated container terminals?
 - (a) What are Path Planning problems algorithms?
 - (b) What are state of the art solutions for Path Planning problems at container terminals?
2. What gap is found in the literature regarding Reinforcement Learning applied to Path Planning?
 - (a) What is Reinforcement Learning?
 - (b) Which Reinforcement Learning algorithms are applied in path planning?
3. What RL methodology suits best for this case?
 - (a) What data from the automated container terminal can be used for the algorithm?
 - (b) What are the key performance indicators?
 - (c) How will the scientific academic innovation and value for TBA be determined?
4. What is the Markov Decision Process (MDP) for this case?
 - (a) What are the states?
 - (b) What are the transition probabilities?
 - (c) What are the actions?

- (d) What is the total reward function?
5. How does the algorithm perform in the experimental setup?
- (a) How does the algorithm compare to the state of the art machine learning path planning, in terms of the KPIs?
 - (b) How does the algorithm compare to the path planning algorithm of TBA, in terms of the KPIs?

1.4. Research scope

This section describes the scope within which the research is performed. The scope is limited to path planning for AGV's in the internal transport section of the port. The following sections describe the layout of the terminal, the infrastructure within the internal transport section and the environment during the research.

Layout of automated container terminals A common container terminal exists of a waterside, a horizontal transport area, a stacking area and a land side area, as shown in figure 1.3. Each of these sections has its own equipment and vehicle types. The equipment of a typical sea terminal is shown in figure 1.4.

In section 1 (Waterside), the transport between ship and shore takes place. This can be done using a Quay Crane (QC) or mobile harbour cranes. Section 2 (Internal transport) consists of the horizontal transport by AGVs and by straddle carriers. For each Quay Crane, six to eight AGVs are deployed. The stacking area in section 3 (Stacking area/yard) prepares the transport for the landside. At some container terminals the external trucks are directly loaded/unloaded from the stacking area. Section 4 (Landside: trucks) is used to transport containers to and from the train. Lastly, the gate in section 5 (Landside: rail) is used for safety and registration of the containers on the port [16].

The internal transport section is an area where path planning is key for the performance of the terminal. The Quay cranes should never have to wait for a vehicle to place its load on or to take it from. Therefore the scope of this research is limited to the internal transport (section 2,) where the AGVs move between the Quay Crane and the stacking area.

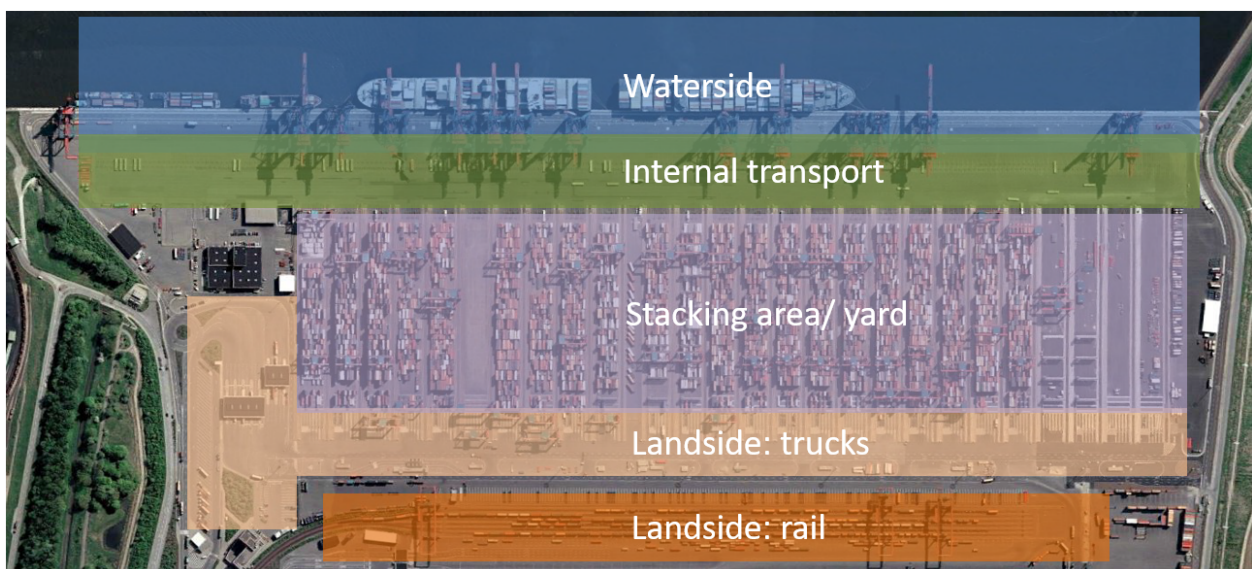


Figure 1.3: Layout of a typical container terminal

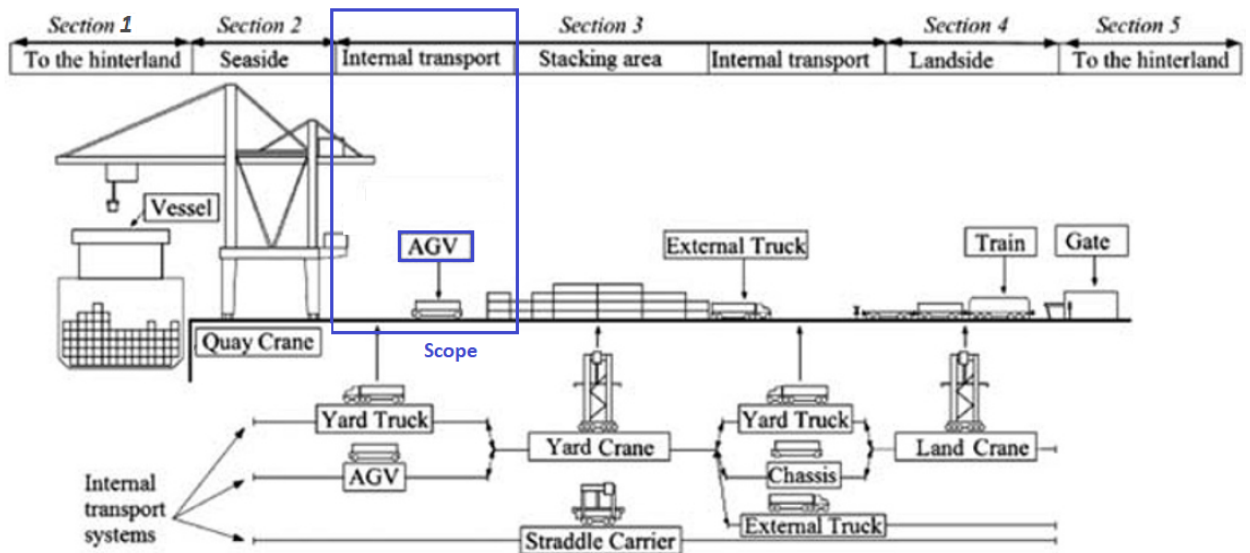


Figure 1.4: Equipment of a typical sea terminal [16]

Internal transport infrastructure Automated container terminals can use different patterns for the AGVs to travel between the stacking area and the Quay Crane (QC). One of them is the *traditional loop topology* where the AGVs travel in recurring loops, independent from their current location or destination, from the stacking area to the QC, as shown in 1.5. The advantages of this methodology are that AGVs always travel in the same direction, the chances of crossing each other are relatively low, and the predicted travel time is more constant. Therefore this system is robust. A disadvantage is that the travel time and distance are relatively high. Another type of infrastructure is the Mesh layout, using a Manhattan-grid of nodes, as shown in Figure 1.6. This layout allows AGVs to travel more directly to their destination, while the grid provides a certain structure in the terminal that minimizes the chance of collision. A disadvantage is that deadlocks, situations where two AGVs wait for each other to pass, can occur.

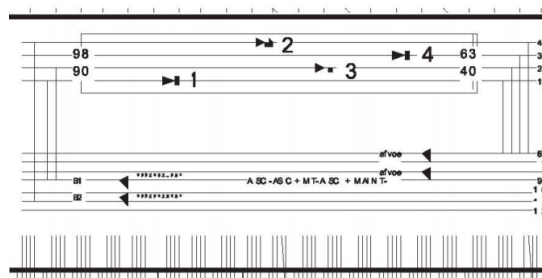


Figure 1.5: Loop topology of AGVs in a Container Terminal [10]

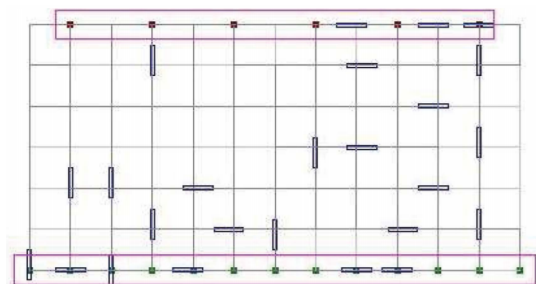


Figure 1.6: Manhattan-grid of nodes [10]

Environment during the research During this research, the AGVs drive along Manhattan grid nodes, as shown in figure 1.7. The bottom areas labelled with A are the claim areas in front of the stacking area. The six horizontal lines are the highways, and the dots at the top represent the buffer areas. Since this research is limited to the Internal transport area, the AGVs in this study will only interact with each other and not with the equipment in the other areas.

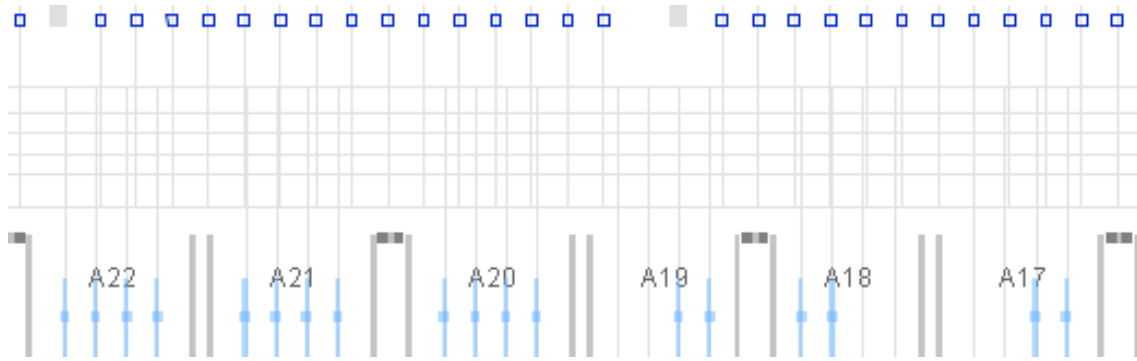


Figure 1.7: Manhattan grid of a container terminal

1.5. Research structure and methodology

The research structure is based on the method proposed by Bahill and Briggs. It consists of seven phases which form the acronym SIMILAR. The SIMILAR method is used by system engineers and project managers [1]. Figure 1.8 shows the SIMILAR applied to this project.

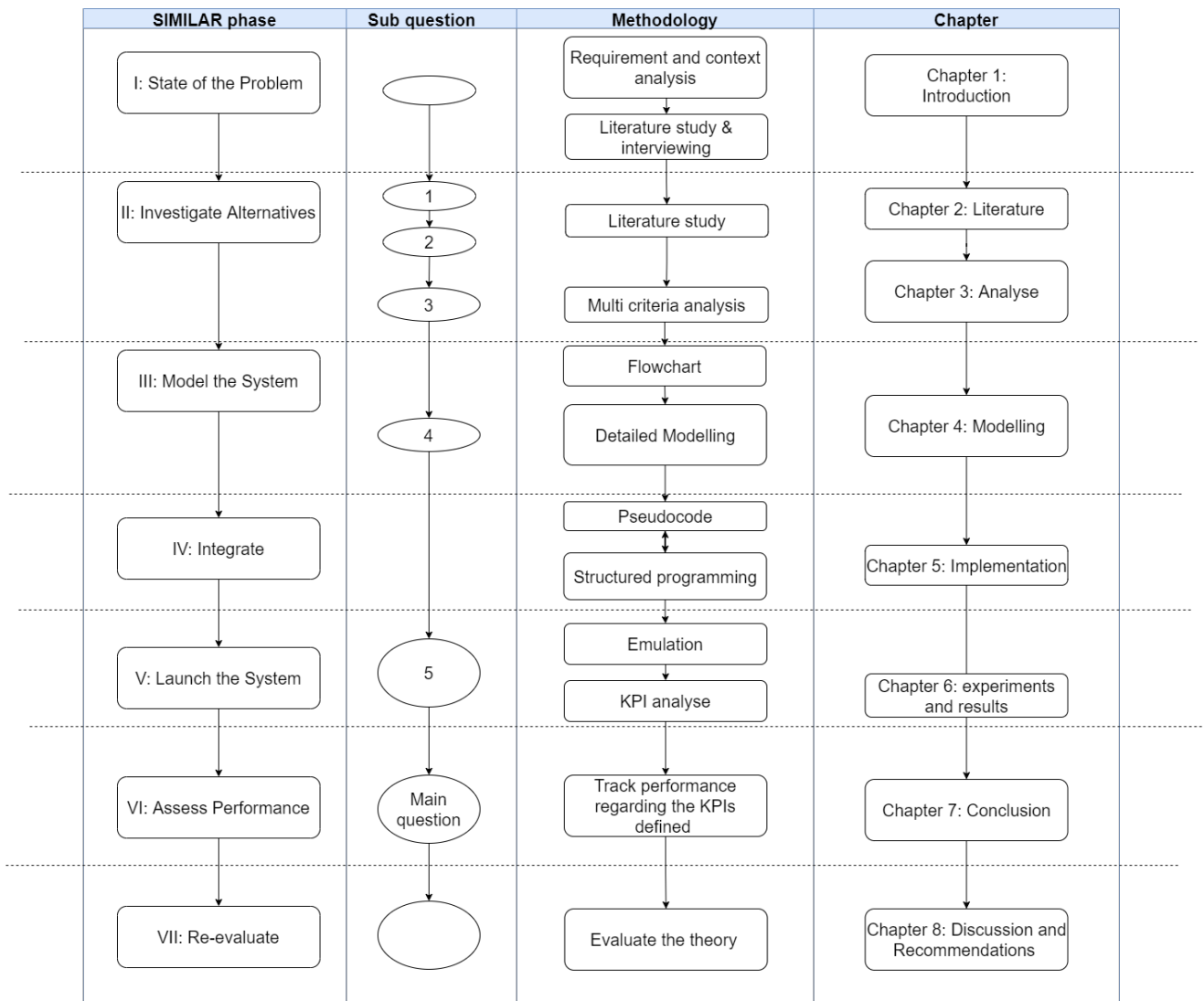


Figure 1.8: SIMILAR method for this project

2

Fundamentals and practical Case

The current research will propose an innovative algorithm that uses machine learning and is applied to a container terminal emulated by software from TBA Group. This chapter describes the background of TBA Group and the fundamentals of machine learning.

2.1. Research context: TBA Group

TBA Group is a global consultancy and software company that focusses on improving logistic systems. They deliver cutting-edge software and services to design, implement and optimise solutions for the following three categories:

- Ports & terminals that are able to handle containers and/or general bulk cargo;
- Warehouse & distribution;
- Industrial & manufacturing.

TBA is headquartered in Rijswijk, the Netherlands, and has centres in Leicester and Doncaster in the UK, Satu Mare in Romania and Düsseldorf in Germany. The rest of this section will discuss the container terminal department of TBA Rijswijk.

TBA covers three elements: design, implement, optimise. These elements are shown in figure 2.1. This research is focused on the Implementation element, which is further discussed in the sections below.

Lifecycle approach & added value

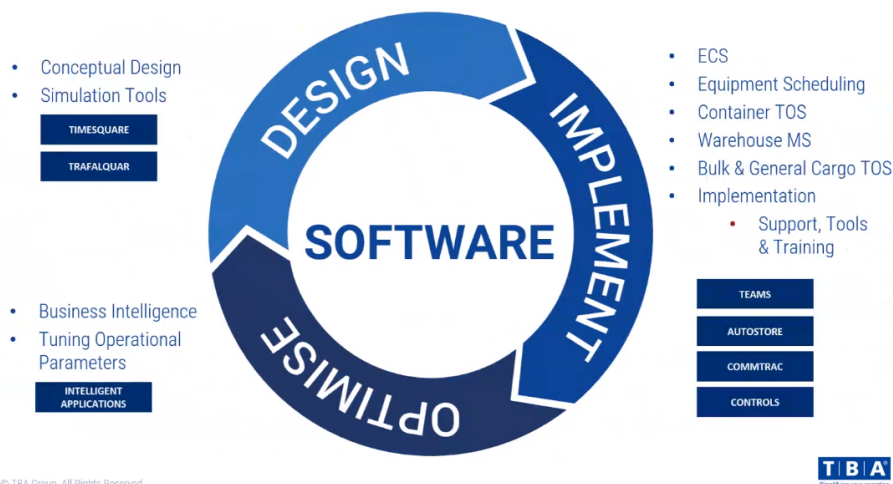


Figure 2.1: Scope of TBA Group

2.1.1. Implementation

The Implementation department deals with deploying a new design or upgrading the software in an existing design. The team at TBA prepares, tests, tunes, trains users and deploys new complex systems. This research will mainly focus on management system TEAMS, which controls the movement of the AGV in the terminal.

Terminal Equipment Automated Management System (TEAMS) TEAMS controls the equipment in the port based on the orders from the Terminal Operating System (TOS). Figure 2.2 clarifies the position of the TEAMS software between the TOS and the equipment.

TEAMS determines the routing movement, avoids collisions, interchange and interface functionality of the vehicles. TEAMS can operate with various kinds of TOS, since there are different TOS applied in all kinds of ports. Figure 2.3 shows the interface of TEAMS that gives an insight in the status of the equipment it controls.

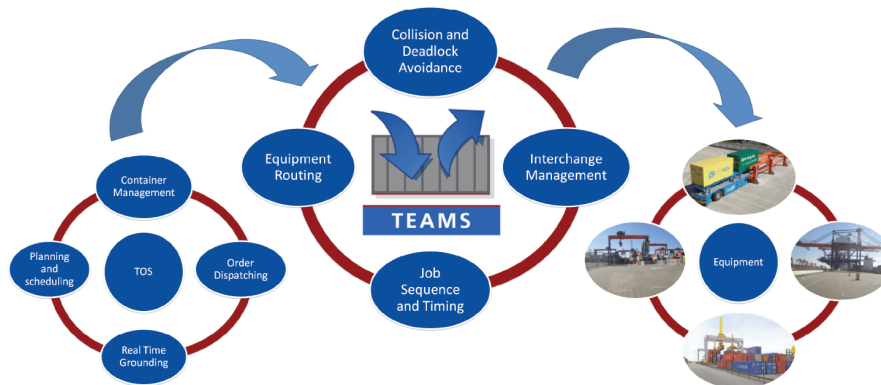


Figure 2.2: Terminal Equipment Automated Management System (TEAMS) software integration

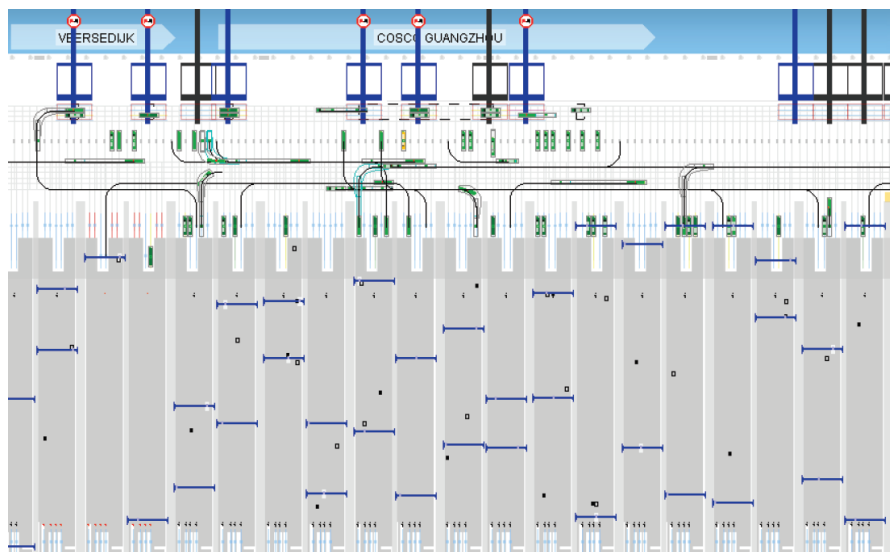


Figure 2.3: Terminal Equipment Automated Management System (TEAMS) interface

Container TOS The Terminal Operating System (TOS) is software which is responsible for the logistics of a container terminal. The information is based on the way a ship has to be loaded/unloaded, as well as instructions for loading/ unloading the riverboats, trains and trucks.

Equipment routing: the pathfinding of TEAMS The pathfinding is based on the A* algorithm for each individual AGV. This is a widely used algorithm to find the exact solution.

Emulation The Emulation software allows for testing a virtual port, using the Terminal Operating System of the customer. It can be described as the digital twin of the terminal. The emulation software of TBA is called CONTROLS.

2.2. Research context: Machine learning

Machine learning is the ability of a computer to convert their experience based on the input into knowledge or expertise [31]. In 1959 A. L. Samuel wrote a paper about machine learning where he states that “Programming computers to learn from experience should eventually eliminate the need for much of detailed programming effort.”[30]. In 2002 Lim et al. proposed a machine learning algorithm for an AGV guide-path design to optimize the travel time considering physical distance, interference of vehicles and the waiting time[22].

2.2.1. Why would one apply machine learning?

Numerous tasks that can be done by humans, such as speech regression or driving, have been shown to be achievable by machine learning programs as well. They can translate huge amounts of data in patterns that humans can use for their benefit [31]. Machine learning can be the preferred choice over explicitly programming if:

- The human is not able to do the task or will be outperformed by the machine;
- The environment is adaptive;
- The task changes over time;

2.2.2. Types of machine learning

The literature typically divides machine learning algorithms into different categories. One of the categories is based on learning style: Unsupervised Learning (UL), Supervised Learning (SL) or Reinforcement Learning (RL) [8], as shown in figure 2.4.

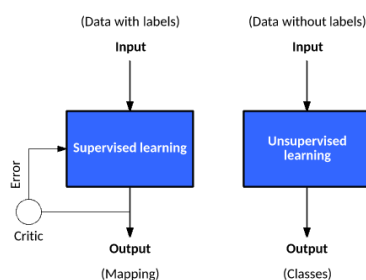


Figure 2.4: Supervised and unsupervised machine learning [2]

Supervised Learning The input data of SL is labelled. Labelled data is data that is tagged with information, such as a name or a category. An example is an algorithm that needs to distinguish dogs or cats from images of dogs and cats that are labelled with the tags 'dog' and 'cat'. The algorithm determines from the image whether it contains a dog or a cat, while the supervisor checks if the answer corresponds to the tag. The algorithm can estimate the success of its answer, using the labelled training data by, for example, computing the empirical loss [31], or in case of the dog or cat example, the percentage of cats versus dogs. The goal of SL is to predict outcomes for new data, e.g. the prediction of prices, text classification and image recognition.

Unsupervised Learning The input data of UL is unlabelled, which means that the data is not defined. UL can be applied for statistical modelling, creating graphical models, summarizing or compressing input data, also called clustering [31] [15]. The goal of clustering the data is to find patterns in it. Hence, the goal of UL is to get insights from large volumes of data, such as grouping Web pages based on their subjects, decision making, predicting future inputs and efficiently communicating the inputs to another machine[4]. There are many variants of UL and this topic is enormous [15].

Reinforcement Learning When the goal is to teach an agent to interact with an environment, a dataset is not always available. In that case, Reinforcement Learning (RL) is the preferred choice. The advantage of RL over UL and SL is that it can improve itself, whereas the UL and RL agents can never become smarter than the best data they use.

There are many variants of RL. Most of them learn a policy by the (positive or negative) reward the algorithm gets as a consequence of its action. For every action A_t the agent receives a new State S_t and Reward R_t . These are based on the effect the action A_t had on the environment, as seen in figure 3.2 [15] [34].

Examples where RL is applied is the control of vehicles [22] or learning how to play games. One breakthrough was that an algorithm using RL was able to win the game "Go", that has over 2.1^{1070} legal board positions, from the best players in the world [2].

For RL to learn from scratch can be very hard, time consuming or even impossible. Sometimes the algorithm comes up with a solution that maximizes the rewards while not leading to the desired behaviour, or the algorithm cannot find a solution at all as the environment and the rewards are just too unstable. Therefore, the reward signal is very important and the use of RL requires simulations. These simulations allow the algorithm to learn before being applied in the real world. Combining different algorithms and settings can be important, as some are better at exploring and others need a model of the environment.

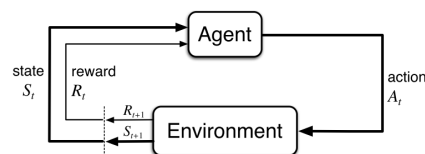


Figure 2.5: Reinforcement Learning [34]

Multi Agent Reinforcement Learning (MARL) The cooperative variant of RL systems is Multi Agent Reinforcement Learning (MARL) where different agents take a shared objective into account [5]. This has potential for cases where multiple agents share a reward, e.g. the throughput time of packages in a warehouse. The states of the agents of the MARL are merged, therefore it can become exponentially complex with each added agent [5]. Moreover, the new policy of the agents is behaviour based on the previous states. This slows down the learning process with respect to the single agent RL.

2.3. Research context: Path Planning

When planning a path from origin to destination, path planning algorithms can provide the solution. One of the first algorithms finding the shortest path was the Dijkstra algorithm, created in 1956 [7]. It is a subject that shows potential for the increasing demand for efficient movement of cargo. Therefore the subject has drawn more attention in recent years, especially in the design of ports. According to Zhong et al. "Path planning and integrated scheduling are two important problems to be resolved in the design of automated container terminals. There has been relatively little research on automated guided vehicles (AGVs) conflict-free path planning with quay cranes (QCs) and rail-mounted gantry (RMG) cranes." They use a mixed integer programming model for Path planning. Their conclusion is that this is very time consuming and they suggest that applying Machine Learning can increase the efficiency of automated container terminals [45].

2.4. Practical case

An example of path planning software in container terminal is a case study at TBA. The path planning of AGVs in some container terminals is now automated by their software TEAMS. Their algorithm is programmed so that the lanes parallel to the quay, called a highway, are taken into one direction by the AGVs. This is assumed to be efficient based on the experience. The integration of machine learning might enable the software to adapt the direction of the highway without having to find and program this optimal direction by their engineers.

3

Literature Review

This chapter discusses the literature research which is done to find the current knowledge in the field of using reinforcement learning for the equipment control systems of (fully) automated container terminals. First, the methodology for this literature search is discussed, then the challenges for path planning in container terminals, next the state of the art path planning in container terminals and finally the literature discussing reinforcement learning.

This chapter will answer sub-question 1 and 2:

1. What are the challenges in AGV Path Planning in automated container terminals?
 - (a) What are Path Planning algorithms?
 - (b) What are state of the art solutions for Path Planning problems at container terminals?
2. What gap is found in the literature regarding reinforcement learning applied to Path Planning?
 - (a) What is reinforcement learning?
 - (b) Which reinforcement learning algorithms are applied in path planning?

3.1. Review methodology

The methodology of the literature research is based on the methodology of [36]; selecting a time frame, searching for literature by the use of snowballing (the effect of finding new literature related to the found literature), and selecting papers by explicit criteria. Sub questions are formulated that need to be answered by quantitative research in the literature.

For the search of literature, we use the largest abstract and citation databases of peer-reviewed literature, Elsevier Scopus and Google Scholar.

Selection criteria: Keywords related to the sub question are chosen. The keywords are sorted by relevance. When the papers found are not related to the sub question; keywords are deleted in the order of their relevance.

Papers that are cited a lot are preferred over the ones that are cited less.

Time frame: To make sure that the information is state of the art it may not be older than a decade.

3.2. Sub question 1: Path planning algorithms

This section describes the findings for sub question 1a: *What are Path Planning algorithms?* and 1b: *What are state of the art solutions for Path Planning problems at container terminals?* in order to answer sub question 1: *What are the challenges in AGV Path Planning in automated container terminals?*

3.2.1. What are Path Planning algorithms?

Path planning algorithms are used in a variety of situations and environments where routes need to be determined. The algorithm is used to find the shortest path based on the given data. Examples are logistic environments, such as finding the best route for a mobile robot [9], guiding autonomous vehicles across a campus environment [28], navigation of ships in wind farm areas [40], and, as is the case in the current paper, Automated Guided Vehicle (AGV) in container terminals. The goal of the algorithm is dependent on the needs of the user. Examples are to find the lowest cost, shortest path or most fuel efficient-path. Thus, for each situation another path planning algorithm fits best. The fundamental algorithm for Path planning is the Dijkstra algorithm. A variant of it, the A* algorithm is widely used [44]. Both are discussed in the coming subsections.

Dijkstra algorithm One of the first and best known path planning algorithms is the Dijkstra algorithm. The algorithm is given by forehand the start and end position, nodes of the environment and the cost value of the edges between the nodes [7]. In case of the container terminal this could be the cost in terms of the distance or travel time between nodes. The algorithm is a deterministic algorithm, which means that for a given input there will always be the same solution.

A brief explanation of the algorithm: From the starting point, each edge towards the neighbouring nodes are examined to find the combination of nodes with the lowest cost value, i.e. the shortest paths between each of the nodes. The path with the lowest cost towards a node which has not been examined yet is picked next. The algorithm keeps track of the examined edges, so nodes will not be examined twice. This process is repeated until the destination node is reached and there is no other node that, if visited, will increase the path cost. The node connections with the lowest edge cost are connected and a the the shortest path between the given start and end position is found.

A* algorithm A* is based on the Dijkstra algorithm. It is in many circumstances the fastest shortest pathfinding algorithm in a state space search when the road network is static due to the fact that in many cases checks less nodes or vertices compared to the algorithm of Dijkstra [19][23]. It does so by an added heuristic, as seen in equation 3.1. The heuristic is an estimation of the cost between the current node and the end point, e.g. the euclidean distance. As the cost of the heuristic decreases the further the algorithm advances towards the end point, the algorithm prefers the nodes that are closer to the end point. This gives the algorithm a clear search direction, as well as speed, because the more distant nodes relative to the end point will have a higher cost due to the added heuristic, and will not be examined. One important condition for the A* algorithm to give an exact solution is that the heuristic must be admissible: it will never overestimate the real cost.

Figure 3.1 gives an insight into how the algorithm searches for the end position (arrival node).

$$F(v) = H(v) + G(v) \quad (3.1)$$

Where:

- $H(v)$ is the heuristic path between the node and destination, i.e. the estimated cost to move from the current cell to the destination. One method to do so is the Manhattan method. This is named after an area of New York where if one takes a top view of Manhattan one can see that when driving towards a street which is not in a straight line connected to the current one, one has to take a minimum of one corner. This way it it will never underestimate the cost of the route. The A* is equivalent to the Dijkstra algorithm if the heuristic variable is equal to 0.
- $G(v)$ is the cost of moving from the previous cells to the current one.
- And $F(v)$ is used to find the least cost, responsible to find the optimal path between source and destination [9].

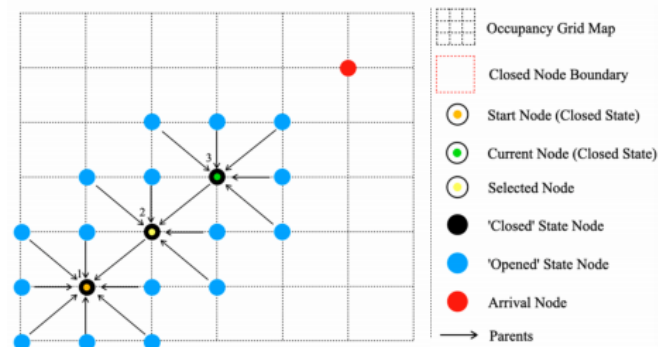


Figure 3.1: A* algorithm [28]

Unknown scenarios Dijkstra and A* are based on a static environment and will not deliver the shortest path when the costs are unknown. However, the path planning algorithms are used in environments that still need to be explored. Examples are re-planning the route in local areas when collisions are predicted [42].

Zhang and Li [42] suggest a local target selection where, if a collision segment is detected in a radius around the vehicle, a new target is selected to re-plan the path around this collision segment in order to reach the end position. In this way, the path planning algorithm can be used in a unknown environment.

3.2.2. The state of the art in path planning in automated container terminals

In this subsection the following sub-question will be answered: *What are state of the art solutions for Path planning problems at container terminals?*

The sections below describe the developments of path planning in container terminals. The approaches are divided into two categories based on the infrastructure: fixed layout and free range infrastructure.

Static routing on a fixed layout There are different types of infrastructure for container terminals. A fixed layout means that there are certain guidelines the AGVs have to follow. For example, the static route on a fixed layout means that the AGVs drive along a fixed path in the area.

An important factor to consider when designing an AGV system is preventing collisions, deadlocks and congestion. The work of Gawrilow et al. [14] suggests a static path planning algorithm that is able to avoid deadlocks by considering a time-expanded network. Their algorithm adds waiting time to the route to allow other vehicles to pass. Their goal is to be more robust with respect to dynamic routing. Their research only shows improvement with medium and small traffic scenarios due to the calculation of their deadlock avoiding reservation schedule.

Dynamic routing on a fixed layout Another type of infrastructure is dynamic routing on a fixed layout, where the AGVs can choose a path based on a grid structure in the area, also known as mesh routing.

Li et al. [21] propose and test a Quantum ant colony optimization algorithm (QACO). Quantum stands for the usage of quantum computing. This is a way of mechanical computing where a quantum mechanical system can do several operations at the same time, whereas the classical computer cannot. The Ant Colony Optimization algorithm is based on the behaviour of ants in a colony, who spread pheromones when they find food. In the same way, the algorithm is programmed to find the shortest path within parts of the map and will spread 'pheromones', i.e. the reward. At the same time, the AGVs spread a repulsive pheromone that will make the AGVs avoid each other in their path, thereby preventing collisions. This way the complete map can be investigated efficiently.

Hu et al. [17] propose a solution that combines preplanning with real time planning to create a time window based A* algorithm. They strive for the shortest collision-free path with the least number of turns. The algorithm generates multiple shortest paths using A* and selects the one with the lowest number of turns. Their approach minimized AGV travel distance, which reduced the completion time and the average response time significantly.

Another approach is suggested by Zhong et al.[46], who aim for conflict-free AGV movements, which increases the operational efficiency of the container terminal. They do this by applying a heuristic algorithm:

the Dijkstra Depth-First Search algorithm for their priority-based speed control strategy. This is a heuristic algorithm to obtain the optimal path in an automated container terminal. Their results show operational efficiency and reduce the conflict probability of AGVs.

Free range infrastructure In contrast to the fixed layout, free range routing means that the AGVs do not have to follow any guidelines and can move completely freely in the area. Duinkerken and Lodewijks [10] proposes a free range path planning algorithm based on the behaviour of pedestrians, called Dynamic, Evasive Free-ranging Trajectories. The algorithm calculates trajectories for individual AGVs, based on costs for speed of movement, time pressure, static obstacles and other vehicles. The results showed a reduction in trajectory length, waiting time and a significant improvement in the moves per hour with respect to mesh routing.

The intensive interaction between different vehicles at a container terminal leads to longer executing time of the orders. Xin et al. [38] therefore investigate collision-free scheduling of interacting machines in automated container terminals, which incorporates the planning of free range AGVs and the scheduling of different types of equipment. The authors suggest a methodology of path planning in a hierarchical order. The system sequentially solves collections of mixed integer linear programming problems. Compared to the widely used mesh routing, the average distance travelled by the AGV is reduced significantly. This is because the range free AGVs drive a shorter route. However, this result was only found when an insufficient number of AGVs were involved. An insufficient number occurs when there are not enough AGVs per Quay crane, so the Quay Crane cannot unload the container and has to wait for an AGV. The order time is not improved when a sufficient number of AGVs are involved. The advantage of this method is that fewer AGVs are needed to reach the same capacity. However, this method also brings great challenges for operational terminal control, because of its complexity and computational issues.

Table 3.1 gives an overview of literature regarding the path finding technologies which are leveraged in container terminals in the last decade.

Table 3.1: Path planning solutions in automated container terminals

Reference	A*	DA	DEFT	GA	LB	MIP	QACO	TW	RL	Focus
Gawrilow et al., 2012 [14]	-	-	✓	-	✓	-	-	-	-	Be collision and deadlock-free and outperforming dynamical PP algorithms
Li et al., 2018 [21]	✓	-	-	-	-	-	✓	-	-	AGVs path planning at an ACT using QACO
Hu et al., 2020 [17]	✓	-	-	✓	-	-	-	✓	-	Minimizing AGV travel distance, reducing operation and response time
Zhong et al., 2020 [46]	-	✓	-	-	-	-	-	-	-	Priority-based SCT for ACTs, reducing conflicts and TT
Duinkerken and Lodewijks, 2015 [10]	-	-	✓	-	-	-	-	-	-	AGV PP based on pedestrian behaviour
Xin et al., 2015 [38]	-	-	-	-	-	✓	-	-	-	Planning of free-ranging AGVs and the scheduling of different types of equipment.
Xin et al., 2020 [39]	-	-	-	-	-	✓	-	-	-	PP of AGVs using a local variant of a time-space network in a manhattan grid ACT
This research	✓	-	-	-	-	-	-	-	✓	Applying RL to create vehicle aware input data for an A* Path Planning (PP) algorithm in a container terminal environment.

Abbreviation algorithms: **A***(A* search algorithm), **DA**(Dijkstra (based) algorithm), **DEFT**(Dynamic, Evasive Free-ranging Trajectories) **GA**(Graph Algorithm), **IA**(Interpolation Algorithm) **LB**(Load Balancing algorithm) **MIP**(Mixed Integer Programming) **QACO**(Quantum Ant Colony Optimization Algorithm) **TW**(Time window based)

Abbreviation terminal: **AGV**(Automated Guided Vehicle), **PP**(path planning) **ACT**(Automated Container Terminal), **SCT**(speed control strategy) **SA**(Stacking Area), **TD**(Travel Distance), **TT**(Travel Time)

3.2.3. The challenges in path planning in automated container terminals

This subsection answers the question: *What are the challenges in path planning in automated container terminals?*

The key objective of path planning is to discover a path which avoids certain obstacles and passes over the given points while being as efficient as possible [13]. Inefficiency of multi-AGV processes will have direct influence on the complete automated container terminal.

Several methods have been developed in order to reach this objective. Static routing on a fixed layout has been proven to be robust, but not efficient. Dynamic routing on a fixed layout is more efficient, but deadlocks, collisions and congestion are widely spread problems that need to be manually solved in the algorithm each time they occur. The free range layout does not always prove to be more efficient, while at the same time bringing challenges in terms of computational time and operational terminal control.

3.3. Sub question 2: reinforcement learning for path planning

This section describes the findings for sub question 2a: *What is reinforcement learning?* and 2b: *Which reinforcement learning algorithms are applied in path planning?* in order to answer sub question 2: *What gap is found in the literature regarding reinforcement learning applied to Path Planning?*

3.3.1. Sub question 2a: What is reinforcement learning?

Machine learning is the ability of a computer to convert their experience into knowledge [31]. A reinforcement learning (RL) algorithm will learn from the interaction with its environment. An agent decides what actions will be taken based on the observations of the environment. The *state* describes what an agent observes in a numeric matter. Furthermore, the *action* the agent initiates is based on the *policy*. The *reward* is a calculation based on the consequences of the action, which reflects how well the action performed regarding the goals. This process is shown in figure 3.2.

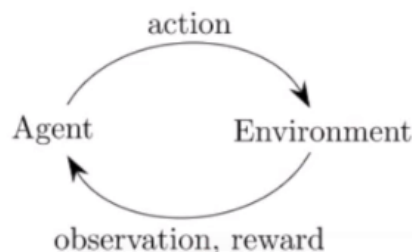


Figure 3.2: Basic RL loop

The basic idea of RL is learning on the job. For every *time step*, which can be a pre-defined increase in time, the agent receives a *state* (s) in a *state space* (S) and selects an *action* (a) from an *action space* (A) based on the current *policy* $\pi(s)$ for that state. After the action, the agent receives a *reward* (r) and a new *state* (s'). This process is shown in figure 3.4.

The goal of the agent is to find an optimal *policy* π^* which maximizes the total *reward*. As Zhang and Li [41] state: "Reinforced learning, also known as reward-learning, learns the optimal behavioural strategies of dynamic systems by perceiving changes in the dynamic environment and obtaining uncertain rewards and punishments from the resulting actions, and evaluating the pros and cons of the movements".

To do so, the agent can use two strategies: exploration and exploitation. Exploration is finding new information on the environment in order to discover a new and potentially better policy. Exploitation is using the current information for reward maximization. Finding the right balance between exploration and exploitation is essential in RL.

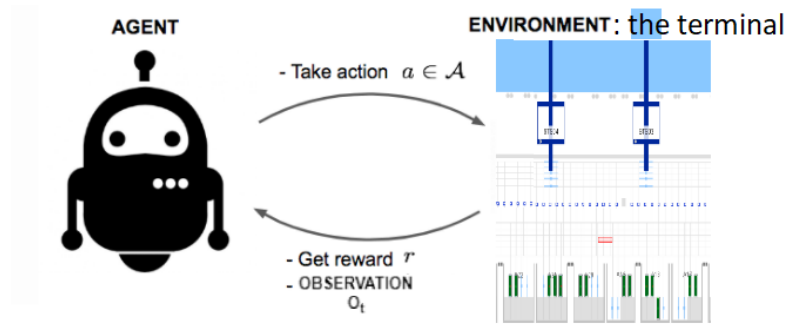


Figure 3.3: Reinforcement learning coupled to environment [37]

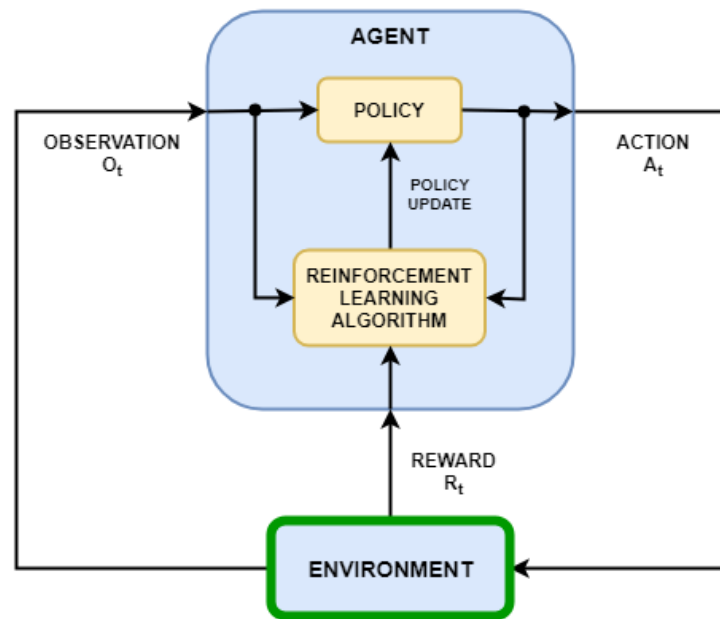


Figure 3.4: Reinforcement learning basic [?]

Reward function The input of the *reward* function is the *state* and the *action*, and the output is the value of the *reward*. The rewards can be programmed as either positive or negative rewards. The following are possible ways the value can be calculated.

Sparse reward returns either a reward or no reward at all. In this case this could be the reaching of a destination. The downside of this is that it can take a long time of training, because the agent is not guided into the right policy direction.

Shaping reward is the reward system that can reward the agent for every step along the way. An example of a positive shaping reward is providing a higher reward when the AGV is closer to the end point. This way, the agent is guided into the right direction. One has to be careful with this type of reward, as the downside can be that the agent will never reach its goal but keeps driving around it, as it keeps getting positive rewards for getting close. The negative reward is a solution in this case, e.g. when the agent gets a negative reward for every step it takes, it will minimize the amount of steps it takes to reach its goal in order to maximize its reward. Another example is when one wants to drive as environmentally friendly as possible, one can program a negative reward for taking a corner, breaking or accelerating.

Terminal conditions Terminal conditions can be set during the learning phase; when these are reached the environment will reset. This is beneficial when for example all AGVs are unable to continue their path. In this case, resetting the environment will avoid wasting valuable learning time. There are different types of

terminal conditions: *time terminal* is when a time limit is reached and *positive terminal* is when the agent reached its goal.

Active and passive RL Passive reinforcement learning evaluates how well a model is performing by the given policy. On the other hand, the goal of active reinforcement learning is to optimize or learn a policy [3], by updating the policy as it learns. For this project, the active reinforcement learning is used, as we want to optimize the policy of the AGVs.

3.3.2. Types of reinforcement learning algorithms

There are different types of RL that are useful for specific challenges [25]. They can be categorized by their learning method: model-based or model-free (value-based or policy-based). This section will describe the definitions of these categories, the best known algorithms and how they work.

Model-based Model-based reinforcement learning requires a model of the environment. A model is defined as anything an agent can use to predict the response of the environment. Hence, the behaviour of the environment is modelled and all the states are known beforehand [29].

The model simulates the environment and returns a simulated experience. The model can be either stochastic or deterministic. Maximizing the reward often leads to a deterministic policy, but in some cases, this is not desirable. For example, some parts of games need to be unpredictable to maximize the reward, so stochastic models have several next states and rewards for each action, based on probability.

The advantage of model-based reinforcement learning is that it can converge to an optimal policy in order of a magnitude faster than a model-free approach [20]. The downside is that it can only be used when there is a model available.

An example case of model-based reinforcement learning is Atari games, which utilize a pre-defined model [34]. Model-based RL is rarely applied to real world scenarios, as they are complex to model.

Value-based Value-based learning is a model-free approach which bases its algorithm on trial and error [34]. The algorithm will discover its policy during the interaction with the environment. An environment is defined as anything that cannot be changed by the agent. One of the advantages of this method is that it can be applied to many environments.

The value-based algorithm uses a table, the Q-table, to store a value for all the actions in a certain state. Initially, this table is filled with zeros. The Q-values are then calculated by a function containing the previous Q-value and the reward for the action, which depends on the objective of the algorithm. Each new action influences the state of the model, which creates a new row in the Q-table. This way, the Q-values for all actions will be updated after every iteration.

An example of a case where a value-based algorithm can be applied is the game Tic-tac-toe. The robot has nine actions it can take, i.e. the nine positions on the board where it can place the sign. The states are the ways that the board is filled. The states and actions combined create the Q-table, as shown in figure 3.5. In this case, the reward can depend on the chance of the opponent to win, as well as the chance of the robot to win. During the learning process, the agent learns what the best action is for each current state.




Game State	Top Left	Top Middle	Top Right	Middle Left	Middle Middle	Middle Right	Bottom Left	Bottom Middle	Bottom Right
	N/A	0.5	N/A	N/A	N/A	N/A	0	0	N/A
	N/A	N/A	N/A	N/A	N/A	N/A	0.5	N/A	N/A
	0.3	0.5	0.3	0.5	0.7	0.3	0.3	0.5	0.3
...

Figure 3.5: Actions and states Q-table [12]

Policy-based "Instead of computing learned probabilities for each of the many actions, we instead learn statistics of the probability distribution" [34]. In other words, as opposed to value based learning, the policy-based learning chooses its actions based on an approximation of the value, which is calculated with a probability distribution.

The policy-based method is often applied in cases that include a near infinite amount of actions. Because the algorithm works with an approximation, it reduces the number of state variables it has to interact with. In addition, prior knowledge can be used to estimate the values more efficiently [34]. Furthermore, policy-based methods are known to converge to the optimal policy while remaining stable when approximating functions [26]. A disadvantage is that there is a possibility that it will converge to a sub-optimal local policy.

3.3.3. Sub question 2b: Which reinforcement learning algorithms are applied in path planning?

In this subsection the related literature with ML and PP problems are reviewed to see what is already there regarding the subject. Table 3.2 gives an overview of the papers that are reviewed based on their relation with container terminals, the reinforcement learning type, the reinforcement learning algorithm and path planning.

Table 3.2: Reinforcement learning in a container terminal related environment

Reference	Machine Learning type	Machine Learning algorithm	Learning method	Applied in a container terminal	Path planning related	Focus
Zhou et al.,2017 [47]	MARL	NegoSI	VB	-	✓	MARL algorithm "NegoSI" applied in an intelligent warehouse problem
Eiffert et al.,2020 [11]	RL	MCTS	VB	-	✓	Path planning in a close range dynamic environment
Sichkar,2019[33]	RL	Q-learning and SARSA	VB	-	✓	Path planning in static environment using Q-learning and SARSA algorithm
Zhang et al.,2020 [43]	MARL	MATS	VB	-	✓	Multi agent tree search algorithm to operate in denser environments
Keselman et al.,2018 [18]	RL	Q-learning	VB	-	✓	Model-based simulations to learn the heuristic for A* using a neural network
Liu et al.,2019 [23]	RL	Q-learning	VB	-	✓	Combines A* with ML for a best PP algorithm that is suitable for traffic jams, accidents and temporary limits for intelligent driving vehicle
This thesis	MARL	MC	VB	✓	✓	Applying RL to create vehicle aware input data for an A* PP algorithm in a container terminal environment

Machine Learning type: **RL** (reinforcement learning), **MARL**(Multi Agent reinforcement learning), **NegoSI**(Negotiation-based MARL with sparse interactions)

Machine Learning algorithm: **MATS**(multi-step ahead tree search)
MCTS(Monte Carlo Tree Search),**SARSA**(State-action-reward-state-action)

Abbreviations: **AGV**(Automated Guided Vehicle), **VB**(Value-Based) **PP**(Path Planning), **ML**(Machine learning)

Negotiation-based algorithm Zhou et al. [47] have created a multi vehicle reinforcement learning algorithm where agents first learn their policy and reward models while interacting individually with the environment. Part of the algorithm is designed so that when other agents are close, the state space of the agent is expanded so the amount of states is limited with respect to the agents it will interact with. The state space of the agents that the agent will not interact with are excluded. If an agent chooses a relatively high risk state action pair, other agents are informed in order for them to negotiate for a policy leading to the best found equilibrium. The authors propose a sparse-interaction version of the traditional negotiation-based multi agent reinforcement learning algorithm. This way, it maintains better performances regarding characteristics such as coordination ability, convergence, scalability and computational complexity, especially for practical problems with respect to the method without negotiation.

Path planning in a close range dynamic environment Eiffert et al. [11] created an algorithm that can interact with the world around it, e.g. pedestrians, using a Monte Carlo Tree Search (MCTS) algorithm. The algorithm gets a reward by socially planning its path around objects. It is focused on collision avoidance. The MCTS allows it to solve the Markov Decision Process (MDP) where the action for the states are unknown. Eiffert et al. state that the behaviour of the agent can be adapted by changing the state and is still effective while maintaining the same policy. The model has shown to be trained by observed data and can lead to accurate prediction of close range interactions [11].

Online Path planning using neighbouring grids for multi step ahead tree search Zhong et al. [45] state that: "There has been relatively little research[...] on automated guided vehicles (AGVs) conflict-free path planning [...]". They suggest the usage of machine learning for future research. Therefore, Zhang et al.[43] apply a decentralized multi-agent RL framework on AGVs in a warehouse. They propose a multi-step ahead Monte-Carlo tree search . The state space area of each AGV is limited to a 5x5 grid where the agent observes obstacles or other AGVs. The available actions are the directions in which the agent can move or the option

to maintain its position. The advantage is that the algorithm showed to be suitable to use in a real-world warehouse case and showed improvement with respect to decentralized agents when the number of agents was below 150.

Combining static and exploring algorithms The A* algorithm is an effective path-finding algorithm when the heuristic is known. The A* algorithm is relatively fast to other algorithms [9]. It converges significantly faster if the heuristics are calculated well. The A* algorithm chooses the next node based on the lowest value of $F(V)$ where

$$F(v) = G(v) + H(v) \quad (3.2)$$

In the above, $G(v)$ is the total cost of going from the source node to the optional next node. The function $H(v)$ is a heuristic function, estimating the total cost of going from the optional next node to the destination node. This heuristic function must be admissible, i.e. it may never overestimate the cost of the actual path from the node that it is evaluating. It must also be fast to calculate, as long computational times will slow down the algorithm significantly. When this heuristic is inadmissible this will lead to the algorithm searching in the wrong direction, which can ultimately result in not finding the optimal path.

The heuristic function is not always known or can be hard to determine. Examples are areas where the maximum speeds vary, traffic jams and varying travel times due to local roads or highways that are in the environment. Keselman et al. [18] suggests a method in which the heuristic is calculated using a general model-based reinforcement learning algorithm. In a simulation environment, searches are generated where the next node is either chosen based on A* using the current heuristic or randomly with a fixed probability epsilon. After the learning is done, the policy is evaluated using full exploitation settings. The results showed that it can find paths with little branching. Little branching is defined as stopping the searching when a path is found, which improves the computational speed of the A* algorithm.

Liu et al. [23] made a path planning algorithm which combines the A* algorithm with reinforcement learning. A* is used to obtain a reference path, then reinforcement learning is used to guide the vehicle on a more detailed level. It can encounter traffic disturbances, which the algorithm uses to improve the search direction and check if the path is correctly judged. Every node in the algorithm has eight possible nodes surrounded by it. Only the four cheapest are selected to be evaluated. During the learning process, the algorithm learns state action pairs by interacting with the environment until the values are within a certain range. When a new obstacle is found, it gains knowledge by trial and error. This trial and error knowledge is used by a reinforcement learning algorithm. This is repeated until the path around this obstacle is found. The gained data can be reused the next time the path is planned. Results show advances in path length and computational time with respect to traditional algorithms.

3.3.4. Conclusion: What gap is found in the literature regarding reinforcement learning applied to Path Planning?

Under satisfied conditions, A* is the fastest optimal path planning algorithm [6]. Liu et al. [23] and Keselman et al. [18] show the advantage of combining A* with ML. This way, the exploring part of the ML algorithm is combined with the faster and more precise properties of graph search PP algorithms. However, this has not been applied to container terminals yet. There is potential in combining the A* algorithm with a reinforcement learning algorithm to create a stronger path planning algorithm in container terminals. There is possible gain in using machine learning for the heuristic or the cost part of the A* algorithm. Because the environment is unknown, during exploitation costs change over iterations. Another gap is to combine the A* algorithm with machine learning and vehicle aware planning. Currently there is no literature on this subject. Consequently, the current research investigates the possibilities of combining the A* algorithm with machine learning and vehicle aware planning. This concept is worth investigating, as this has never been done before for a container terminal.

4

Experimental Setup

The algorithm proposed by this research will be applied to the environment and the software from TBA. In order to create this algorithm, a methodology needs to be selected that suits the environment and the objectives of this research.

Therefore, this chapter will answer research question 3: *What RL methodology suits best for this case?*

First, the functioning of the container terminal emulator by TBA is analysed. In doing so, sub-question 3a is answered: *What data from the automated container terminal emulator can be used for the algorithm?* With this knowledge, the literature review of chapter 3.3.3 is revisited in order to find a suitable RL methodology. Next, in order to reach the research objectives, sub-question 3b is answered: *What are the key performance indicators?* Lastly, sub-question 3c is answered: *How will the scientific academic innovation and value for TBA be determined for the algorithm?*

4.1. The functioning of the container terminal emulator by TBA

To create an algorithm and teach it a policy, it is important to make the training environment as realistic as possible, in order to make the learned policy as realistic as possible. To build a new environment based on a container terminal takes a lot of time and research. Fortunately, the Terminal Equipment Automated Management System (TEAMS) environment already contains a realistic emulator that has been proven to work within container terminals all over the world. This TEAMS emulator will be used for this research. Additionally, communication between the learning based path planning algorithm and the emulator was built for this research.

4.1.1. Choice of highway

This research is focused on the improvement and scientific academic innovation of the path planning software of container terminals. In the environment at TBA, the AGVs use highways to get to their destination. Highways are defined as the paths that are available for the AGVs to take that are parallel to the quay, as shown in figure 4.1. The AGVs use these to guide their path to and from the buffer area and the stacking area. The moment when the highway is entered and the choice of which highway is taken influence the performance of the terminal.

The current behaviour of the AGVs The paths of the AGVs are determined by a path planning algorithm designed by TBA, based on the A* algorithm. This algorithm is described in subsection 3.2.

AGVs can depart either from the buffer area or the stacking area, each with their own preferred highway. AGVs departing from the buffer area prioritize the highway that is closest to the buffer area, whereas AGVs departing from the stacking area prioritize the highway that is closest to the stacking area. Each highway has a fixed driving direction for the AGVs.

4.1.2. Order generator

In the real world functioning of TEAMS, scheduling is input from another part of the software. Because this is too complex for the scenarios that will be tested, a script is created to generate orders for the AGVs. These orders will send the AGVs from the buffer to the stacking area and back.

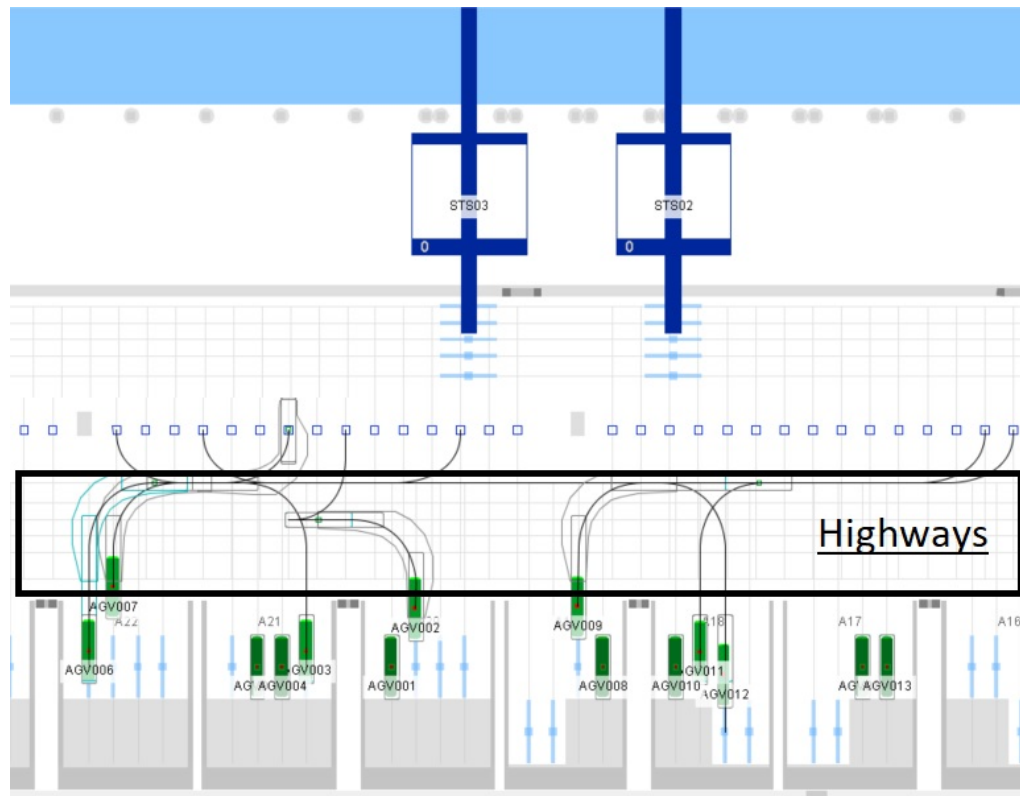


Figure 4.1: Area with the highways, numbered bottom up

4.2. What data from the automated container terminal emulator can be used for the algorithm?

The state of the system can be defined by the following real-time state variables (observations) for each AGV, which can be extracted from the emulator:

1. Position x of each AGV with respect to a point in the terminal;
2. Position y with respect to a point in the terminal;
3. Orientation in degrees;
4. State of the order: executing, valid or finished;
5. Destination code

4.3. What are the key performance indicators?

One of the objectives of the container terminal is to have a high productivity: to make as many movements as possible within a certain time frame. There is a risk involved when this is the only Key Performance Indicator (KPI): it can make the container flow of the terminal unstable. An unstable container flow is defined as any scenario where the AGVs are not optimally utilized. An example of this is when AGVs with the shortest path will get priority over AGVs with a longer path. As a consequence, the amount of completed orders can be high because of the high frequency of the short-path AGVs, but meanwhile, the longer-path AGVs are compromised. In this case, the KPI is fulfilled, but the container flow is unstable.

In order to prevent an unstable container flow, a second KPI is added: *Container flow: The lowest number of completed orders by an AGV.* For the Productivity KPI and the Container flow KPI, the target value for the proposed algorithm is to be within 15% of the value for the current TEAMS algorithm or higher.

4.4. How will the scientific academic innovation be determined?

The current state of the art algorithm where a path planning algorithm is guided by machine learning was created by Liu et al. in 2019 [23]. Their strategy combines the A* path planning algorithm with a reinforcement learning strategy. No papers that use machine learning to benefit A* path planning have been published ever since.

In this paper another approach of path planning guided by machine learning is suggested. In order to determine the scientific academic innovation of the current research, the proposed algorithm is compared to the Liu algorithm. Furthermore, the innovative element is tested by experimenting as described in section 4.5 and comparing the suggested algorithm with and without this element.

4.5. How will the value for TBA be determined?

The proposed algorithm will be tested using experiments, the setup of which will be extracted from challenges in container terminals. This section describes the experiments which are used to test the scientific academic innovation and value for TBA.

4.5.1. Challenging situations in the container terminals

In order to structure the container terminal, each highway has been assigned a direction by the programmer based on the test results of the emulation of the terminal. The one-way direction of the highway is chosen to keep the terminal more stable, while also avoiding congestion and deadlocks. The advantage of the philosophy of one-way driving on the highway is shown in figure 4.2. The AGVs avoid each other and AGV002 takes the shortest route.

However, in some situations this policy is not beneficial, and instead, provides challenges. Such situations are described in the next paragraph.

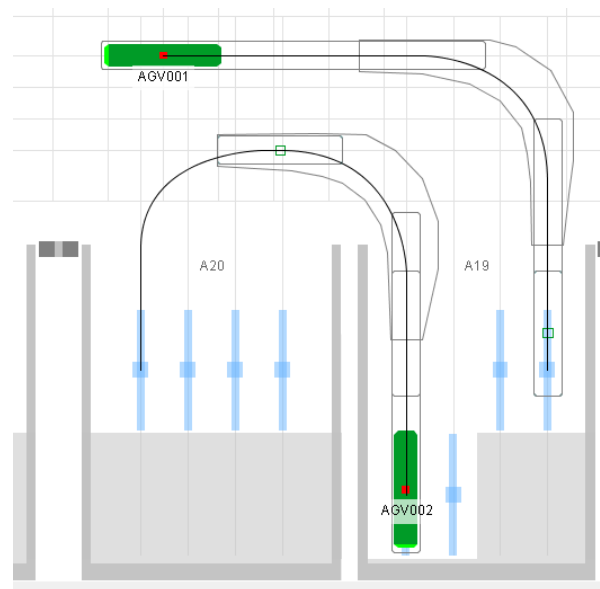


Figure 4.2: Advantage of the highway directions

Disadvantage of the set highway direction Having a set highway direction is not beneficial to the paths taken by the AGVs. In figure 4.3 there is a chance that the AGVs will drive in their way and both have to take a longer route due to the direction restriction. In figure 4.4 AGVs are crossing each other.

Another example is shown in figure 4.5 where AGV002 has to wait for AGV001. Lastly in figure 4.6 both AGVs are in a position where one has to wait for another.

In all of the four examples given, a different policy could increase the efficiency regarding the order completion time. Therefore it is desired to design the experiments in such a way that it is likely for these situations to occur. This way, the proposed algorithm is tested as to how it copes with these challenges. These setups are described in the following section.

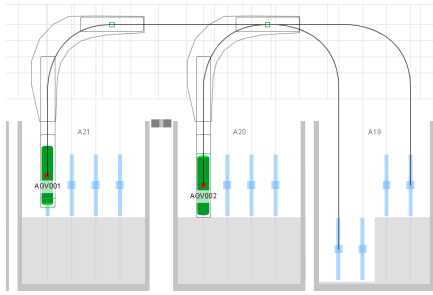


Figure 4.3: Long route example 1

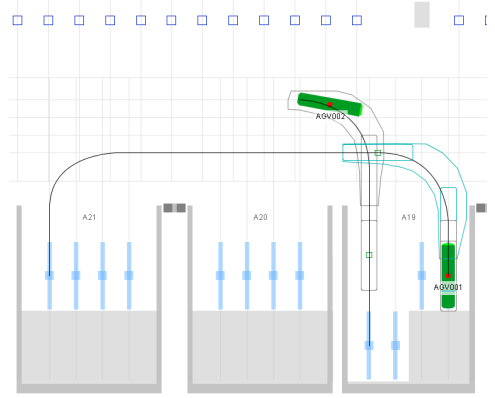


Figure 4.4: AGVs crossing example 2

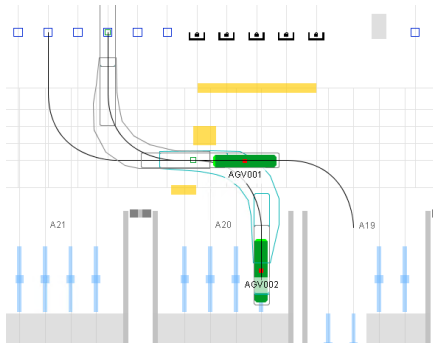


Figure 4.5: AGVs crossing example 3

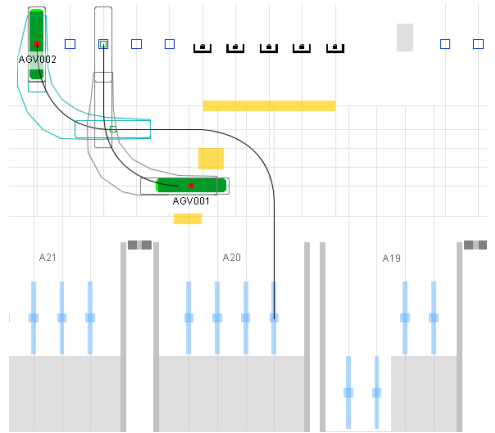


Figure 4.6: AGVs crossing example 4

4.5.2. The experiments

To show that the innovative algorithm is capable of applying Reinforcement Learning to create vehicle aware input data for an A* Path Planning (PP) algorithm in a container terminal environment the experience of TBA is used. They come across challenges where the path planning can be more efficient.

Environment The environment where the algorithm is tested is a part of a container terminal controlled by software of TBA, which consists out of four buffer and stacking areas. Figure 4.7 and 4.8 show an abstract depiction and a picture of the environment. The top area is the buffer area and the lower area is the stacking area. Within this environment, the AGVs have access to two highways, which are indicated by the yellow arrows and lines.

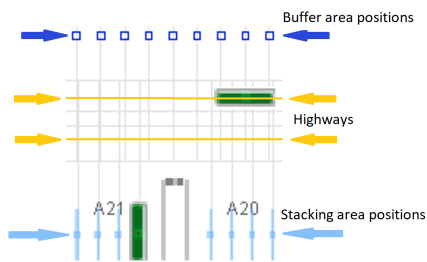


Figure 4.7: Environment with AGVs, highways are highlighted



Figure 4.8: Picture of the environment

Two scenarios are used to compare the proposed algorithm to the TBA algorithm and the Liu et al. algorithm. In both scenarios, two AGVs have access to two highways. This leads to either both AGVs choosing their own highway or both choosing the same. Figure 4.9 shows scenario A, in which the red circles indicate the start and end point of one AGV one and the green circle for the other AGV, AGV two. The desired situation is that both AGVs learn to drive their own highway and as a consequence never cross.

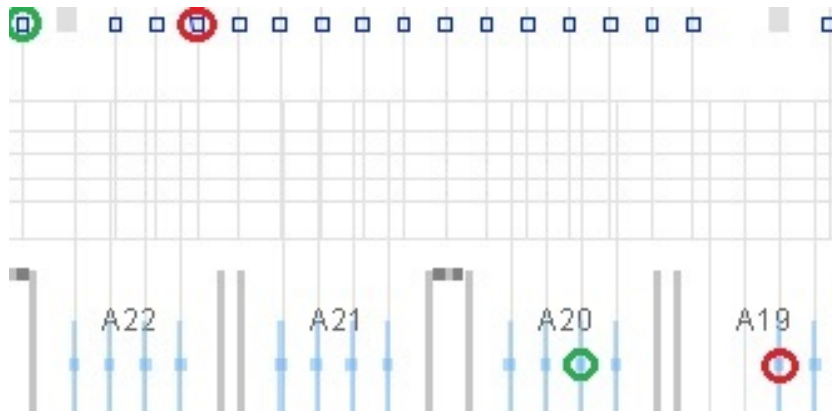


Figure 4.9: Scenario A

In scenario B, the desired situation depends on the state of the AGVs, such as their location and destination. In figure 4.10 displaying scenario B, the red circles have the shortest route when the AGV takes the highway closest to the starting point. However, this is only preferable when there is no breaking involved to give priority to the other vehicle.

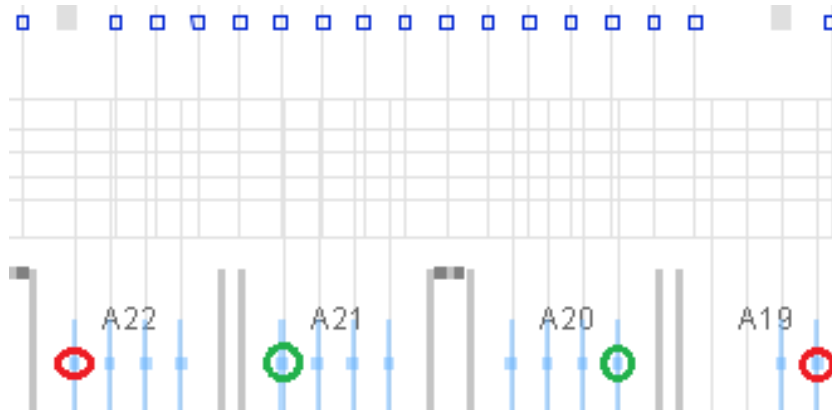


Figure 4.10: Scenario B

Table 4.1 displays the amount of AGVs in the experiment scenarios and their start and end point. This will very likely result to the challenging situations in container terminals as described in subsection 4.5.1.

Scenario	AGVs	Start/endpoint 1	Start/endpoint 2
A	2	Stacking areas	Buffer Area (BA)
B	2	Stacking areas	Stacking areas

Table 4.1: Test scenarios reinforcement learning script

AGV behaviour The boundaries of the AGV behaviour in the environment are as follows:

1. The AGVs will ride without load;
2. The AGVs will ride between predefined locations;

3. Two highways are used;
4. One highway is used per order;
5. The number of AGVs is defined for each test;
6. The only vehicles the AGVs interact with are one other;
7. The behaviour of the AGVs regarding cornering and speed is defined in TEAMS;
8. The behaviour of the AGVs regarding cornering and speed is defined in TEAMS;
9. The maximum speed of an AGV is 6,7 m/s;
10. Each order has the same priority.

4.6. What RL methodology suits best for this case?

The section will answer 3: *What RL methodology suits best for this case?*. During the literature research different machine learning algorithms were found.

4.6.1. RL algorithms used for path planning

As discussed in chapter 3, two main types of reinforcement learning were used for path planning: Monte Carlo learning and Temporal difference learning. Both are briefly discussed below.

Monte Carlo Learning Lonza [24] states: "Monte Carlo methods are a powerful way to learn directly by sampling from the environment." For each state in the trajectory, an action is chosen based on a policy. This policy is created by pre-set parameters and formulas calculating the reward based on the reaction in the environment. For each action a new reward is calculated and when the final state is reached, the value of each action is calculated. When using Monte Carlo Learning, the final reward at the end of the episode is required to calculate the value of each state. Therefore, this method will converge to the optimal policy if the state-action pairs are visited of all trajectories towards the end point. An advantage is that it is not required to know the probability that a state will go into another, e.g. when using dynamic programming, the algorithm will not converge into a policy without the knowledge of state transition probability. Monte Carlo learning will converge by gaining knowledge from the environment, using calculations for the state values from this data.

Monte Carlo Control A variant of Monte Carlo Learning is Monte Carlo Control. This algorithm is used to approximate optimal policies. The values learned are used for the decision of the action taken [34]. In figure 4.11 the basics of Monte Carlo Control are shown, where: π_n is the policy, E is the episode, q_π is the action value of the complete episode, I is a complete policy improvement. By following this method, the optimal policy π_* can be found.

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} q_*,$$

Figure 4.11: Loop of Monte Carlo Control [34]

One disadvantage of Monte-Carlo Learning and Control is that the value is only known at the end of the episode. When used in environments with long learning trajectories or episodes without an end, other algorithms such as Temporal-Difference (TD), explained in paragraph 4.6.1, are advised.

Epsilon greedy algorithms: when the action is always based on the highest state action value, there is a risk involved in large environments that the optimal policy will never be discovered. Therefore an element ϵ can be added to balance between exploration and exploitation. Algorithms that include the epsilon greedy element pick a (pre-set) value of the ϵ , which influences the amount of actions chosen based on randomness. This way the agent can discover a new policy or can confirm that the action is indeed one of low value.

Temporal-Difference (TD) Learning TD Learning can adapt the state value at every time step. This way the policy can be adapted during each learning episode, in contrast to the Monte-Carlo (MC) methods which need to visit all states of the episode and learn after the completion. The TD learns during the episodes by an estimation of the state values. This process defined as pping: "estimates of the values of states based on the value of successor states" [34]. The time interval in which this estimation is done differs per algorithm and is optimal if a stable state is reached.

A disadvantage of TD Learning is that the values the policy converges to are always based on an estimation of the true value. This way the learning can be unstable and never converge to an optimal policy.

The advantage of these methods is that when a stable setting is found, they can converge faster to a optimal policy, seeing as it learns during the episode.

Examples of methods using Temporal difference are SARSA and Q-Learning [34].

4.6.2. Reinforcement Learning aspects

In the literature review and the previous section, three aspects were found to categorize reinforcement learning algorithms. Table 4.2 summarizes these aspects and their corresponding options. The following section argues which of the options is chosen for each aspect.

Aspect		
A. Learning method	Model Free	Model based
B. Policy	On-policy	Off-policy
C. Epsilon-Greedy	Yes	No

Table 4.2: Aspects used in reinforcement learning algorithms

Learning method: Model free The model free approach is the best option for this case, as it unknown how the AGV will respond in the states that will be created in the environment. The biggest advantage of the model free approach is the ability to learn in different kinds of terminals, which eliminates the need for a model to be built for each terminal. The challenge will be the converge speed, as we do not know all the behaviours of the environment beforehand.

Policy: Off-policy During the training phase, there is a choice between an on-policy or off-policy method. As S.Sutton and G.Barto describe it, "the distinguishing feature of on-policy methods is that they estimate the value of a policy while using it for control. In off-policy methods these two functions are separated. The policy used to generate behavior, called the behavior policy, may in fact be unrelated to the policy that is evaluated and improved, called the target policy" [34]. For example, during the training phase, the behaviour policy might be to explore multiple solutions, while the target policy is to find the optimal solution. Although off-policy algorithms are less stable than on-policy, off-policy methods make use of the new data in every iteration, as the old policy is not used for the next iteration. In case of this experiment, the Q-values of the future are important as it is expected that the policy will develop during the learning phase. This is based on the new expected states that will differ from the policy and the reward during the interaction with the A* algorithm.

Epsilon-Greedy: Use Epsilon-Greedy policy The Epsilon-Greedy policy will allow the algorithm to (re)discover the effects of the actions in the learning phase. The Epsilon-Greedy will not be used during the execution of the algorithm after the learning phase.

4.6.3. This case: learning-based path planning for AGVs in container terminals

As the proposed algorithm will be combined with A* path planning, TD Learning is not an option, because the A* needs stable input. Neither is Monte Carlo Learning, because the environment of the container terminal is too large. Therefore, the most suitable methodology would be the Monte Carlo Control. This method includes all preferred aspects discussed above. The disadvantage of the long learning trajectories will be taken into account when designing the algorithm.

4.7. Conclusion

This chapter has discussed research question 3: *What RL methodology suits best for this case?* The most suitable methodology for this case is the Epsilon-Greedy Monte Carlo Control.

5

A Markov Decision Process

This chapter describes the modelling of the algorithm. The model is developed using the choices and conclusions of chapter 3 *Literature Review* and chapter 4: *Experimental Setup*. The goal of this chapter is to describe the algorithm forming the base for the implementation of the code.

First MDP will be explained and why it is important. Secondly the environment will be discussed. Based on this the research question 4: *What is the MDP for this case?* is answered. Combining this will lead to the flow chart diagrams that will be used during the implementation of the code.

5.1. Markov Decision Process

The Markov Decision Process (MDP) is a mathematical framework that can be used for decision making. It can be solved by RL or Dynamic programming. It specifies how the environment is transitioning with respect to the different actions that are taken.

The use of the Markov Decision Process The theoretical framework for the RL is the MDP. It contains the information used for the goal to find the policy function that the agent will use in every state to decide its next action. For most real world problems the MDP is unknown or known but too big to be used so an approximation is made.

The Markov Decision Process Contains the description of the process by four main variables:

1. Finite set of states S

2. State transition probabilities \mathcal{P}

Equation 5.1 represents the state transition probability matrix $\mathcal{P}_{ss'}^a$. The content of the matrix defines the probability of the current state s ending up into one of the successor states s' when action a is taken. $S_{t+1=s'}$ is defined as the possible successor states s' . From the current state s when an action a is taken from the action set A_t .

$$\mathcal{P}_{ss'}^a = P[S_{t+1=s'} | S_t = s, A_t = a] \quad (5.1)$$

All information that the agent needs to know about the future behaviour of the environment has to be in a state. When it does, the past can be disregarded and this is called the *Markov Property*. Every reaction to a action in a state is described by the \mathcal{P} [34].

3. Finite set of actions A

4. Reward \mathcal{R}

The reward function \mathcal{R}_s^a , shown in equation 5.2, is the expected reward of the current state for a certain action $A_t = a$.

$$\mathcal{R}_s^a = \mathbb{E}[\mathcal{R}_{t+1} | S_t = s, A_t = a] \quad (5.2)$$

5.2. Environment interaction

The TEAMS environment is used as an emulator during the experiments. Three important reasons for this decision are the fact that it is an emulator that is already there, it is well developed and shown to be realistic as it is used in real ports and lastly the emulator can be used to generate data to compare the Key Performance Indicators with the contemporary algorithm.

During this research the order generator Terminal Operating System (TOS) is not used. The system is too complex for the orders needed for the testing of the algorithms. Therefore an order generator is designed that replaces the TOS part in the environment. As a result, there are four main systems that work together: The order generator, emulator TEAMS, a path planning algorithm and a RL algorithm.

5.3. Markov Decision Process definition for the case

This section described the MDP for this case.

5.3.1. States

It is important for a reinforcement learning system to choose the state variables thoughtful. This is due the fact that for every added state variable that does not add value to the agent it will slow down the policy learning process. The state of the system is defined by the following real time updated variables for each AGV:

- **Position x** of the AGV requesting its state
- **Position y** of the AGV requesting its state
- **Destination x** of the AGV requesting its state
- **Destination y** of the AGV requesting its state

For the environment of the AGV:

- **Position x** of AGV_n
- **Position y** of AGV_n
- **Destination x** of AGV_n
- **Destination y** of AGV_n
- **Orientation** of AGV_n

Orientation The orientation can give a lot of extra information: there is a high possibility that the AGV is driving in a certain direction or is taking a corner which influences the driving speed as well. The information about the orientation is split into less than the full 360 degrees. but for example 4 different orientations. The reason for this is that precision will not benefit the system but increase the complexity. The orientation is not included for the AGV which request its state as these will always start at the buffer or stacking area. To be able to park on this position, the AGV must be in a certain orientation and will therefore not add any value to the policy learning.

Degrees AGV	Code
315 till 45	0
45 till 135	1
135 till 225	2
225 till 315	3

Table 5.1: Code to Degrees

Enlarging the number of states with for example more angles, makes the algorithm more complex and might slow down the learning. Though, when doing so, the information where the agent bases its action on is more detailed.

Table 5.2: State of the AGV

$(Sx, Sy);$	<i>/* x,y position of AGV requesting route */</i>
$(Ex, Ey);$	<i>/* x,y target of AGV requesting route */</i>
$(Cx_n, Cy_n);$	<i>/* x,y position of AGV_n */</i>
$(Ex_n, Ey_n);$	<i>/* x,y target of AGV_n */</i>
$O_n;$	<i>/* Orientation of AGV_n */</i>
<i>Combining the elements to the state:</i>	
$((Sx, Sy), (Ex, Ey)(Cx_1, Cy_1)(Ex_1, Ey_1)O_1... (Cx_n, Cy_n)(Ex_n, Ey_n)O_n);$ <i>/* State */</i>	
$((2, 1), (3, 4), (1, 1), (4, 3), 0);$ <i>/* Example state of a two AGV environment */</i>	

The maximal value of n is the total number of AGV's -1. The count is sorted based on the position of the nearest AGV based on Euclidean distance. When there are AGVs that are exactly at the same distance from the AGV for which the route is calculated, the name of the state is sorted secondly on the closest end positions with respect to the end position of the AGV for which the route is calculated

5.3.2. Actions

The environment of the port is unknown and it is important that the path planning algorithm gets the right data to base its route on. A heuristic function of the A* algorithm as the Manhattan method described in chapter 2, will point the path planning algorithm effectively into the direction of the goal. Therefore, it is not chosen to actively influence this with a machine learning action. The gain can be found by finding the correct route based on the knowledge about the travel cost of the terminal. As a consequence the action of this machine learning algorithm is to learn to pick the best possible cost matrix where the A* algorithm will base the route on. The amount of possible actions keeps on growing during the learning phase as it is an epsilon greedy algorithm. The actions which are added are generated from the last result of the measured cost during the specific state are memorized. The algorithm will choose this action in case it meets the ϵ -greedy an GA conditions. The value of these parameters are set before the learning phase.

Algorithm 1: Action

```

 $\epsilon \leftarrow rand(0,1);$       /* Setting for random or generated actions between 0 and 1 */
 $GA \leftarrow rand(0,1);$     /* Setting for generated actions between 0 and 1 */
 $MA \leftarrow False;$       /* If 1, time spent in each part of the terminal is measured and
added as an action for the state */
if  $NA \leq NH$  then
   $MA \leftarrow 1;$       /* NA is the Number of actions for the state. NH is the number of
highways in the environment */
  if  $NA = 0$  then
     $A_{state} = 0;$       /* The zero matrix is assigned so a path based on the heuristic
is chosen */
  end
end
if  $(n < \epsilon)$  then
  if  $(n < GA)$  then
     $A \leftarrow \max Q(S);$       /* Cost matrix with highest Q value for this state */
     $MA \leftarrow 1$ 
  end
  else
     $A \leftarrow A_r;$       /* A random cost matrix linked to this state */
  end
end
else
   $A \leftarrow \max Q(S);$       /* Cost matrix with highest Q value for this state */
end

```

5.3.3. Reward

The goal of the algorithm is to estimate the cost of the nodes as accurately as possible. If the algorithm finds a cost matrix that is very close to the values in the environment, the reward should be high. Therefore the closer the algorithm reaches this action, the higher the reward will be when a reward function is chosen that divides the difference. This function converges to unlimited if $dt = 0$ as 1 is divided by dt . For this reason if dt equals to 0, the value of dt is set near the lowest possible value in a float.

Algorithm 2: Reward R

```

T ← ∑ |(tmm(m, n) - tam(m, n))|; /* Time difference between estimation and measurement
*/
tmm, tam; /* time spent at area, mm is measured matrix and ma action matrix */
m = 1, 2..M; /* with M the number of x grid lines */
n = 1, 2..N; /* with N the number of y grid lines */
if (T = 0) then
  | T = 6 * 10-4; /* Near the lowest possible value in a float */
end
R ← 1/|T|; /* The closer to the solution, the higher the reward */

```

5.3.4. The value of the action: the Q value

The episodic tasks of the determination of the cost matrix consist of one step. The destination is independent of the action of the AGV. The action has influence on the accuracy of the cost matrix and therefore on the path the AGV takes. Because the actions the AGV takes into the future does not influence the accuracy of this action, and therefore the value of the chosen action, they are not taken into account. The value of gamma is set to zero which is shown into algorithm 3. The Q value in this algorithm is determined by:

Algorithm 3: Q value

```

if (s ∉ S) then
  newline Q(s, a) ← p; /* If the state action pair is never visited before, it is
  initialized with p>0. This makes sure that if new actions are chosen by the
  epsilon greedy part of the algorithm, they have more chance that they will
  converge to their real value with respect to Q initialized = 0 */
else
  | Q(s+1, a+1) ← (1 - α) * Q(s, a) + (α * R(s, a))
  | ; /* The closer to the solution, the higher the reward */
end
end

```

Where:

α is the Learning Rate

$R(s, a)$ the reward

$Q(s, a)$ the expected value of the quality of a certain action in a given state.

Learning rate α As the environment is explored, information about in the environment can change over time, so do the rewards of the actions. The higher the Learning rate α , the more aggressive the algorithm will react to changes in the value of the last reward. Because of the unknown environment we want the Q value to adapt relatively quickly, without the system becoming unstable.

6

A Reinforcement Learning Based A* Algorithm

This chapter describes the design and implementation of the Reinforcement Learning Based A* Algorithm. First the overview and the choices in the top level are described. Then for each part of the algorithm a detailed flow chart is given.

The overview of the algorithm The algorithm consist out of nine phases: the initialization, checking availability of an AGV, translation of the measurement to the cost matrix, calculation of the action value, observation of the current state, action selection, route calculation, communication with the emulator, and when the set time is reached, storage of data. Figure 6.1 shows how these parts of the algorithm are connected. Each of these phases consist out of multiple steps, which are described in further detail in the sections below. The complete flowchart with all of the detailed steps is included in appendix A.1

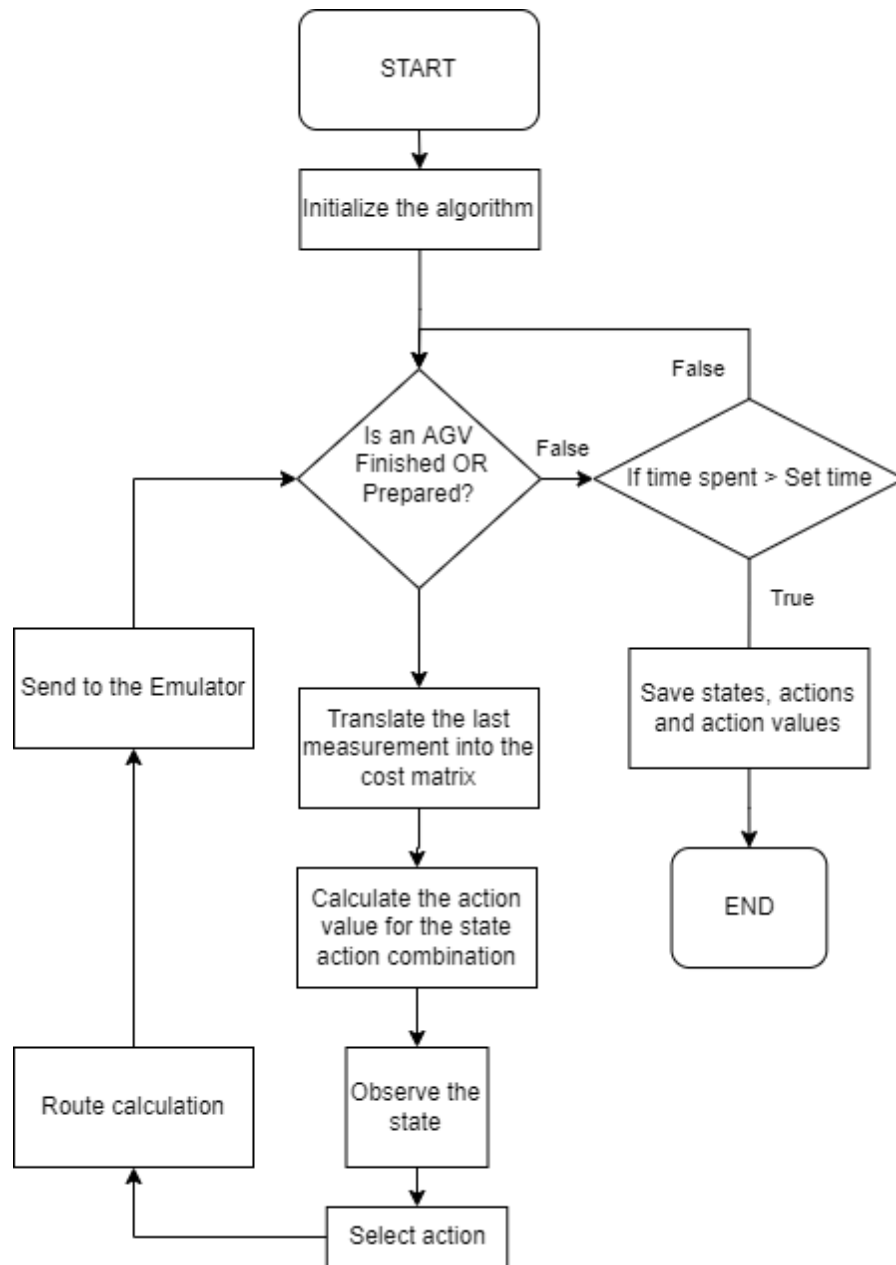


Figure 6.1: Top level algorithm

6.1. Initialize the algorithm

Figure 6.2 shows the steps taken during phase 1: initialization of the algorithm. Each of the steps are further explained below.

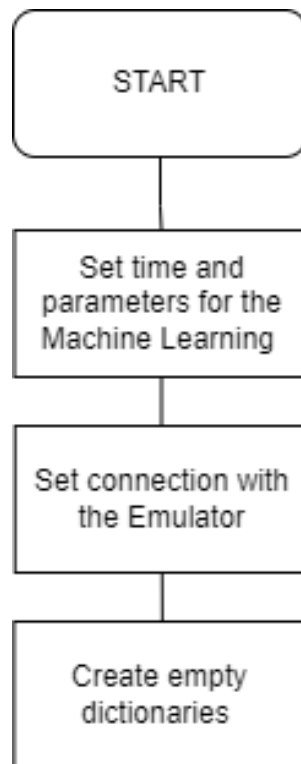


Figure 6.2: Phase 1: Initialization of the algorithm

6.1.1. Set time and parameters for the Machine Learning

The algorithm is flexible in the environment it can operate in when the boundaries of the environment are given. This is gathered in initialization phase: the user customizes the areas in which the AGVs can travel, i.e. what parts of the area are unreachable.

For each machine learning algorithm certain parameters can be set. For this algorithm it is the Learning rate α , the number of new generated actions, abbreviated by GA , the epsilon greedy level ϵ and the simulation/learning time.

The environment matrices The path planning algorithm cannot process the raw data. This is why the the operating environment is translated into an environment matrix, based on the shape of the terminal. The matrix has two purposes: first, it is used to describe the positions of the AGVs as states in a position matrix. This is used to determine which positions are available for the AGVs for their route. Second, it is used to describe the cost for each part of the quay in an cost matrix. Each node in the matrix is assigned a travel cost based on the measurement by the system. This is used by the path planner to determine the route. As there are no measurements yet in this phase, the costs are initialized at zero. The main task of the RL part of the algorithm is to create an accurate estimation for the travel cost.

As the matrix grows, so does the data used for machine learning. Therefore a smart distribution of the area that is represented by the rows and columns of the matrix benefits the performance of the algorithm.

6.1.2. How to set the x and y distributions of the cost and position matrix

The algorithm can process the data faster if the amount of data is limited. To limit the amount of data used by the algorithm the matrices should be designed carefully. First the y-direction is chosen and after that the x-direction.

Row distribution of the matrix: y- direction We first take a look at the behaviour of the AGVs when randomly driving around to see where it is useful to apply limitations of the horizontal lines. AGVs were waiting at their parking positions and the when an they had to wait for another AGV driving on a highway they did so after driving a few meters. To measure this the positions of two horizontal lines were chosen close to the parking position. The last horizontal line is positioned between the highways.

The state of the system is a snapshot of the system. Information about the AGV when it is at rest or accelerating from its rest position is essential because the AGVs will spend a longer time at this position due to the relative lower speed with respect to other positions of the quay. Therefore a more narrow line is chosen between the buffer and stacking area positions and the first line.

Seeing that there are two highways, the choice is made to separate the highways exactly into the middle. Each highway has the same amount of length between the middle grid line and the highway. Distinguishing AGVs that are performing an action or stationary, just started an action is the balance between amount of data and precision. Therefore the final horizontal y-rows are chosen as shown in figure 6.3.

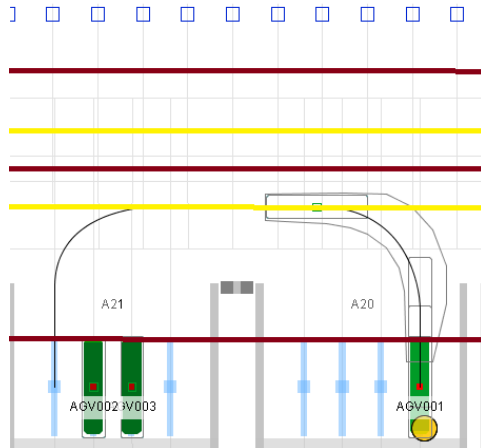


Figure 6.3: Horizontal lines separating the Y positions

Column distribution of the matrix: x- direction Since the stacking areas at the bottom of figure 6.4 and buffer areas at the top of the figure are not exactly lined up to each other one is chosen as a base for the x distribution. As a consequence the stacking areas are chosen as the base. The distance between the lines is based on the stacking positions. Due to the fact that the model needs to be able to distinguish whether an AGV is almost at its destination, it can do so combined with the angle of the vehicle. The final grid shown in figure 6.4 is aimed to have the right balance between precision, reliability and data sufficiency. It is expected to provide just enough information to the model about the positions of the AGVs when a state is asked, and the time spent at the positions of the quay.

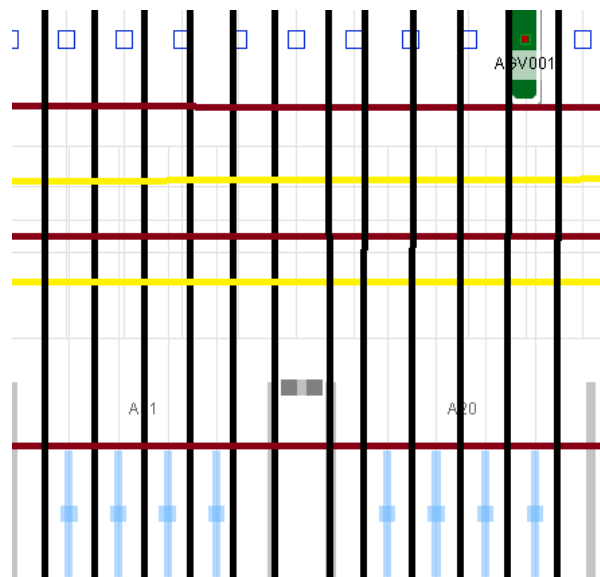


Figure 6.4: Final Grid

The x and y lines are translated into a matrix. Figure 6.5 shows how this is done with the corresponding x and y direction as in the example.

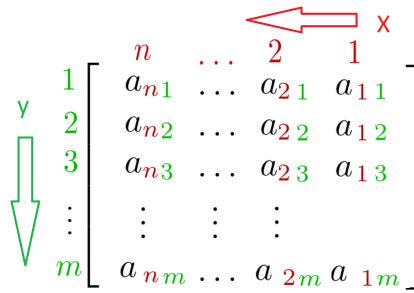


Figure 6.5: Example of the matrix with the directions as in the container terminal

6.1.3. Connection with the emulator

Via Open Database Connectivity a connection is set where via a SQL Server, data is communicated between the algorithm and emulator.

6.1.4. Creating empty dictionaries

The algorithm creates a place in the memory where the cost matrices, actions, states, Q values are stored. For this purpose, dictionaries are created or emptied when the algorithm is started.

6.2. Availability

Figure 6.6 describes how the algorithm knows whether an AGV is available for an action. This is the case when an AGV has completed its route or when the AGV never had an order before. Then its status is Finished or Prepared, which allows the algorithm to continue on to phase 3: create cost matrix.

If the time that is set by the user has passed, the algorithm will save the data and end the algorithm as described in section 6.9.

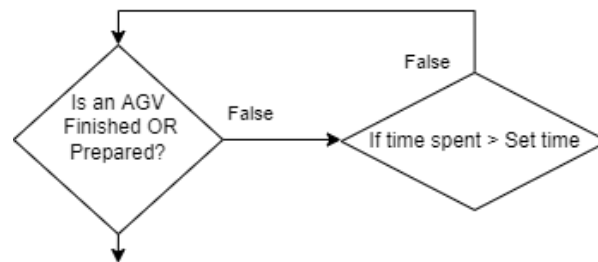


Figure 6.6: Phase 2: Availability of AVGs

6.3. Create cost matrix

This section describes how the cost matrix is created. The definitions of the cost matrix are already given at the initialization phase. When an AGV has never executed an order before, there is no measurement data which the algorithm can process. Therefore in this case the measurement is done as if all cost were zero. After that the state is determined. The state of the system describes the positions of the AGVs and their orientation. For each state, there is a unique cost matrix. If there is no measurement yet, the state of the system also needs to be determined here.

When there has been a previous order, the raw data measurement of the last executed order is translated into the cost matrix by adding up the amount of cost, for example time spent into a position of the quay. When the cost matrix is created the algorithm moves on to phase 4: action value calculation.

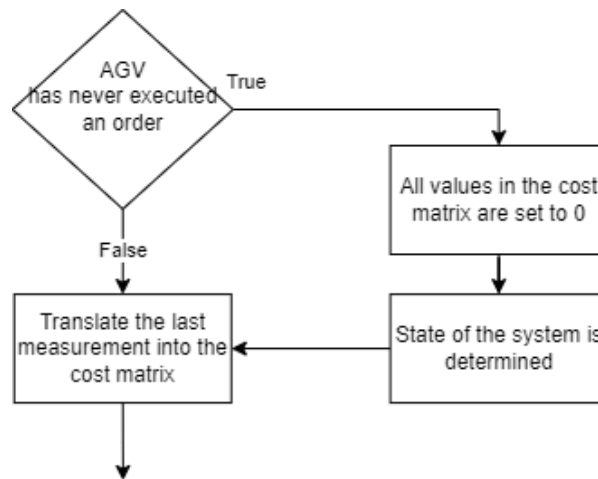


Figure 6.7: Phase 3: Create cost matrix

6.4. Action value calculation

This section of the algorithm calculates the action value of the previous state action combination. The action value is often referred to as the Q-value and indicates the value of the action in the particular state.

During the previous cycle of the algorithm there was determined if the data gathered during the previous action was an Epsilon greedy action. As described in section 6.6 this is determined by a random generated number and, if this is higher than the epsilon greedy value set by the user an epsilon greedy action is executed.

If the epsilon greedy action is true a new action number is connected to a cost matrix which is based on the last measured data of the AGV during the latest state. In every state it is essential to know the cost of the environment to be able to plan a route based on the complete environment in that particular state. Therefore actions that do not have a value included to all highways, will be assigned a low action value. This way the action with the most data will be picked. Until the learning phase for this state is finished and the start action value is assigned to the state action.

When there is no epsilon greedy action picked the reward and action values are calculated as described in chapter 5.

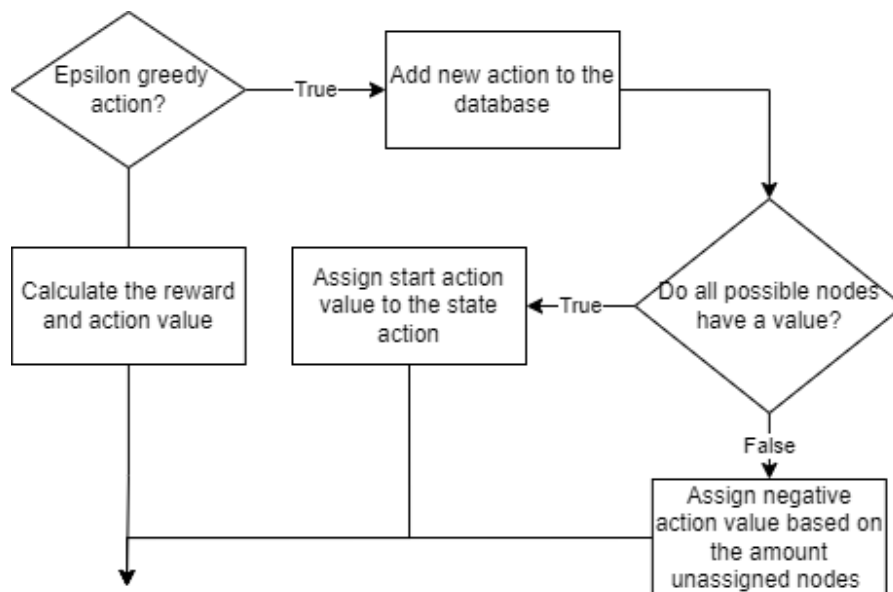


Figure 6.8: Phase 4: Action value calculation

6.5. Observe the state

First the state is determined as shown in figure 6.9. The state consists out of 4 variables for the main AGV plus 5 variables per extra AGV. This is the current x and current y position of the AGV, which is measured from the emulator, the x and y destination of the AGV, this requested by the algorithm via the given order. The orientation is not included as the AGVs always start from a position which has a fixed orientation, this information does not add any extra value for the algorithm. The five variables per remaining vehicle are their current x,y position and x,y destination and their orientation. They are sorted by the euclidean distance between the vehicles and the AGV requesting the route. In this way there will never be a description of multiple states representing the same situation. When the state is unknown it will be created and stored in the state dictionary. This dictionary will be saved afterwards so that the reinforcement learning can be used for the path planning. Once the state is known the algorithm moves on to phase 6: select action.

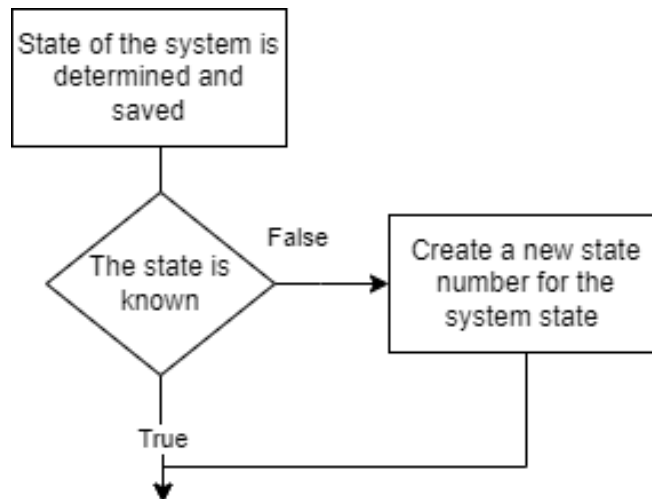


Figure 6.9: Phase 5: Observe the state

6.6. Select action

Figure 6.10 shows the selection of the action. There are four main ways to determine the action of the algorithm.

Action for a never visited state The goal of this part of the algorithm is to determine which cost matrix will be sent to the path planning algorithm. If the state the AGV is in has never been visited before, the first state action combination that will be used is the zero matrix.

Action for a state that is visited less than the number of highways When an AGV travels a path, the data of this is saved into a the action matrix used for the choice of the path. This action matrix is the latest action used for the state. Each element related to the path the AGV just travelled, is replaced by the measured values. The rest of the elements remain their value. The first cost matrix used for a new state is the zero matrix. This is done so that the path planning algorithm will take a path that is not taken before due the low cost of this route. As a consequence the part of the matrix that is not travelled remains at zero cost and is very likely to be chosen. This way the cost for all of the highways are at least measured once. Therefore the first n times every time new cost matrices are created and are added to the action library.

A random Action If the state is visited before and the number of actions for the state are equal or higher than the number of highways, the algorithm gets at the epsilon greedy part. Epsilon greedy makes sure that the algorithm sometimes takes an unexpected action to discover the consequences of the action. When an algorithm does not have such a choice it will keep focused on a fixed set of actions, while there may be better ones around. At the initialization part of the algorithm the user decides which percentage of the time an epsilon greedy action is taken.

A generated Action The user can decide a percentage of the time a new action is created when a random action is chosen. This new action is created using the epsilon greedy database where the latest cost matrix is stored related to the state and action. This new action number and cost matrix is then stored into the database.

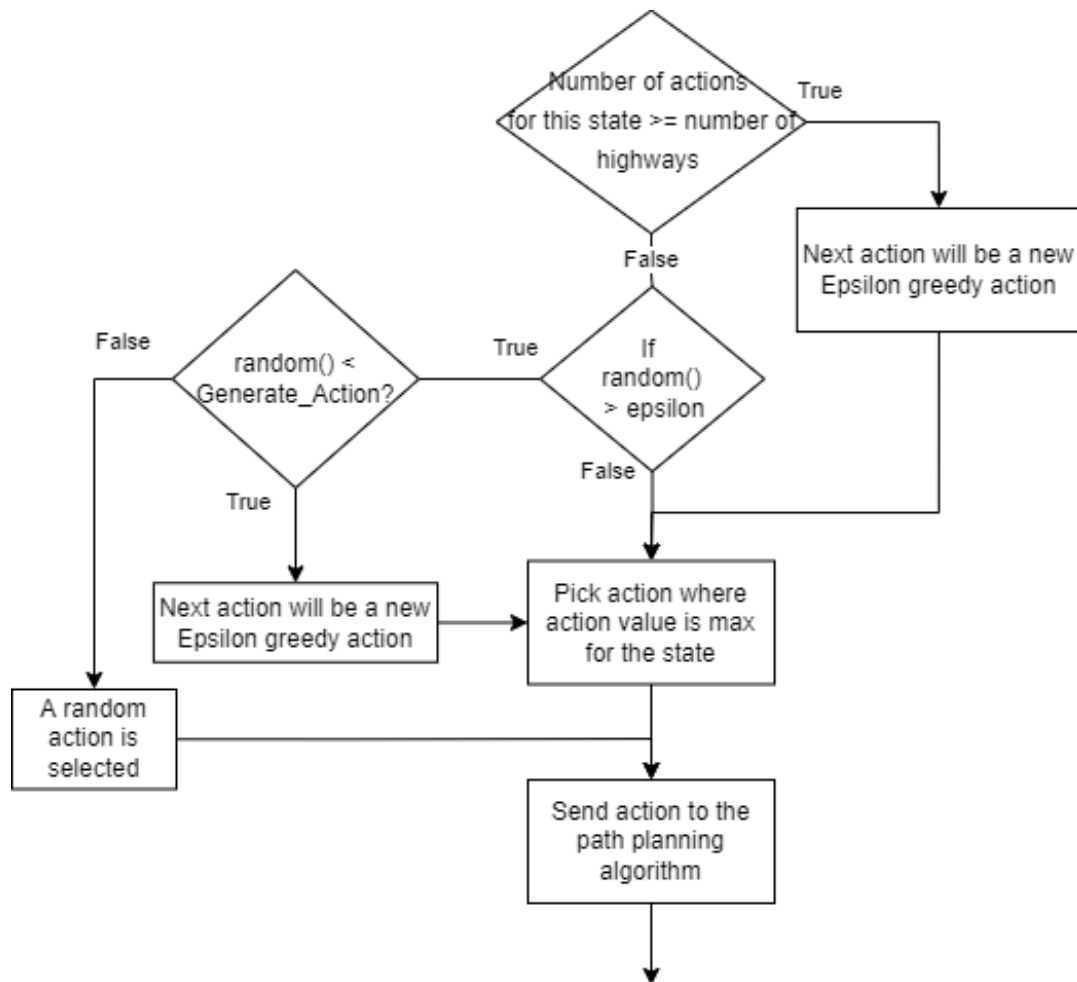


Figure 6.10: Phase 6: Select action

6.7. Route calculation

The selected action in phase 6 is send to the route calculator. In this case the A* algorithm is used with the heuristic function as described in section 6.7.1. In phase 5 is determined whether this action is a random selected action. The goal of a random selected action is that a the AGV will discover a new cheaper route. Though during the testing it often happened that the actions which where generated lead to the same highway route. As this random selected action has the goal to rediscover the best route once the AGVs behaviour start to settle. Therefore, when a random action is picked, the second best route is selected so that the AGV will rediscover a route or confirm that the previous found route is the better one.

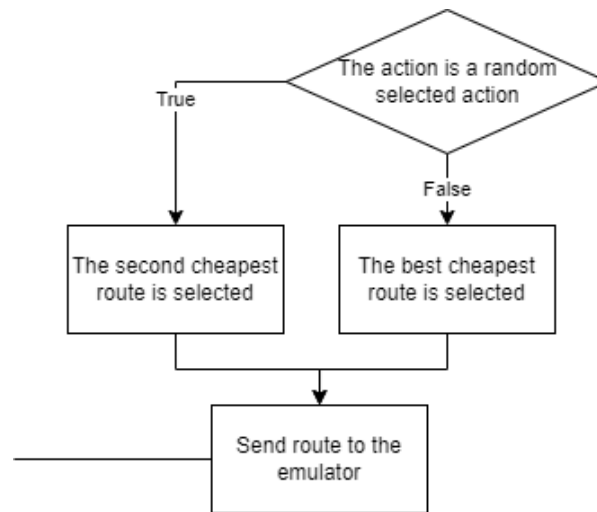


Figure 6.11: Phase 7: Route calculation

6.7.1. Settings of the path planning algorithm

During the experiment it is assumed that paths with more than two corners is always less efficient for the system than a route taking two or less. An AGV changing its highway is rarely beneficial. Therefore, the action of an AGV changing a highway is assumed to be so costly that paths generated by the Machine Learning Path Planning (MLPP) algorithm contains one highway. The A star is limited to generate a path where a maximum of two corners is allowed.

Heuristic function The heuristic of the A* path planning algorithm can be calculated using different methods. This is depended of the way the vehicle can move. Choosing the right heuristic has influence on the speed but also on the correctness of the outcome of the calculation. Another heuristic function is the Euclidean distance which takes a straight line from the current node to the goal. This heuristic is preferred when the vehicle can drive into an straight line towards its goal. Another heuristic function is the the Manhattan distance. When one takes a top view of Manhattan one can see that when driving towards a street which is not in a straight line connected to the current one, one has to take a minimum of one corner. This distance is calculated as shown in equation 6.1

This research is focused on container terminals with highways. Therefore the AGVs cannot drive in straight lines towards their goal as they have to follow the highway grid. The Manhattan heuristic takes this behaviour into account and is chosen as heuristic function for the A*.

$$H(v) = dx + dy \quad (6.1)$$

Where:

$$dx = \frac{|(x_{position} - x_{destination})|}{V_{max(AGV)}}$$

$$dy = \frac{|(y_{position} - y_{destination})|}{V_{max(AGV)}}$$

Admissibility A condition of the heuristic function to be admissible is that it will never overestimate the cost of the node to the destination. Within the algorithm the total cost of the path is given in seconds. Therefore the algorithm will calculate the heuristic by the distance between in x direction divided by the maximum speed plus y direction divided by the maximum speed. If the emulation speed increases, the $V_{max(AGV)}$ does thosen which the AGV will never be able to over is calculated

6.8. Path planning and communication with the emulator

The layout of the quay is very important for the path planning. If a path is suggested which is not allowed, the emulator will give an error or choose one by itself. Therefore the action is adjusted so that the path planning algorithm can never plan a path over the areas that are not allowed e.g. the water.

In the algorithm a module is build so that the output of the path planning gives a suggested highway. This is used during the complete thesis, but can be turned off. An AGV changing its highway is assumed to be not beneficial. Given that, the action of an AGV changing a highway is assumed to be so costly that paths generated by the MLPP algorithm contain one highway. The project boundaries are limited to suggesting one highway per order. This highway is calculated using a path planning algorithm.

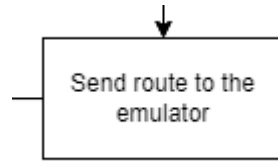


Figure 6.12: Phase 8: Send to the emulator

6.9. Save states, actions and action values

As shown in figure 6.13, if the set learning time is expired the last part is saving all data regarding the state, action and action value. This way the data can be loaded next time and the algorithm can continue controlling the AGVS. Also the data related to the KPI are saved into .csv files with unique names regarding the time, data, and the settings of the machine learning. This way it is easier to process the data later on if needed.

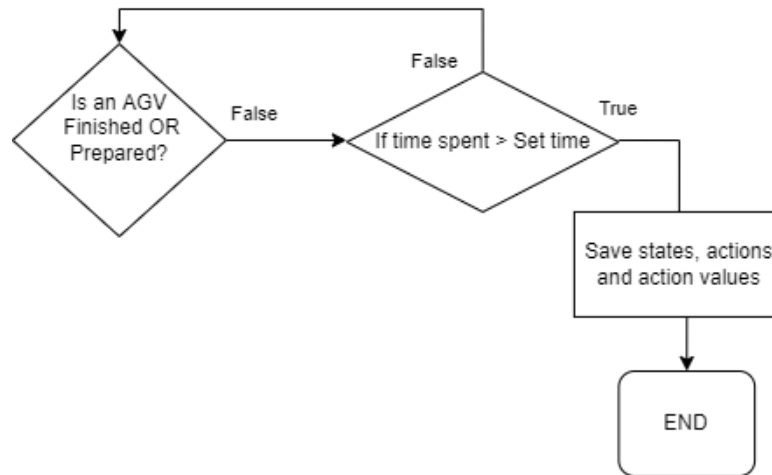


Figure 6.13: Phase 9: Learning phase

7

Experiments and Results

The current research centres around the main question *How can Machine Learning increase the efficiency of path planning for automated guided vehicles at a container terminal?* In order to answer this question, the current research has developed the proposed algorithm as described in the previous chapters. Next, this algorithm will be tested for efficacy. Hence, this chapter answers research question 5: *How does the algorithm perform in the experimental setup?*

First, the in this thesis proposed algorithm Vehicle Aware Reinforcement Learning Path Planning Algorithm (VARLPPA) is compared to the state of the art algorithm Optimized Path Algorithm Based on Reinforcement Learning (OPABRL) designed by Liu et al. (2019), in order to find the scientific academic innovative element. Next, a sensitivity analysis is conducted, then the algorithm is verified and validated.

After this, several experiments are conducted, as described in chapter 4. The KPIs for VARLPPA are compared to the OPABRL algorithm in order to answer sub-question 5a: *How does the algorithm compare to the state of the art machine learning path planning, in terms of the KPIs?* The KPIs for VARLPPA are compared to the TEAMS algorithm in order to answer sub-question 5b: *How does the algorithm compare to the path planning algorithm of TBA, in terms of the KPIs?* Lastly, research question 5 is answered.

7.1. Scientific academic innovation of the algorithm

This section compares the OPABRL with VARLPPA. Liu et al. uses the following phases. Below is description of each phase, the comparison is made to the Vehicle Aware Reinforcement Learning Path Planning Algorithm (VARLPPA), which is shown in italics.

1. Create a grid: the map used in the Liu algorithm is converted into a grid, in order to “clearly identify safe and unsafe areas on which we can plan the path of the intelligent vehicle”.
- This is also done in the VARLPPA by identifying the areas in the grid which the path planning algorithm is not allowed to include in its path determination.
2. Boundary learning process: in this phase, the vehicle drives along the edges between the nodes in the grid, while utilising sensors to scan the environment. This is how it learns the distribution of the obstacles.
- In the VARLPPA algorithm, this pre-learning phase is integrated into the learning phase, by starting with initialized zero cost matrices so the A algorithm will include all zero nodes into its first paths planned. This way the opportunity arises to record cost data for all nodes.*
3. Input start and end point.
This input is given to the algorithm, just like the OPABRL.
4. A priori reinforcement learning training: in this phase, the vehicle explores a reference path: the optimized shortest path algorithm determines the next grid position from the starting point while at the same time, the vehicle scans every node to determine whether it is safe. If this is not the case, the node is marked as unsafe and the previous node is set as the starting point. This cycle continues until the

reference path is determined.

- The VARLPPA does not optimize for every node while driving, but updates the costs after each measurement instead. This is possible because the routes in industrial environments such as ports and warehouses are less variable and often shorter than the ones drove by road vehicles. Furthermore, every lane change requires making a corner, which is always more costly than continuing in the same lane. In the case of the road vehicle, this would be a minor effect on the goal of the system, but in the industrial environment, every action of the individual vehicle affects the goal of the whole system: to move as many vehicles per time unit. Therefore, the action of the vehicle will not be changed during the driving process, which differs from the approach by Liu et al.

5. Machine learning phase: using reinforcement learning to recalculate the value of the path while driving.
 - The VARLPPA collects data from all vehicles in the port to determine the state of the system within a split second. This is not possible in the environment of Liu et al.

The innovative element The most innovative element of the VARLPPA, compared to the design of the OPABRL, is the vehicle aware element: at every state of the VARLPPA, the state of the other vehicles is measured and taken into account, whereas the OPABRL is only used for the path planning of a single vehicle. In order to determine the added value of this element, the experiments discussed below will compare the VARLPPA with and without the vehicle aware element Reinforcement Learning Path Planning Algorithm (RLPPA). In the sections below this is called 'Vehicle Aware' and 'Single Vehicle', respectively.

7.2. Sensitivity analysis

A sensitivity analysis is conducted to examine how the settings of the model influence the outcome of the amount of completed orders during the experiment phase. Therefore four different set ups regarding the parameters are created and compared to each other in a preliminary experiment.

7.2.1. Determining the parameter settings during learning

In this preliminary experiment, the settings of the algorithm are manipulated to see whether this has a significant influence on the results of the algorithm. Four experiments are conducted with different setups, as shown in table 7.1.

The first column shows the setup number for reference. The second shows the learning rate α , the third shows the value for the epsilon greedy ϵ , and the last column shows the setting for the new Generated Action (GA).

Learning rate α The learning rate influences the effect the measured reward has on the value of the action. The higher the Learning rate, the more aggressive the algorithm will react to the value of the last reward. Because of the unknown environment we want the Q value to adapt relatively quickly, without the system becoming unstable.

For setup 1, an α value of 0.1 is chosen. This is relatively low compared to other values in the literature. For setup 2, an α value of 0.3 is chosen. This is relatively high compared to other values in the literature. Between setup 1 and 2 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Epsilon ϵ Finding the right balance between exploration and exploitation is essential in RL. Exploration is finding new information on the environment in order to discover a new and potentially better policy. Exploitation is using the current information for reward maximization by choosing the current action with the highest action value.

The settings for ϵ influences the chance that a greedy action is chosen. When epsilon is 1, all actions are exploitation and therefore never a greedy action is chosen. If the epsilon is 0, always a greedy action is chosen and no actions will be exploited. Therefore, ϵ indicates the balance between exploitation and exploration.

For setup 2, an ϵ value of 0.9 is chosen. This is relatively high compared to other values in the literature. For setup 2, an ϵ value of 0.7 is chosen. This is relatively low compared to other values in the literature.

Between setup 2 and 4 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Generated Action (GA) When the algorithm is exploiting, it measures the action with the highest value. When the algorithm is exploring, it has two options: to explore an existing action that does not have the highest value, or to generate a new action. The setting for GA influences the chance that either option is chosen. The higher the value of GA, the higher the chance a new action is measured.

For setup 3, a GA value of 0.5 is chosen, so the algorithm opts for new actions equally as often as for existing actions. For setup 4, a GA value of 0.3 is chosen. In this case, the algorithm opts more often for existing actions than for new actions. This way, comparing these setups can show the effect of the amount of generated actions on the results of the algorithm.

Between setup 3 and 4 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Setup during learning	alpha	epsilon	Generated Action (GA)
1:	0.1	0.9	0.3
2:	0.3	0.9	0.3
3:	0.3	0.7	0.5
4:	0.3	0.7	0.3

Table 7.1: Parameter settings experiments during learning phase

7.2.2. Parameter settings evaluation time

With the learned policy the agent executes the orders during 30 minutes. The agent is able to change the action values, as alpha is not equal to 0. This is done so that the learned best action will continuously be evaluated. To keep the behaviour changes minimized, the greedy actions are taken out of the learning process so the policy is more stable during the evaluation. The settings of the parameters are shown in table 7.2.

Setup during evaluation	alpha	epsilon	Generated Action (GA)
1:	0.1	1	0
2:	0.3	1	0
3:	0.3	1	0
4:	0.3	1	0

Table 7.2: Parameter settings experiments after learning phase

7.2.3. Results parameter settings

The results in table 7.3 are based on the average of 2 emulations of 30 minutes after a learning phase of 3 hours in an emulation where the time is 4x accelerated.

Comparing the learning rate α using setup 1 and 2, we can see that in both experiment A and B, setup 2 performs better. So far the α setting of $\alpha = 0.3$ is preferred.

Comparing the balance between exploitation and exploration ϵ using setup 2 and 4, we can see that in both experiment A and B, setup 2 performs better. So far setup 2 the ϵ setting of $\epsilon = 0.9$ is preferred.

Comparing the balance between exploring new and existing actions GA using setup 3 and 4, we can see that in both experiment A and B, setup 4 performs better. So far the GA setting of $GA = 0.3$ is preferred.

Setup	Movements experiment A (S-curve)	Movements experiment B (U-curve)
1:	246	238
2:	265	261
3:	201	181
4:	229	213

Table 7.3: Results 30 min emulation experiments after learning phase of 180 minutes

Sensitivity analysis As shown above, the settings of the parameters influence the behaviour of the model and this behaviour can be argumentative declared. One can conclude that the model is sensitive to the parameters of the model.

Preferred settings for the experiments The preferred setup is alpha = 0.3, epsilon = 0.9 and GA = 0.3. This corresponds to setup 2, which performs the best overall.

7.2.4. Learning time

To determine the Learning time of the experiments, three emulations were conducted during 7 hours at a speed of 4x real time. All emulations were chosen to take place in scenario A: the S-curve, as the AGVs have to travel more in this scenario compared to scenario B. Therefore, the amount of new states is expected to be higher in scenario A. The results are shown in figure 7.1. The blue line is emulation 1, the orange line is emulation 2, the green line is emulation 3 and the red line is the average of all three emulations. The y axis shows the amount of new states that were found during the time interval of 30 minutes. The x axis shows the emulation time in hours.

In all three emulation, the amount of new states decreased to nearly zero after seven hours. As the graph shows, the amount of new states fluctuates up to 4.5 hours. After 4.5 hours, the fluctuation is negligible. Therefore the algorithm is trained with a learning time of 4.5 hours, which is 270 minutes.

The equipment emulator is set at a speed of 4x real time. The learning time of 4.5 hours thus corresponds to 18 hours of real time.

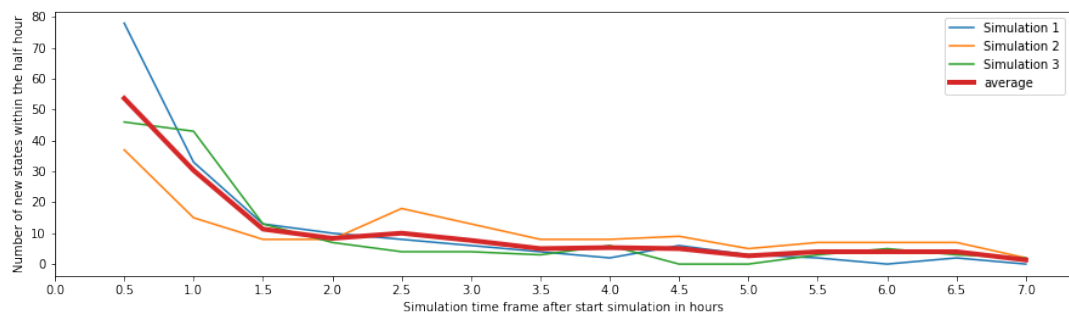


Figure 7.1: New states per hour simulating the S scenario

7.3. Verification

Action matrix During the experiments the action matrices are created based on the historical data. The data that is created by the model and shown in the emulation of the terminal is verified with the matrix that is created by the model. The route that is taken by the AGV and the time spent in each part of the map corresponds with the matrix created by the model.

An example is matrix 7.1 where the AGV drove without any disturbances on highway 1. The time changes when accelerating and decelerating near the start and end point. While driving on the highways there is no time spent in other areas than the highway. The areas of the highways in between the stacking areas are slightly larger and this can be seen in that matrix as well.

$$A = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1.303 & 0.79 & 0.53 & 0.227 & 0.523 & 0.297 & 0.243 & 0.297 & 0.25 & 0.3 & 0.483 & 0.233 & 0.54 & 0.787 & 1.367 & 0.0. \\ 0. & 2.017 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 2.336 & 0.0. & 0.0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0. \end{pmatrix} \quad (7.1)$$

Order The AGV needs to give a signal to the model when it arrives at its final destination, this way a new measurement can start when a new order is created.

Highway check The highway is determined by the algorithm and sent to the emulator. The emulator executes the order as given by the algorithm as we can see that the AGV drives along the route as ordered by the algorithm.

7.4. Validation

To see whether the algorithm can be applied in the real world, the choice was made to connect the algorithm to TEAMS. The algorithm is able to control the AGVs without issues. The TEAMS software is used at container terminals all over the world. A side note is that this has only been done at a small scale of the terminal. When the

idea described and tested in this thesis will be scaled up, the algorithm can be made more efficient regarding the use of data and be more by for example integrating it more into the software of TEAMS.

7.5. Results scenario A: the S-curve

In this section the performance of the VARLPPA (Vehicle Aware) is compared to the performance of the RLPPA (Single Vehicle) and the TEAMS algorithm. The following subsections display the results of the ratio of highway choice and both Key Performance Indicators. The learning time is 4.5 hours and the emulation time is 60 minutes at a speed of 4x real time.

7.5.1. Highway distribution

To see how the behaviour of the AGVs is influenced by the algorithms, this experiment has collected data regarding the choice of highway. Figure 7.2 and 7.3 show the behaviour of AGV 1 and AGV 2, respectively. Each of the graphs show the ratio of which highway is chosen by each AGV. The X axis shows the percentage of the choice of Highway one and Highway two. The Y axis show the results for each algorithm.

AGV1 The Single Vehicle algorithm learned a preference for highway one by 56.4 percent, the Vehicle Aware algorithm learned a preference for highway two by 60.1 percent and the TEAMS algorithm had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, the TEAMS algorithm distributed the movements to both highways equally.

AGV2 The Single Vehicle algorithm learned a preference for highway one by 70.7 percent, the Vehicle Aware algorithm learned a preference for highway two by 59.2 percent and the TEAMS algorithm had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, the TEAMS algorithm distributed the movements to both highways equally.

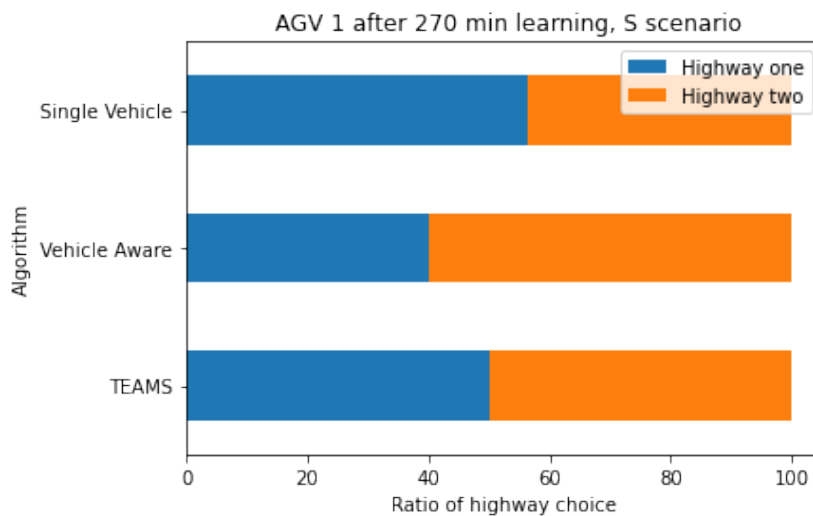


Figure 7.2: Ratio of highway choice AGV 1 S scenario

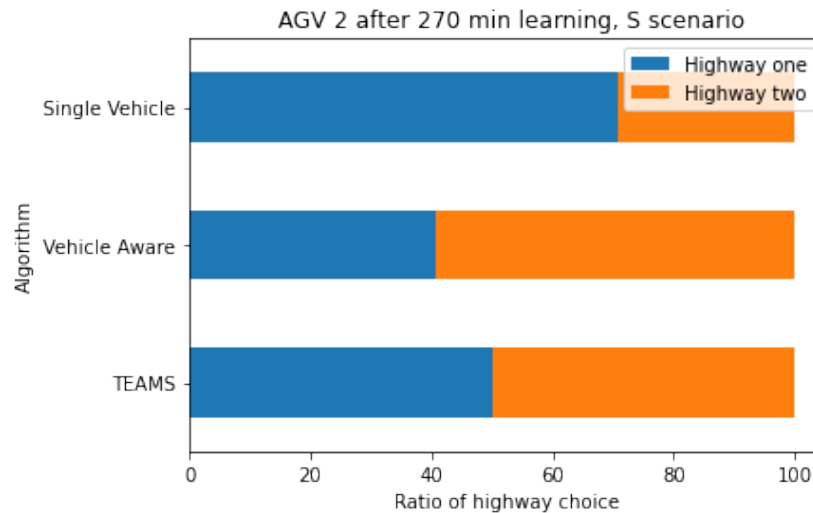


Figure 7.3: Ratio of highway choice AGV 2 S scenario

7.5.2. KPI 1 Productivity: Total number of completed orders

To measure the productivity of each algorithm, the total number of completed orders (movements per hour) was measured. Figure 7.4 shows the total movements per hour for both AGVs for each algorithm after 270 minutes of learning.

The TEAMS algorithm completed 476 movements per hour, the Single Vehicle algorithm completed 501 movements per hour and the Vehicle Aware algorithm completed 530 movements per hour.

Both the Single Vehicle algorithm and the Vehicle Aware algorithm performed better in terms of movements per hour compared to the TEAMS algorithm.

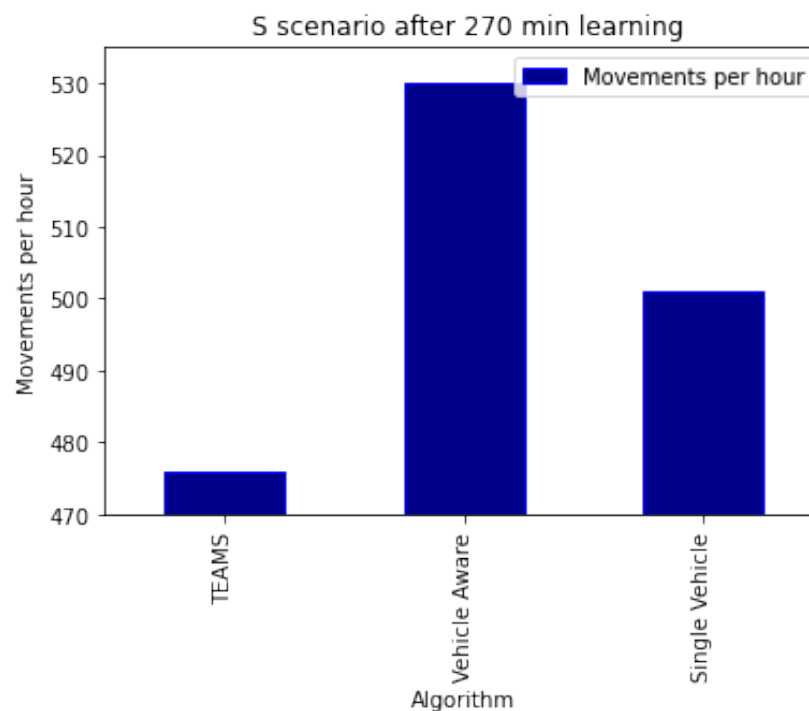


Figure 7.4: Total number of movements per hour S scenario

7.5.3. KPI 2. Container flow: The lowest number of completed orders by an AGV.

Figure 7.5 shows the lowest number of completed orders by each AGV.

When the Single Vehicle algorithm is executed, there is a small difference between the AVGs. The lowest amount of orders per hour was achieved by AGV 2, which completed 250 movements per hour. This is 49.9 % of the total amount of orders .

The Vehicle Aware algorithm also results in a small difference between the AGVs: AGV 1 completed 264 movements per hour, which corresponds to 49.8 % of the total amount of orders.

When using the TEAMS algorithm, both AGVs make an equal amount of movements per hour.

The target value for the KPI is to be within 15% of the value for the current TEAMS algorithm or higher. Seeing as this is the case, the KPI is achieved.

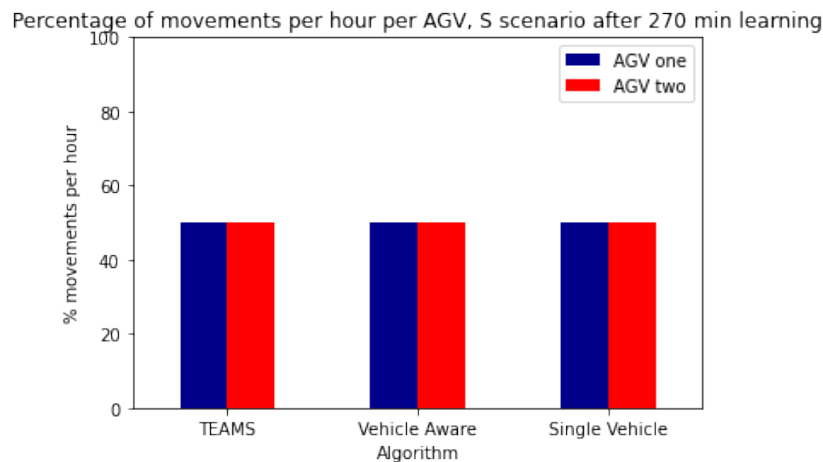


Figure 7.5: Percentage of movements per hour per AGV, S scenario after 270 min learning

7.6. Results scenario B: the U-curve

In this section the performance of the VARLPPA (Vehicle Aware) is compared to the performance of the RLPPA (Single Vehicle) and the TEAMS algorithm. The following subsections display the results of the ratio of highway choice and both Key Performance Indicators. The learning time is 4.5 hours and the emulation time is 60 minutes at a speed of 4x real time.

In this U-scenario, it is possible for the AGVs to choose the same route every time and never cross each other. Therefore, the researchers found a hypothesis for a best performing scenario, in which AGV 2 always takes the small U via highway 2, and AGV 1 always takes the large U via highway 1. To test this hypothesis, a new scenario is added to the emulation. This scenario is called 'Forced Highway algorithm', and manipulates AGV 1 into taking the large U and AGV 2 into taking the small U. These results are described below.

7.6.1. Highway distribution

To see how the behaviour of the AGVs is influenced by the algorithms, this experiment has collected data regarding the choice of highway. Figure 7.6 and 7.7 show the behaviour of AGV 1 and AGV 2, respectively. Each of the graphs show the ratio of which highway is chosen by each AGV. The X axis shows the percentage of the choice of Highway one and Highway two. The Y axis show the results for each algorithm.

AGV1 The Single vehicle learned a preference for highway two by 73.3 percent, the Vehicle Aware algorithm learned a preference for highway two by 67 percent, and the TEAMS algorithm had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, the TEAMS algorithm distributed the movements to both highways equally. The 'Forced Highway algorithm' is forced to travel via Highway 1.

AGV2 The Single vehicle learned a preference for highway two by 70 percent, the Vehicle Aware algorithm learned a preference for highway two by 93.3 percent, and the TEAMS algorithm had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, the TEAMS algorithm distributed the movements to both highways equally. The 'Forced Highway algorithm' is forced to travel via Highway 2.

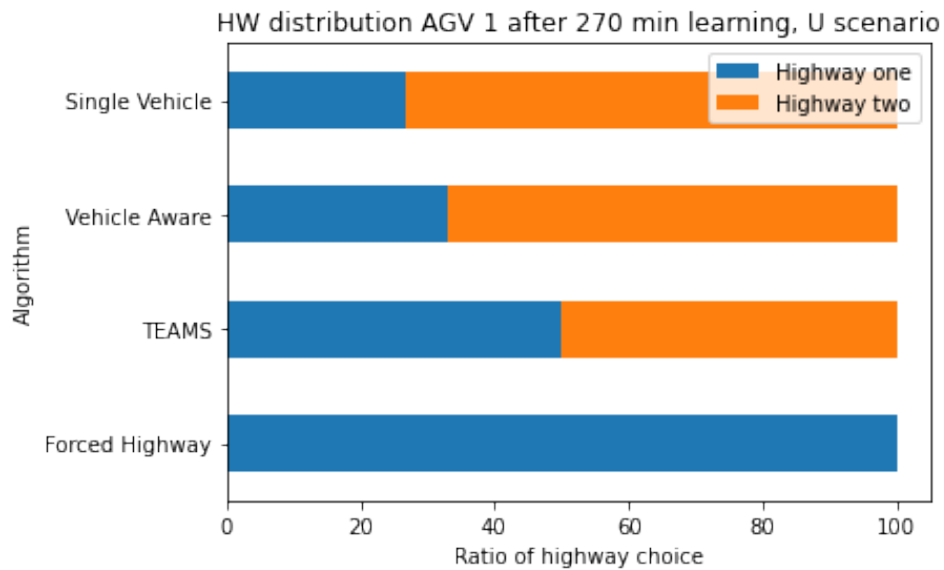


Figure 7.6: Ratio of highway choice AGV 1 U scenario

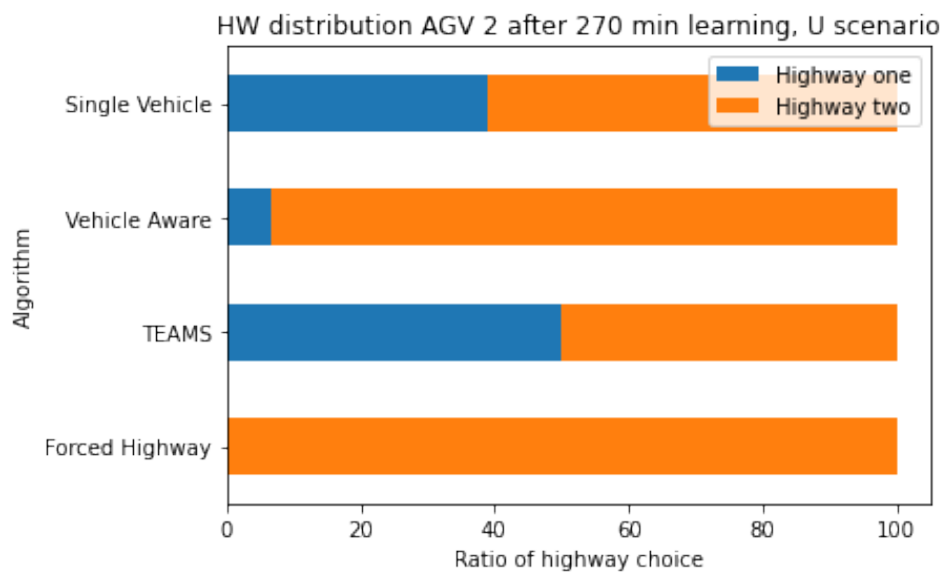


Figure 7.7: Ratio of highway choice AGV 2 U scenario

7.6.2. KPI 1 Productivity: Total number of completed orders

To measure the productivity of each algorithm, the total number of completed orders (movements per hour) was measured. Figure 7.4 shows the total movements per hour for both AGVs for each algorithms after 270 minutes of learning.

The forced highway algorithm completed 504 movements per hour, the TEAMS algorithm completed 473 movements per hour, the Single vehicle algorithm completed 494 movements per hour and the Vehicle Aware algorithm completed 518 movements per hour.

All algorithms are within 15% of the movements per hour, compared to the TEAMS algorithm. The Vehicle Aware algorithm achieved the highest amount of movements per hour.

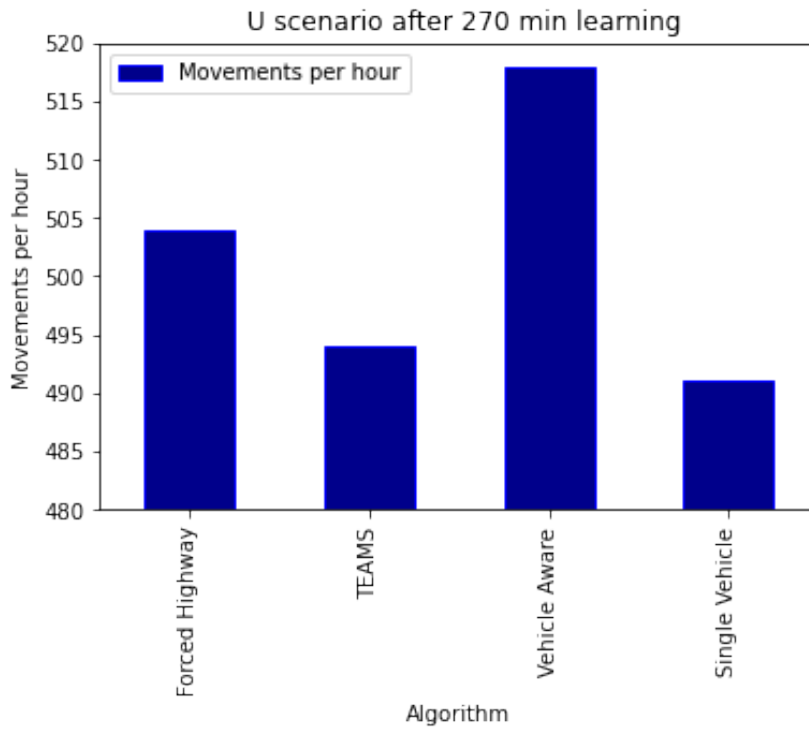


Figure 7.8: Total number of movements per hour U scenario

7.6.3. KPI 2. Container flow: The lowest number of completed orders by an AGV.

Figure 7.9 shows the lowest number of completed orders by each AGV. When the Single Vehicle algorithm is executed, the lowest amount of orders per hour was achieved by AGV 1, which completed 226 movements per hour. This is 45.7 % of the total amount of orders. In case of the Vehicle Aware algorithm, AGV 1 completed 234 movements per hour, which corresponds to 45.2% of the total amount of orders. When using the TEAMS algorithm, AGV 1 completed 221 movements, which corresponds to 44.7 % of the total. Lastly, in the Forced Highway scenario, AGV 1 completed 218 movements per hour, which is 43.3% of the total.

As described above, each scenario shows the same behaviour of AGV 1 achieving fewer movements per hour than AGV 2. This behaviour can be explained by the fact that AGV 1 has to drive the large U which is a longer route and can therefore complete fewer movements per hour with respect to AGV 2. The target value for the KPI is to be within 15% of the value for the current TEAMS algorithm or higher. Seeing as this is the case, the KPI is achieved.

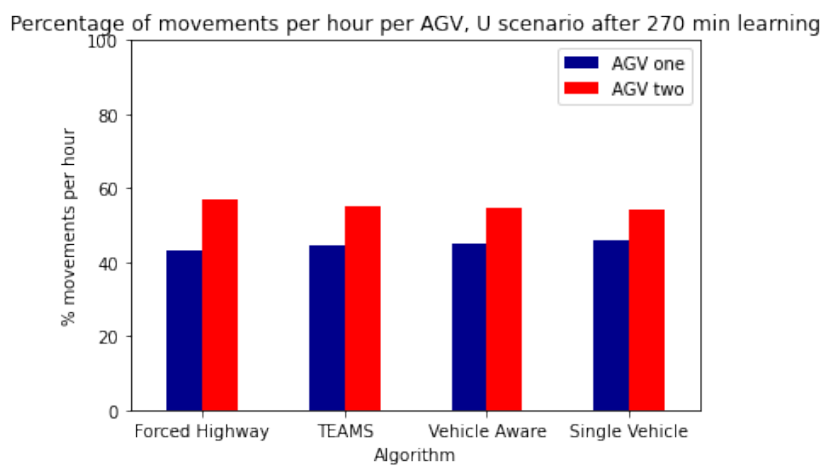


Figure 7.9: Percentage of movements per hour per AGV, U scenario after 270 min learning

7.7. Evaluation of the algorithm

This chapter has provided an analysis of the proposed algorithm.

7.7.1. Sub-question 5a: *How does the algorithm compare to the state of the art machine learning path planning, in terms of the KPIs?*

The most innovative element of the VARLPPA, compared to the design of the OPABRL, is the vehicle aware element: at every state of the VARLPPA, the state of the other vehicles is measured and taken into account, whereas the OPABRL is only used for the path planning of a single vehicle. In order to determine the added value of this element, the experiments discussed above have compared the VARLPPA with and without the vehicle aware element. These algorithms were called 'Vehicle Aware' and 'Single Vehicle', respectively.

KPI 1: Productivity The Vehicle Aware algorithm outperformed the Single Vehicle algorithm in both scenarios, in terms of KPI 1: movements per hour. In Scenario A, the performance was increased by 5.8% which is 29 movements per hour, and in scenario B, the performance was increased by 4.9%, which is 24 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm performed equally as well as the Single Vehicle algorithm in both scenarios, in terms of KPI 2: The lowest number of completed orders by an AGV. In both scenarios, the value of the Vehicle Aware showed a negligible difference between both AGVs.

7.7.2. Sub-question 5b: *How does the algorithm compare to the path planning algorithm of TBA, in terms of the KPIs?*

KPI 1: productivity The Vehicle Aware algorithm outperformed the TEAMS algorithm in both scenarios, in terms of KPI 1: movements per hour. In Scenario A, the performance was increased by 11.3% which is 54 movements per hour, and in scenario B, the performance was increased by 9.5%, which is 45 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm performed equally as well as the Single Vehicle algorithm in both scenarios, in terms of KPI 2: The lowest number of completed orders by an AGV. In both scenarios, the value of the Vehicle Aware algorithm was within 15% of the value for the current TEAMS algorithm.

7.7.3. Hypothesis evaluation: *How does the algorithm compare to the path planning of the Forced Highway algorithm, in terms of the KPIs?*

In the U scenario the hypothesis is tested whether the forced highway of the AGV in which they will never be able to run into each other outperforms the algorithm.

KPI 1: productivity The Vehicle Aware algorithm outperformed the Forced Highway algorithm, in terms of KPI 1: movements per hour. The performance was increased by 2.8% which is 14 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm outperformed the Forced Highway in terms of KPI 2: The lowest number of completed orders by an AGV. The lowest number of completed orders for Vehicle Aware was AGV 1 with 226 movements per hour, and the lowest number for the Forced Algorithm was 218 movements per hour.

7.7.4. Research question 5: *How does the algorithm perform in the experimental setup?*

The proposed algorithm outperforms both the state of the art algorithm and the TEAMS algorithm in both scenarios. Therefore, the scientific academic innovation and the innovation for TBA have been demonstrated.

8

Managerial insights and discussion

This chapter describes the Managerial insights and the discussion of the insights found during the thesis.

8.1. Managerial insights

This section describes the Managerial insights regarding the behaviour of the vehicles, how to cope with the terminal size regarding the evaluation areas and travel time, and finally learning from a real world terminal.

8.1.1. Machine vs human

As was demonstrated by the disproved hypothesis in subsection 7.7.3 the assumption of the engineer was outperformed by the Vehicle Aware algorithm. This demonstrates how machine learning can provide innovative solutions by learning a new policy.

8.1.2. Innovation by machine learning

As was shown by section 7.7.1, the proposed machine learning algorithm outperformed the current path planning algorithm in the tested scenarios. Therefore, it is recommended to research how machine learning can be applied on a broader scale to explore other scenarios.

8.2. Discussion

This section discusses limitations of the current research as well as suggestions for further research.

Vehicle behaviour The vehicle behaviour influences the results of the machine learning algorithm. For example, the AGVs in the TEAMS emulator are programmed to wait for other vehicles to pass when they are crossing their path. This is shown in figure 8.2. AGV 2 waits for AGV 1 to pass. In this case, the more efficient solution would be for AGV 1 to choose highway 1 instead of highway 2, so the AGVs never have to cross. However, since AGV 2 just waits for its turn, AGV 1 doesn't have to brake, which would indicate that highway 1 would be the better option, and thus never learns the optimal policy.

A solution would be to take the waiting time into account, so the action is influenced by the waiting time. This is a topic for further research.

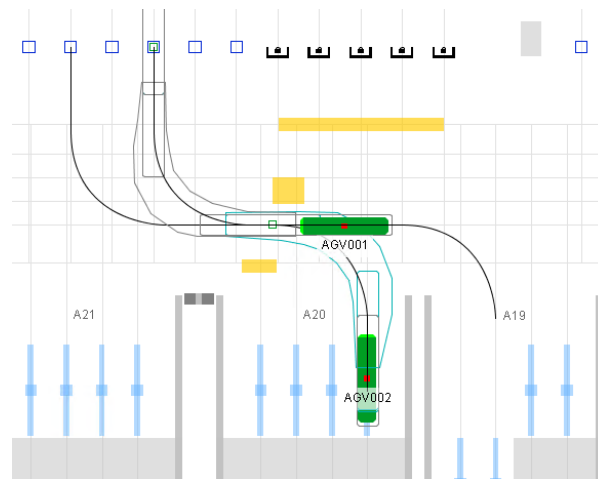


Figure 8.1: Influence of the AGV behaviour

Effect on other vehicles It is not possible for an AGV to learn the influence it has on the other AGVs. Each AGV learns the best route for itself based on the environment state. Although it has some information about the other AGVs' position and goal, as long as it is not disturbed by other AGVs, it will never learn to take another route. An example is given in image 8.2. Further research is necessary to improve the algorithm.

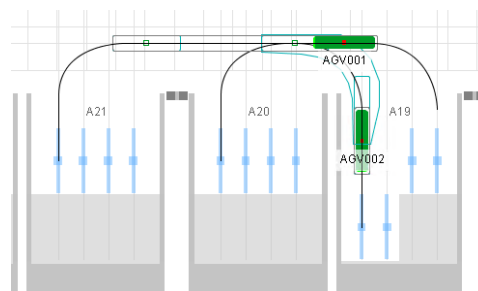


Figure 8.2: Influence of the AGV behaviour

The size of the terminal The size of the terminal during this research is limited due to the scope of this research. Therefore, further research can be done to indicate if the scale of the environment influences the performance of the algorithm.

Learning from a real world terminal The model which has been used during the tests is based on the data delivered by the manufacturers of the vehicles. Although this data is very close to the reality, there is still some variation. Further research can be done by connecting the algorithm to a real terminal without executing the highway choice.

Improve the heuristic function The heuristic function for this algorithm is the same for every node. Future research can be done by integrating more information in the heuristic function of the A* algorithm. This might have a positive impact on the results and the search time of the A* algorithm.

An example is when the AGV needs to travel along an area where containers are stacked, the suggested heuristic function will not avoid searching in that direction. Therefore, this should be integrated into a smarter heuristic function, to converge to the optimal path faster. The recommendation is to add this information to the state of the algorithm.

9

Conclusion

This thesis has provided insight into how machine learning can be beneficial to path planning in container terminals.

In automated container terminals, the Automated Guided Vehicles (AGVs) are guided by a path planning algorithm. In some cases, the layout leads to exceptional situations where case-specific solutions have to be found by the engineers to optimize the behaviour of the AGVs. Tuning or even building new software for each single situation requires a lot of manpower. Therefore there is potential in the integration of machine learning in software for the path planning systems in automated container terminals.

This thesis has proposed the machine learning algorithm Vehicle Aware Reinforcement Learning Path Planning Algorithm (VARLPPA). This algorithm uses Monte Carlo Control method. The algorithm consists out of nine phases: the initialization, checking availability of an AGV, translation of the measurement to the cost matrix, calculation of the action value, observation of the current state, action selection, route calculation, communication with the emulator, and when the set time is reached, storage of data.

The algorithm was tested in terms of scientific academic innovation and the value for the company TBA by multiple experiments.

9.1. Main research question

How can Machine Learning increase the efficiency of path planning for automated guided vehicles at a container terminal?

The proposed Machine learning algorithm has been proven by the experiments to be capable to come up with a policy which can outperform the current TEAMS algorithm, as well as the algorithm with the single vehicle states used in state of the art reinforcement learning path planning algorithms. By observing the position and destination of the AGV and the position, orientation and destination of the surrounding AGVs, the algorithm is capable to find a policy that is able to plan paths efficiently based on the A* algorithm.

The results show that the algorithm which takes the Vehicle Aware into account performs better regarding the KPIs then the algorithm that does not. The Machine Learning can increase the efficiency of path planning for automated guided vehicles at a container terminal if the states are well chosen.

The thesis has shown that it is possible to increase the efficiency of path planning in a container terminal by applying machine learning.

Bibliography

- [1] A. Terry Bahill and Clark Briggs. The systems engineering started in the middle process: a consensus of systems engineers and project managers. *Systems Engineering*, 4(2):156–167, 2001. ISSN 10981241. doi: 10.1002/sys.1013.
- [2] David Banyas and Daniel Kobran. Supervised, Unsupervised and Reinforcement Learning, 2020. URL <https://docs.paperspace.com/machine-learning/wiki/supervised-unsupervised-and-reinforcement-learning>.
- [3] Shweta Bhatt. Explaining Reinforcement Learning: Active vs Passive, 2018. URL <https://towardsdatascience.com/explaining-reinforcement-learning-active-vs-passive-a389f41e7195>.
- [4] Olivier Bosquet, Ulrike von Luxburg, and Ratsch Gunnar. *Advanced Lectures on Machine Learning*. 2004. ISBN 3540231226.
- [5] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Studies in Computational Intelligence*, 310, 2010. ISSN 1860949X. doi: 10.1007/978-3-642-14435-6_7.
- [6] Rina Dechter and Judea Pearl. Generalized Best-First Search Strategies and the Optimality of A. *Journal of the ACM (JACM)*, 32(3):505–536, 1985. ISSN 1557735X. doi: 10.1145/3828.3830.
- [7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0029599X. doi: 10.1007/BF01386390.
- [8] Happiness Ugochi Dike, Yimin Zhou, Kranthi Kumar Deveerasetty, and Qingtian Wu. Unsupervised Learning Based On Artificial Neural Network: A Review. *2018 IEEE International Conference on Cyborg and Bionic Systems, CBS 2018*, pages 322–327, 2019. doi: 10.1109/CBS.2018.8612259.
- [9] František Duchon, Andrej Babinec, Martin Kajan, Peter Beno, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified A star algorithm for a mobile robot. In *Procedia Engineering*, volume 96, pages 59–69, 2014. doi: 10.1016/j.proeng.2014.12.098.
- [10] Mark B. Duinkerken and Gabriel Lodewijks. Routing of AGVs on automated container terminals. *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2015*, pages 401–406, 2015. doi: 10.1109/CSCWD.2015.7230993.
- [11] Stuart Eiffert, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. Path Planning in Dynamic Environments using Generative RNNs and Monte Carlo Tree Search. Technical report, 2020.
- [12] Carsten Friedrich. Actions and states Qtable, 2018. URL <https://medium.com/@carsten.friedrich/part-3-tabular-q-learning-a-tic-tac-toe-player-that-gets-better-and-better-fa4da4b0892a>.
- [13] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, dec 2013. ISSN 09218890. doi: 10.1016/j.robot.2013.09.004.
- [14] Ewgenij Gawrilow, Max Klimm, Rolf H. Möhring, and Björn Stenzel. Conflict-free vehicle routing. *EURO Journal on Transportation and Logistics*, 1(1-2):87–111, jun 2012. ISSN 21924376. doi: 10.1007/s13676-012-0008-7.
- [15] Zoubin Ghahramani. Unsupervised Learning. *Gatsby Computational Neuroscience Unit University College London, UK*, 55(2):2659–2660, 2004.
- [16] Amir Hossein Gharehgozli, Debjit Roy, and René De Koster. Sea container terminals: New technologies and or models. *Maritime Economics and Logistics*, 18(2):103–140, 2016. ISSN 1479294X. doi: 10.1057/mel.2015.3.

- [17] Yejun Hu, Liangcai Dong, and Lei Xu. Multi-AGV dispatching and routing problem based on a three-stage decomposition method. *Mathematical Biosciences and Engineering*, 17(5), 2020. ISSN 15510018. doi: 10.3934/mbe.2020279.
- [18] Ariel Keselman, Sergey Ten, Adham Ghazali, and Majed Jubeh. Reinforcement Learning with A* and a Deep Heuristic. Technical report, 2018. URL <http://arxiv.org/abs/1811.07745>.
- [19] Reinhard Klette and Azriel Rosenfeld. Dijkstra Algorithms. *Digital Geometry*, pages 117–157, 2004. doi: 10.1016/b978-155860861-0/50006-7.
- [20] Chong Li. Model-Based Reinforcement Learning. In *Reinforcement Learning for Cyber-Physical Systems*, pages 69–92. 2019. doi: 10.1201/9781351006620-4. URL <https://goo.gl/itykP8>.
- [21] Junjun Li, Bawei Xu, Yongsheng Yang, and • Huafeng Wu. Quantum ant colony optimization algorithm for AGVs path planning based on Bloch coordinates of pheromones. *Natural Computing*, 19, 2018. doi: 10.1007/s11047-018-9711-0. URL <https://doi.org/10.1007/s11047-018-9711-0>.
- [22] Jae Kook Lim, Joon Mook Lim, Kazuho Yoshimoto, Kap Hwan Kim, and Teruo Takahashi. A construction algorithm for designing guide paths of automated guided vehicle systems. *International Journal of Production Research*, 40(15 SPEC.):3981–3994, 2002. ISSN 00207543. doi: 10.1080/00207540210137558.
- [23] Xiao Huan Liu, De Gan Zhang, Hao Ran Yan, Yu Ya Cui, and Lu Chen. A New Algorithm of the Best Path Selection Based on Machine Learning. *IEEE Access*, 7:126913–126928, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2939423.
- [24] Andrea Lonza. *Reinforcement Learning Algorithms with Python*, volume 53. 2019. ISBN 9788578110796.
- [25] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep Reinforcement Learning: An Overview. *Lecture Notes in Networks and Systems*, 16:426–440, 2018. ISSN 23673389. doi: 10.1007/978-3-319-56991-8_32.
- [26] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 2776–2786, 2017.
- [27] O Obulesu, M. Mahendra, and M. Thrilokreddy. Machine Learning Techniques and Tools: A Survey. In *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, number Icirca, pages 605–611. IEEE, 2018. ISBN 9781538624562. doi: 10.1109/ICIRCA.2018.8597302.
- [28] Changhyeon Park and Seok Cheol Kee. Online local path planning on the campus environment for autonomous driving considering road constraints and multiple obstacles. *Applied Sciences (Switzerland)*, 11(9), 2021. ISSN 20763417. doi: 10.3390/app11093909.
- [29] Soumya Ray and Prasad Tadepalli. Model-Based Reinforcement Learning. pages 1–4, 2009.
- [30] Arthur L Samuel. Some Studies in Machine Learning. *IBM Journal of Research and Development*, 3(3): 210–229, 1959. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5392560>.
- [31] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*, volume 9781107057. 2013. ISBN 9781107298019. doi: 10.1017/CBO9781107298019.
- [32] Ryuichi Shibasaki. *Data [2] container shipping demand for the present and future*. Elsevier Inc., 2021. ISBN 9780128140604. doi: 10.1016/b978-0-12-814060-4.00009-5. URL <https://doi.org/10.1016/B978-0-12-814060-4.00009-5>.
- [33] Valentyn N. Sichkar. Reinforcement learning algorithms in global path planning for mobile robot. *2019 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2019*, pages 10–14, 2019. doi: 10.1109/ICIEAM.2019.8742915.
- [34] Richard S.Sutton and Andrew G.Barto. *Reinforcement Learning: An Introduction*, volume 4. 2020. ISBN 9780262039246. URL <http://marefateadyan.nashriyat.ir/node/150>.

- [35] UNCTAD. *UNCTAD Handbook of Statistics 2020 - Maritime transport indicators*. 2020. ISBN 9789211129977. URL https://unctad.org/system/files/official-document/tdstat45_en.pdf.
- [36] Bert Van Wee and David Banister. How to Write a Literature Review Paper? *Transport Reviews*, 36(2): 278–288, 2016. ISSN 14645327. doi: 10.1080/01441647.2015.1065456. URL <http://dx.doi.org/10.1080/01441647.2015.1065456>.
- [37] Lilian Weng. A (Long) Peek into Reinforcement Learning, 2020. URL <https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html#deep-q-network>.
- [38] Jianbin Xin, Rudy R. Negenborn, Francesco Corman, and Gabriël Lodewijks. Control of interacting machines in automated container terminals using a sequential planning approach for collision avoidance. *Transportation Research Part C: Emerging Technologies*, 60:377–396, 2015. ISSN 0968090X. doi: 10.1016/j.trc.2015.09.002.
- [39] Jianbin Xin, Liuqian Wei, Dongshu Wang, and Hua Xuan. Receding horizon path planning of automated guided vehicles using a time-space network model. *Optim Control Appl Meth*, 41:1889–1903, 2020. doi: 10.1002/oca.2654.
- [40] Tianqi Zha, Lei Xie, and Jiliang Chang. Wind farm water area path planning algorithm based on A* and reinforcement learning. *ICTIS 2019 - 5th International Conference on Transportation Information and Safety*, (2017):1314–1318, 2019. doi: 10.1109/ICTIS.2019.8883718.
- [41] De Gan Zhang and Wen Bin Li. Novel ID-based anti-collision approach for RFID. *Enterprise Information Systems*, 10(7), 2016. ISSN 17517583. doi: 10.1080/17517575.2014.986221.
- [42] Hong Mei Zhang and Ming Long Li. Rapid path planning algorithm for mobile robot in dynamic environment. *Advances in Mechanical Engineering*, 9(12):2017, 2017. ISSN 16878140. doi: 10.1177/1687814017747400. URL <https://us.sagepub.com/en-us/nam/>.
- [43] Yi Zhang, Yu Qian, Yichen Yao, Haoyuan Hu, and Yinghui Xu. Learning to cooperate: Application of deep reinforcement learning for online AGV path finding. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2020-May(3):2077–2079, 2020. ISSN 15582914.
- [44] Zhanying Zhang and Ziping Zhao. A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm. *International Journal of Smart Home*, 8(3):75–86, 2014. ISSN 1975-4094. doi: 10.14257/ijsh.2014.8.3.07. URL <http://dx.doi.org/10.14257/ijsh.2014.8.3.07>.
- [45] Meisu Zhong, Yongsheng Yang, Yasser Dessouky, and Octavian Postolache. Multi-AGV scheduling for conflict-free path planning in automated container terminals. *Computers and Industrial Engineering*, 142(June 2019):106371, 2020. ISSN 03608352. doi: 10.1016/j.cie.2020.106371. URL <https://doi.org/10.1016/j.cie.2020.106371>.
- [46] Meisu Zhong, Yongsheng Yang, Shu Sun, Yamin Zhou, Octavian Postolache, and Ying En Ge. Priority-based speed control strategy for automated guided vehicle path planning in automated container terminals. *Transactions of the Institute of Measurement and Control*, 42(16):3079–3090, 2020. ISSN 01423312. doi: 10.1177/0142331220940110.
- [47] Luwei Zhou, Pei Yang, Chunlin Chen, and Yang Gao. Multiagent reinforcement learning with sparse interactions by negotiation and knowledge transfer. *IEEE Transactions on Cybernetics*, 47(5):1238–1250, 2017. ISSN 21682267. doi: 10.1109/TCYB.2016.2543238.

A

Flowchart of the Learning-based path
planning for container terminals algorithm

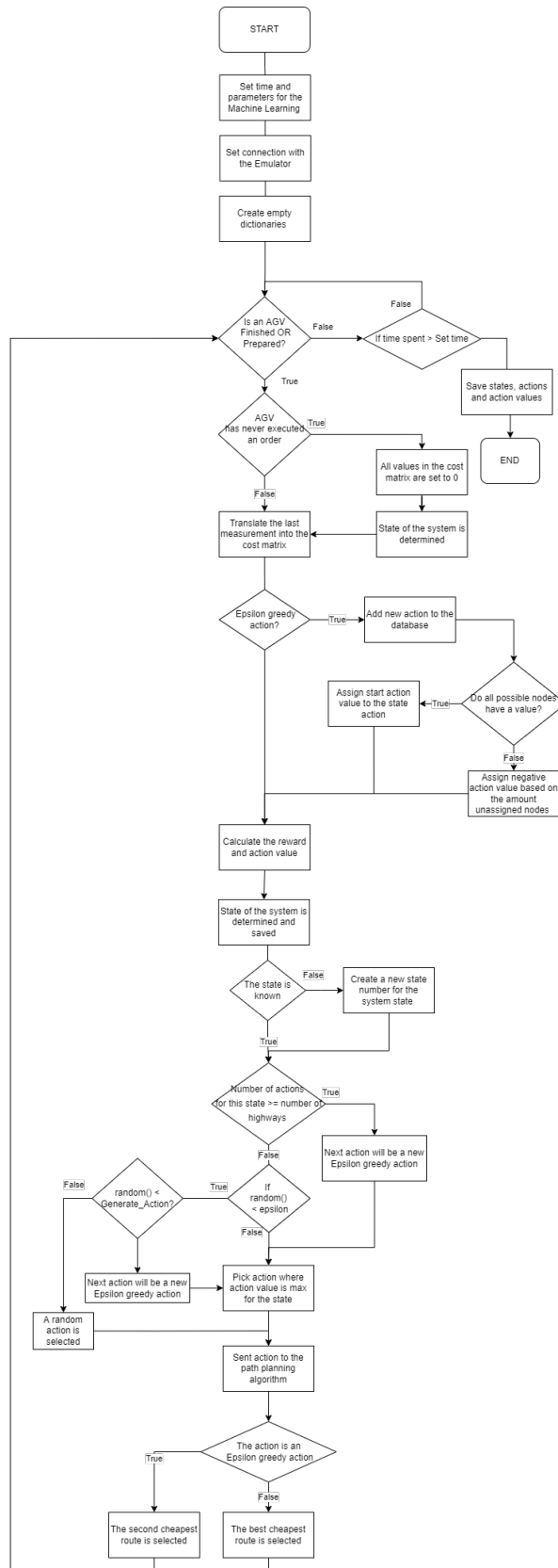


Figure A.1: Flowchart of the proposed algorithm

B

Scientific paper

Learning-based path planning for automatic guided vehicles in container terminals

P. Wijnands, Bokkers, M.B., MSc, A.W. ter Mors Ph. D, Dr. F. Schulte, Prof. Dr. R.R. Negenborn

Delft University of Technology (TU Delft), Mekelweg 2. 2628 CD Delft, the Netherlands
servicepunt-3me@tudelft.nl

Abstract. This thesis has provided insight into how machine learning can be beneficial to path planning in container terminals.

Path planning algorithms can be used in environments with automated vehicles. A well known algorithm is the A* path planning algorithm, which is the fastest optimal path planning algorithm under satisfied conditions. However, the behaviour of a container terminal is unknown beforehand, costs can change over iterations. Therefore, Liu et al. [Liu et al., 2019] and Keselman et al. [Keselman et al., 2018] show the advantage of combining A* with Machine Learning. This way, the exploring part of the ML algorithm is combined with the fast and more precise properties of the A* PP algorithm.

This thesis has proposed the machine learning algorithm Vehicle Aware Reinforcement Learning Path Planning Algorithm VARLPPA. This algorithm uses Monte Carlo Control method. This is a model free approach, which has been shown in both experiments to find more efficient solutions in exceptional situations.

Keywords: Container terminal · Reinforcement Learning · Path Planning

1 Introduction

As the world economy keeps growing, the container cargo shipping demand will keep increasing, particularly in developing countries Shibasaki [2021]. With the growth of the demand for Twenty-foot Equivalent Unit (TEU) transport, the demand for smarter and faster container terminals increases with it.

Their layouts differ among one other because of different surroundings, equipment, degree of automation etc. This causes different challenges for each terminal. In automated container terminals, the Automated Guided Vehicles (AGVs) are guided by a path planning algorithm. In some cases, the layout of the terminal leads to exceptional situations where case-specific solutions have to be found by the engineers to optimize the behaviour of the AGVs. Tuning or even building new software for each single situation requires a lot of manpower. A solution for the path planning software to adapt to situations in (new) terminals can be Machine Learning. Machine Learning is software which uses data to draw conclusions or to choose an action. An example would be to use information about the terminal to choose the most efficient path for the vehicles. In doing so, Machine Learning software can cope with

any new situation. Conclusions of the Machine learning algorithm can be integrated into the existing software to improve the in the industry used software in specific cases. Therefore there is potential in the integration of machine learning in software for the path planning systems in automated container terminals.

1.1 Problem definition

In order to contribute to the improvement and scientific academic innovation of path planning software of container terminals, this paper applies Machine Learning to path planning in container terminals, by designing an innovative Machine Learning path planning algorithm.

This paper is organized as follows: first, the state of the art path planning is described by reviewing the literature in section 2. Second, an innovative algorithm is proposed for path planning in container terminals, using a Markov Decision Process in section 3 and describing the algorithm in section 4. Next, section 5 describes the Key Performance Indicators and multiple experiments. Section 6 and 7 describe the Sensitivity analysis and the verification and validation, respectively. The results of both experiments are described in section 8 and 9. Finally, the algorithm is evaluated in section 10 and the conclusion is drawn in section 11.

2 Related work

This section discusses the literature found on path planning algorithms in combination with reinforcement learning.

2.1 What are Path Planning algorithms?

Path planning algorithms are used in a variety of situations and environments where routes need to be determined. The algorithm is used to find the shortest path based on the given data. Examples are logistic environments, such as finding the best route for a mobile robot [Duchon et al., 2014], guiding autonomous vehicles across a campus environment Park and Kee [2021], navigation of ships in wind farm areas [Zha et al., 2019], and, as is the case in the current paper, Automated Guided Vehicles in container terminals. The goal of the algorithm depends on the needs of the user. Examples are to find the lowest cost, shortest path or most fuel efficient-path. Thus, for each situation another path planning algorithm fits best. The fundamental algorithm for Path planning is the Dijkstra algorithm. A variant of it, the A* algorithm is widely used (Zhang and Zhao [2014]).

Unknown scenarios Dijkstra and A* are based on a static environment and will not deliver the shortest path when the costs are unknown or not precise. However, the algorithms can be used to plan paths in these environments, albeit not the best way. Examples are planning the route in local areas when collisions are predicted [Zhang and Li, 2017].

2.2 Reinforcement learning applied to Path Planning

There are several methods to apply Reinforcement Learning to path planning.

First of all, [Eiffert et al., 2020] created an algorithm that can interact with the world around it, e.g. pedestrians, using a MCTS algorithm. Second, Zhou et al. [2017] have created a multi vehicle reinforcement learning algorithm where agents first learn their policy and reward models while interacting individually with the environment. Third, Sichkar [2019] have created a path planning algorithm used in a static environment using Q-learning and SARSA algorithm. Third, Liu et al. [2019] made a path planning algorithm which combines the A* algorithm with reinforcement learning. Last, [Keselman et al., 2018] suggests a method in which the heuristic is calculated using a general model-based reinforcement learning algorithm.

Under satisfied conditions, A* is the fastest optimal path planning algorithm [Dechter and Pearl, 1985]. Liu et al. [2019] and Keselman et al. [2018] show the advantage of combining A* with Machine Learning. This way, the exploring part of the Machine Learning algorithm is combined with the faster and more precise properties of graph search PP algorithms. However, there is a gap in the literature regarding applying this to container terminals. There is potential in combining the A* algorithm with a reinforcement learning algorithm to create a stronger path planning algorithm in container terminals. There is possible gain in using machine learning for the heuristic or the cost part of the A* algorithm. Because the environment is unknown, costs change over iterations.

Another gap is to combine the A* algorithm with machine learning and vehicle awareness planning. Currently there is no literature on this subject. Consequently, the current research investigates the possibilities of combining the A* algorithm with machine learning and vehicle aware planning. This concept is worth investigating, as this has never been done before for a container terminal.

3 Methodology

The proposed algorithm is designed according to the Markov Decision Process.

This section describes the MDP for this case.

3.1 State elements

It is important for a reinforcement learning system to choose the state variables carefully. This is because every added state variable that does not add value to the agent will slow down the policy learning process. The state of the system is defined by the following real time updated variables for each AGV:

- **Position x** of the AGV requesting its state
- **Position y** of the AGV requesting its state
- **Destination x** of the AGV requesting its state
- **Destination y** of the AGV requesting its state

For the environment of the AGV:

- **Position x** of AGV_n
- **Position y** of AGV_n
- **Destination x** of AGV_n
- **Destination y** of AGV_n
- **Orientation** of AGV_n

Orientation The orientation of the other AGVs provides extra information on the state of the system. For example, when an AGV is taking a corner, it is driving slower compared to driving in a straight line on a highway. This influences the optimal action for the path planning. The orientation will provide an indication of which AGVs are taking a corner, among other things.

The information about the orientation is not described as the full 360 degrees, but rather as 4 different orientations, as shown in table 1. The reason for this is that precision will not benefit the system but rather increase the complexity. The orientation of the AGV which requests its state is not included, as it will always start at the buffer or stacking area. In this position, the AGV is always parked in a certain orientation, which will therefore not add any value to the policy learning.

Degrees AGV	Code
315 till 45	0
45 till 135	1
135 till 225	2
225 till 315	3

Table 1. Code to Degrees

3.2 The state

Table 2 shows an overview of the state of the system. The maximum value of n is the total number of AGV's -1. The count is sorted based on the position of the nearest AGV based on Euclidean distance. When there are AGVs that are exactly at the same distance from the current AGV, the name of the state is sorted secondly on the closest end positions with respect to the end position of the current AGV.

Table 2. State of the system

$(Sx, Sy);$	<i>/* x,y position of AGV requesting route */</i>
$(Ex, Ey);$	<i>/* x,y target of AGV requesting route */</i>
$(Cx_n, Cy_n);$	<i>/* x,y position of AGV_n */</i>
$(Ex_n, Ey_n);$	<i>/* x,y target of AGV_n */</i>
$O_n;$	<i>/* Orientation of AGV_n */</i>
<i>Combining the elements to the state:</i>	
$((Sx, Sy), (Ex, Ey)(Cx_1, Cy_1)(Ex_1, Ey_1)O_1 \dots (Cx_n, Cy_n)(Ex_n, Ey_n)O_n);$ <i>/* State */</i>	
$((2, 1), (3, 4), (1, 1), (4, 3), 0);$ <i>/* Example state of a two AGV environment */</i>	

3.3 Actions

The action of the reinforcement learning algorithm provides the path planning algorithm with data to calculate the route. New actions are based on the time spent by the AGV in each part of the taken route.

Algorithm 1 shows how the actions are chosen.

The amount of possible actions keeps growing during the learning phase, as it is an epsilon greedy algorithm. The algorithm will choose an action if it meets the ϵ -greedy and GA conditions. The value of these parameters are set before the learning phase.

The parameter Measured Action (MA) determines whether the route that is taken is added as a new measured action. If this is True, this is the case.

The algorithm can either pick an action with the highest action value or pick a Greedy Action. When a Greedy Action is chosen, there are two options: a new action is explored, or a random known action is exploited.

When the new action is explored, the route is based on the cost matrix with highest state action value e.g. Q value for this state. The Q value is defined as the action with the highest calculated value for this particular state.

The parameters are assigned a random number between 0 and 1: ϵ is the chance of picking a Greedy Action, GA is the chance of exploring a new action.


```

 $\epsilon \leftarrow rand(0,1);$  /* Setting for random or generated actions
between 0 and 1 */
 $GA \leftarrow rand(0,1);$  /* Setting for generated actions between 0 and
1 */
 $MA \leftarrow False;$  /* If 1, time spent in each part of the terminal
is measured and added as an action for the state */
if  $NA \leq NH$  then
|  $MA \leftarrow 1;$  /* NA is the Number of actions for the state. NH
| is the number of highways in the environment */
| if  $NA = 0$  then
| |  $A_{state} = 0;$  /* The zero matrix is assigned so a path based
| | on the heuristic is chosen */
| end
| end
| if  $(n < \epsilon)$  then
| | if  $(n < GA)$  then
| | |  $A \leftarrow \max Q(S);$  /* Cost matrix with highest Q value for
| | | this state */
| | |  $MA \leftarrow 1$ 
| | | end
| | | else
| | | |  $A \leftarrow A_r;$  /* A random cost matrix linked to this state */
| | | | end
| | end
| end
| else
| |  $A \leftarrow \max Q(S);$  /* Cost matrix with highest Q value for this
| | state */
| end

```

Algorithm 1: Action

3.4 Reward

The more accurate the path planning algorithm is provided with data, the better it can calculate the route for the AGVs. Therefore the closer the algorithm reaches an accurate measurement, the higher the reward will be.

The reward function converges to unlimited if $dt = 0$, as 1 is divided by dt . For this reason if dt is equal to 0, the value of dt is set near the lowest possible value in a float.

```

 $T \leftarrow \sum |(t_{mm}(m, n) - t_{am}(m, n))|;$           /* Time difference between
estimation and measurement */
 $t_{mm}, t_{am};$  /* time spent at area, mm is measured matrix and ma
action matrix */
 $m = 1, 2..M;$           /* with M the number of x grid lines */
 $n = 1, 2..N;$           /* with N the number of y grid lines */
if ( $T = 0$ ) then
|  $T = 6 * 10^{-4};$  /* Near the lowest possible value in a float */
end
 $R \leftarrow 1/|T|;$           /* The closer to the solution, the higher the
reward */

```

Algorithm 2: Reward R

3.5 The value of the action: the Q value

The state action value (Q-value) is based on the Learning rate, the reward and the current state action value, as shown in algorithm 3. The initialized value of the state action values are chosen positive and above zero so that the algorithm will exploit the actions more than when zero is picked. Because the actions the AGV takes in the future do not influence the accuracy of this action, and therefore the value of this action, they are not taken into account.

```

if ( $s \notin S$ ) then
|  $Q(s, a) \leftarrow p;$           /* If the state action pair is never visited
before, it is initialized with p ( $p > 0$ ). This makes sure
that if new actions are chosen by the epsilon greedy part
of the algorithm, they have more chance that they will
converge to their real value with respect to Q
initialized = 0 */
| else
| |  $Q(s+1, a+1) \leftarrow (1 - \alpha) * Q(s, a) + (\alpha * R(s, a))$ 
| | ; /* The closer to the solution, the higher the reward
| | */
| end
end

```

Algorithm 3: Q value

Where:

α is the Learning Rate

$R(s, a)$ the reward

$Q(s, a)$ the expected value of the quality of a certain action in a given state.

Learning rate α As the environment is explored and exploited, information about the environment can change over time, and so do the rewards of the actions. The higher the Learning rate α , the more aggressive the algorithm will react to changes

in the value of the last reward. Because of the unknown environment we want the Q value to adapt relatively quickly, without the system becoming unstable.

4 A Reinforcement Learning Based Path Planning Algorithm

This section describes the design and implementation of the Reinforcement Learning Based A* Algorithm. First the overview and the choices in the top level are described. Then for each part of the algorithm a detailed flow chart is given.

4.1 Reinforcement Learning aspects

In the literature review, three aspects were found to categorize reinforcement learning algorithms. Table 3 summarizes these aspects and their corresponding options. The following section argues which of the options is chosen for each aspect.

Aspect		
A. Learning method	Model Free	Model based
B. Policy	On-policy	Off-policy
C. Epsilon-Greedy	Yes	No

Table 3. Aspects used in reinforcement learning algorithms

Learning method: Model free The model free approach is the best option for this case, as it unknown how the AGV will respond in the states that will be created in the environment. The biggest advantage of the model free approach is the ability to learn in different kinds of terminals, which eliminates the need for a model to be built for each terminal. The challenge will be the converge speed, as we do not know all the behaviours of the environment beforehand.

Policy: Off-policy During the training phase, there is a choice between an on-policy or off-policy method. As S.Sutton and G.Barto [2020] describe it, "the distinguishing feature of on-policy methods is that they estimate the value of a policy while using it for control. In off-policy methods these two functions are separated. The policy used to generate behavior, called the behavior policy, may in fact be unrelated to the policy that is evaluated and improved, called the target policy". For example, during the training phase, the behaviour policy might be to explore multiple solutions, while the target policy is to find the optimal solution. Although off-policy algorithms are less stable than on-policy, off-policy methods make use of the new data in every iteration, as the old policy is not used for the next iteration. In case of this experiment, the Q-values of the future are important as it is expected that the policy will develop during the learning phase. This is based on the new expected states that will differ from the policy and the reward during the interaction with the A* algorithm.

Epsilon-Greedy: Use Epsilon-Greedy policy When the choice of action is always based on the highest state action value, new actions will not be explored. This means that in large environments there is a risk that the optimal policy will never be discovered. Therefore an element ϵ can be added to balance between exploration and exploitation. Algorithms that include the epsilon greedy element pick a (pre-set) value of the ϵ , which influences the amount of actions chosen based on randomness. This way the agent can discover a new policy or can confirm that the action is indeed one of low value.

4.2 Learning-based path planning for AGVs in container terminals

The most suitable methodology for this case is Epsilon-Greedy Monte Carlo Control. This method contains all preferred aspects discussed above. Other methods are Temporal Difference and Monte Carlo Learning, which are both ruled out. The algorithm will be combined with A* path planning, which means that Temporal Difference Learning is not an option, because the A* needs stable input. Neither is Monte Carlo Learning, because the environment of the container terminal is too large. Therefore, the most suitable methodology would be the Monte Carlo Control. The disadvantage of the long learning trajectories will be taken into account when designing the algorithm.

Monte Carlo Control This algorithm is used to approximate optimal policies. The values learned are used for the decision of the action [S.Sutton and G.Barto, 2020]. Figure 1 shows the basics of Monte Carlo Control, where: π_n is the policy, E is the episode, q_π is the action value of the complete episode, I is a complete policy improvement. By following this method, the optimal policy π_* can be found.

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$$

Fig. 1. Loop of Monte Carlo Control [S.Sutton and G.Barto, 2020]

Epsilon greedy The Epsilon-Greedy policy will allow the algorithm to (re)discover the effects of the actions in the learning phase. The Epsilon-Greedy will not be used during the execution of the algorithm after the learning phase.

4.3 Overview of the algorithm

The algorithm consists out of nine phases: the initialization, checking availability of an AGV, translation of the measurement to the cost matrix, calculation of the action value, observation of the current state, action selection, route calculation, communication with the emulator, and when the set time is reached, storage of data. Figure 2 shows how these parts of the algorithm are connected. Each of these phases consists out of multiple steps. The complete flowchart with all of the detailed steps is included in appendix A.1

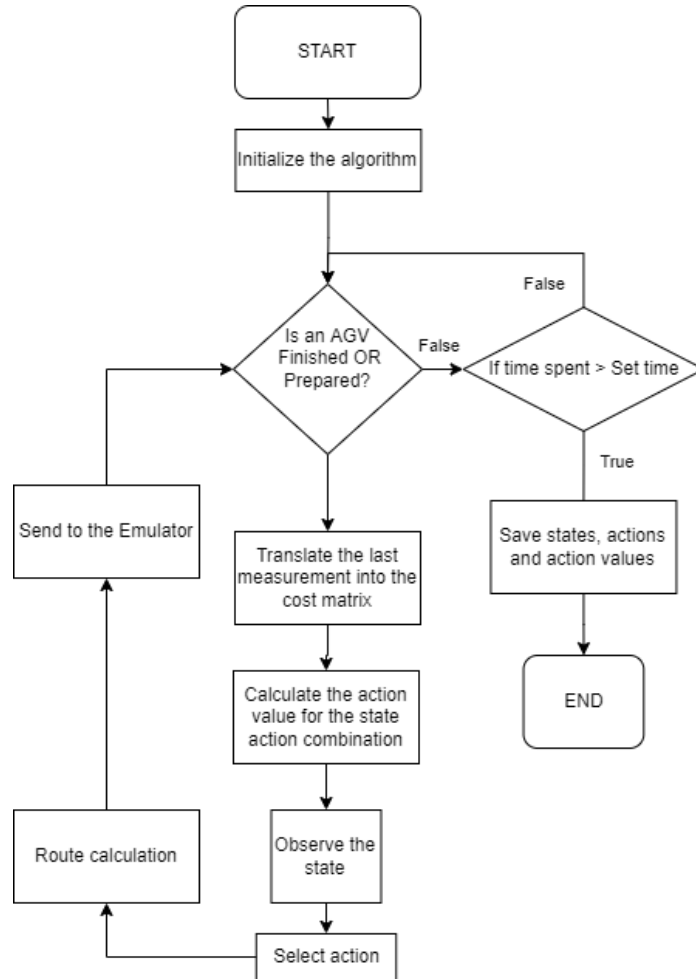


Fig. 2. Top level algorithm

4.4 Comparison with the state of the art

This section compares the algorithm designed by Liu et al. (OPABRL), to the proposed algorithm in this paper (VARLPPA).

The most innovative element of the VARLPPA, compared to the OPABRL, is the vehicle aware element: at every state of the VARLPPA, the state of the other vehicles is measured and taken into account, whereas the OPABRL is only used for the path planning of a single vehicle. In order to determine the added value of this element, the experiments will compare the VARLPPA to a version of the VARLPPA without the vehicle aware element, which is called RLPPA.

5 Case study: Learning based path planning in a container terminal

A case study is conducted to test the value of the VARLPPA. This study uses a container terminal environment emulator from an industrial company. Their software is used in container terminals all over the world.

5.1 Challenging situations in container terminals

The currently used algorithm in the industry (TEAMS) has assigned a direction to each highway, in order to structure the container terminal. The directions are based on the test results of the emulation of the terminal. The one-way direction of the highway is chosen to keep the terminal more stable, while also avoiding congestion and deadlocks. The advantage of the philosophy of one-way driving on the highway is shown in figure 3. The AGVs avoid each other and AGV002 takes the shortest route.

However, in some situations this policy is not beneficial, and instead, provides challenges. Such situations are described in the next paragraph.

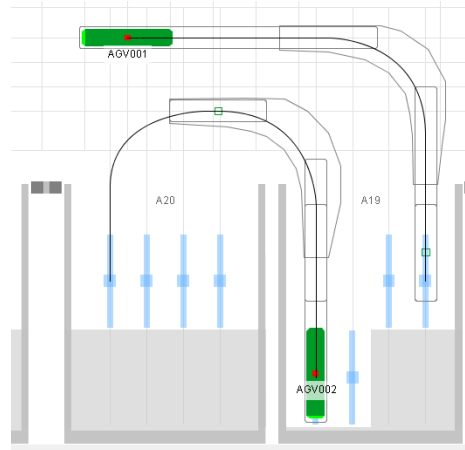


Fig. 3. Advantage of the highway directions

Disadvantage of the set highway direction Having a set highway direction is not beneficial to the paths taken by the AGVs. In figure 4 there is a chance that the AGVs will get in each others way and both have to take a longer route due to the direction restriction. In figure 5 AGVs are crossing each other, which results in breaking.

Another example is shown in figure 6 where AGV002 has to wait for AGV001. Lastly in figure 7 both AGVs are in a position where one has to wait for the other.

In all four examples, a different policy could increase the efficiency regarding the order completion time. Therefore it is desired to design the experiments in such a

way that it is likely for these situations to occur. This way, the proposed algorithm is tested as to how it copes with these challenges.

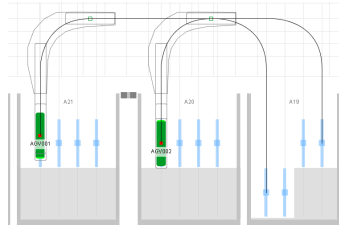


Fig. 4. Long route example 1

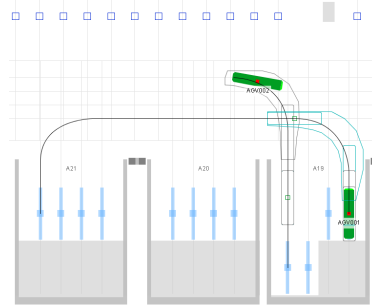


Fig. 5. AGVs crossing example 2

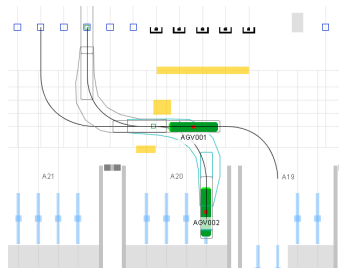


Fig. 6. AGVs crossing example 3

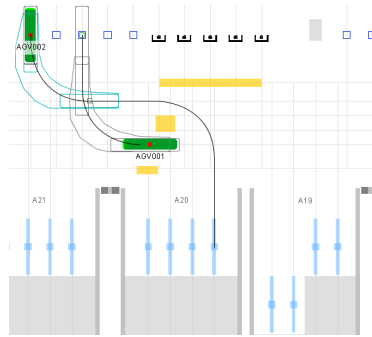


Fig. 7. AGVs crossing example 4

5.2 Key Performance Indicators (KPIs)

One of the objectives of the container terminal is to have a high productivity: to make as many movements as possible within a certain time frame. Therefore, the first KPI is formulated as follows: *Productivity: Total number of completed orders.*

However, there is a risk involved when this is the only KPI: it can make the container flow of the terminal unstable. An unstable container flow is defined as any scenario where the AGVs are not optimally utilized. An example of this is when AGVs with the shortest path will get priority over AGVs with a longer path. As a consequence, the amount of completed orders can be high because of the high frequency of the short-path AGVs, but meanwhile, the longer-path AGVs are compromised. In this case, the KPI is fulfilled, but the container flow is unstable.

In order to prevent an unstable container flow, a second KPI is added: *Container flow*: The lowest number of completed orders by an AGV. For the Productivity KPI and the Container flow KPI, the target value for the proposed algorithm is to be within 15% of the value for the currently used algorithm in the industry (TEAMS) or higher.

5.3 The experiments

In order to test the algorithm proposed in this paper, two experiments are set up.

Environment The environment where the algorithm is tested is a part of a container terminal, which consists out of four buffer and stacking areas. Figure 8 and 9 show an abstract depiction and a picture of the environment. The top area is the buffer area and the lower area is the stacking area. Within this environment, the AGVs have access to two highways, which are indicated by the yellow arrows and lines.

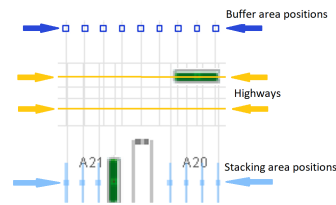


Fig. 8. Environment with AGVs, highways are highlighted



Fig. 9. Picture of the environment

Scenarios Two scenarios are used to compare the proposed algorithm to the industry used algorithm TEAMS and the Liu et al. algorithm. In both scenarios, two AGVs have access to two highways. This leads to either both AGVs choosing their own highway or both choosing the same.

Figure 10 shows scenario A, in which the blue circles indicate the start and end point of AGV one and the red circles indicate the start and end point of AGV two. The desired situation is that both AGVs learn to drive in their own highway and as a consequence never cross.

In scenario B, shown in figure 11, the desired situation depends on the state of the AGVs, such as their location and destination. AGV two (red circles) has the shortest route when the AGV takes the highway closest to the starting point. However, this is only preferable when there is no breaking involved to give priority to the other vehicle.

Table 4 shows the amount of AGVs in the experiment scenarios, as well as their start and end point. This will very likely result in the challenging situations seen in container terminals as described in subsection 5.1.

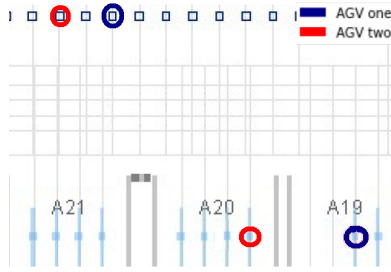


Fig. 10. Scenario A

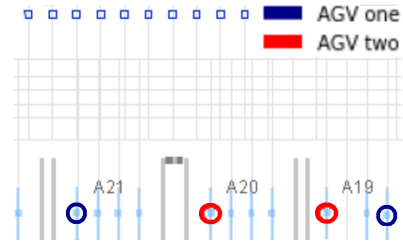


Fig. 11. Scenario B

Scenario	AGVs	Start/endpoint 1	Start/endpoint 2
A	2	Stacking areas	BA
B	2	Stacking areas	Stacking areas

Table 4. Test scenarios reinforcement learning script

6 Sensitivity analysis

A sensitivity analysis is conducted to examine how the settings of the model influence the outcome during the experiment phase. Four different set ups regarding the parameters are created and compared to each other in a preliminary experiment.

6.1 Determining the parameter settings during learning

In this preliminary experiment, the settings of the algorithm are manipulated to see whether this has a significant influence on the results of the algorithm. Four experiments are conducted with different setups, as shown in table 5.

The first column shows the setup number for reference. The second shows the learning rate α , the third shows the value for the epsilon greedy ϵ , and the last column shows the setting for the new Generated Action (GA).

Learning rate α The learning rate influences the effect the measured reward has on the value of the action. The higher the Learning rate α , the more aggressive the algorithm will react to the value of the last reward. Because of the unknown environment we want the Q value to adapt relatively quickly, without the system becoming unstable.

For setup 1, an α value of 0.1 is chosen. This is relatively low compared to other values in the literature. For setup 2, an α value of 0.3 is chosen. This is relatively high compared to other values in the literature. Between setup 1 and 2 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Epsilon ϵ Finding the right balance between exploration and exploitation is essential in RL. Exploration is finding new information on the environment in order to

discover a new and potentially better policy. Exploitation is using the current information for reward maximization by choosing the current action with the highest action value.

The settings for ϵ influences the chance that a greedy action is chosen. When epsilon is 1, all actions are exploitation and therefore a greedy action is never chosen. If the epsilon is 0, a greedy action is always chosen and no actions will be exploited. Therefore, ϵ indicates the balance between exploitation and exploration.

For setup 2, an ϵ value of 0.9 is chosen. This is relatively high compared to other values in the literature. For setup 2, an ϵ value of 0.7 is chosen. This is relatively low compared to other values in the literature.

Between setup 2 and 4 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Generated Action (GA) When the algorithm is exploiting, it measures the action with the highest value. When the algorithm is exploring, it has two options: to explore an existing action that does not have the highest value, or to generate a new action. The setting for GA influences the chance that either option is chosen. The higher the value of GA, the higher the chance that a new action is measured.

For setup 3, a GA value of 0.5 is chosen, so the algorithms opts for new actions equally as often as for existing actions. For setup 4, a GA value of 0.3 is chosen. In this case, the algorithm opts more often for existing actions than for new actions. This way, comparing these setups can show the effect of the amount of generated actions on the results of the algorithm.

Between setup 3 and 4 this is the only parameter which is adapted and can therefore give a good indication of the influence.

Setup during learning	alpha	epsilon	Generated Action (GA)
1:	0.1	0.9	0.3
2:	0.3	0.9	0.3
3:	0.3	0.7	0.5
4:	0.3	0.7	0.3

Table 5. Parameter settings experiments during learning phase

6.2 Parameter settings evaluation time

With the learned policy the agent executes the orders during 30 minutes. The agent is able to change the action values, as alpha is not equal to 0. This is done so that the learned best action will continuously be evaluated. To keep the behaviour changes minimized, the greedy actions are taken out of the learning process so the policy is more stable during the evaluation. The settings of the parameters are shown in table 6.

Setup during evaluation	alpha	epsilon	Generated Action (GA)
1:	0.1	1	0
2:	0.3	1	0
3:	0.3	1	0
4:	0.3	1	0

Table 6. Parameter settings experiments after learning phase

6.3 Results parameter settings

The results in table 7 are based on the average of two emulations of 30 minutes after a learning phase of 3 hours in an emulation where the time is 4x accelerated.

Comparing the learning rate α using setup 1 and 2, we can see that in both experiment A and B, setup 2 performs better. So far the α setting of $\alpha = 0.3$ is preferred.

Comparing the balance between exploitation and exploration ϵ using setup 2 and 4, we can see that in both experiment A and B, setup 2 performs better. So far the ϵ setting of $\epsilon = 0.9$ is preferred.

Comparing the balance between exploring new and existing actions GA using setup 3 and 4, we can see that in both experiment A and B, setup 4 performs better. So far the GA setting of $GA = 0.3$ is preferred.

Setup	Movements experiment A (S-curve)	Movements experiment B (U-curve)
1:	246	238
2:	265	261
3:	201	181
4:	229	213

Table 7. Results 30 min emulation experiments after learning phase of 180 minutes

Sensitivity analysis As shown above, the settings of the parameters influence the behaviour of the model and this behaviour can be argumentative declared. One can conclude that the model is sensitive to the parameters of the model.

Preferred settings for the experiments The preferred setup is alpha = 0.3, epsilon = 0.9 and GA = 0.3. This corresponds to setup 2, which performs the best overall.

6.4 Learning time

To determine the Learning time of the experiments, three emulations were conducted during 7 hours at a speed of 4x real time. All emulations were chosen to take place in scenario A: the S-curve, as the AGVs have to travel more in this scenario compared to scenario B. Therefore, the amount of new states is expected to be higher in scenario

A. The results are shown in figure 12. The blue line is emulation 1, the orange line is emulation 2, the green line is emulation 3 and the red line is the average of all three emulations. The y axis shows the amount of new states that were found during the time interval of 30 minutes. The x axis shows the emulation time in hours.

In all three emulations, the amount of new states decreased to nearly zero after seven hours. As the graph shows, the amount of new states fluctuates up to 4.5 hours. After 4.5 hours, the fluctuation is negligible. Therefore the algorithm is trained with a learning time of 4.5 hours, which is 270 minutes.

The equipment emulator is set at a speed of 4x real time. The learning time of 4.5 hours thus corresponds to 18 hours of real time.

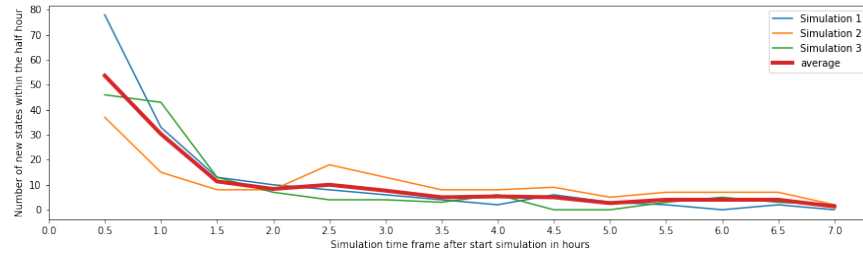


Fig. 12. New states per hour simulating the S scenario

7 Verification and validation

Action matrix During the experiments the action matrices are created based on the historical data. The data that is created by the model and shown in the emulation of the terminal is verified with the matrix that is created by the model. The route that is taken by the AGV and the time spent in each part of the map corresponds with the matrix created by the model.

An example is matrix 1 where the AGV drove without any disturbances on highway 1. The time changes when accelerating and decelerating near the start and end point. While driving on the highways there is no time spent in other areas than the highway. The areas of the highways in between the stacking areas are slightly larger and this can be seen in that matrix as well.

$$A = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1.303 & 0.79 & 0.53 & 0.227 & 0.523 & 0.297 & 0.243 & 0.297 & 0.25 & 0.3 & 0.483 & 0.233 & 0.54 & 0.787 & 1.367 & 0.0. \\ 0. & 2.017 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 2.336 & 0.0. & 0.0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.0. \end{pmatrix} \quad (1)$$

Order The AGV needs to give a signal to the model when it arrives at its final destination. This way a new measurement can start when a new order is created.

Highway check The highway is determined by the algorithm and sent to the emulator. The emulator executes the order as given by the algorithm, as we can see that the AGV drives along the route as ordered by the algorithm.

7.1 Validation

To see whether the algorithm can be applied in the real world, the choice was made to let the algorithm control software which is used in the industry to control container terminals all over the world. The algorithm is able to control the AGVs without issues.

8 Results scenario A: the S-curve

In this section the performance of the VARLPPA (Vehicle Aware) is compared to the performance of the RLPPA (Single Vehicle) and the algorithm used in the industry (TEAMS). The learning time is 4.5 hours and the emulation time is 60 minutes at a speed of 4x real time. The following subsections display the results of the ratio of highway choice and both Key Performance Indicators.

8.1 Highway distribution

To see how the behaviour of the AGVs is influenced by the algorithms, this experiment has collected data regarding the choice of highway. Figure 13 and 14 show the behaviour of AGV 1 and AGV 2, respectively. Each of the graphs show the ratio of which highway is chosen by each AGV. The x axis shows the percentage of the choice of Highway one and Highway two. The y axis show the results for each algorithm.

AGV1 The Single Vehicle algorithm learned a preference for highway one by 56.4 percent, the Vehicle Aware algorithm learned a preference for highway two by 60.1 percent and TEAMS had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, TEAMS distributed the movements to both highways equally.

AGV2 The Single Vehicle algorithm learned a preference for highway one by 70.7 percent, the Vehicle Aware algorithm learned a preference for highway two by 59.2 percent and TEAMS had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, TEAMS distributed the movements to both highways equally.

8.2 KPI 1 Productivity: Total number of completed orders

To measure the productivity of each algorithm, the total number of completed orders (movements per hour) was measured. Figure 15 shows the total movements per hour for both AGVs for each algorithm after 270 minutes of learning.

TEAMS completed 476 movements per hour, the Single Vehicle algorithm completed 501 movements per hour and the Vehicle Aware algorithm completed 530 movements per hour.

Both the Single Vehicle algorithm and the Vehicle Aware algorithm performed better in terms of movements per hour compared to TEAMS.

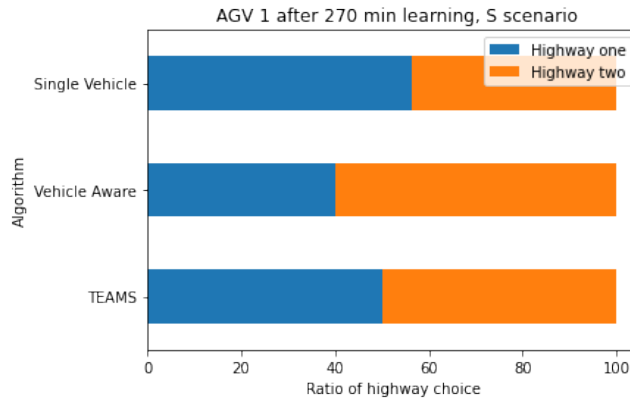


Fig. 13. Ratio of highway choice AGV 1 S scenario

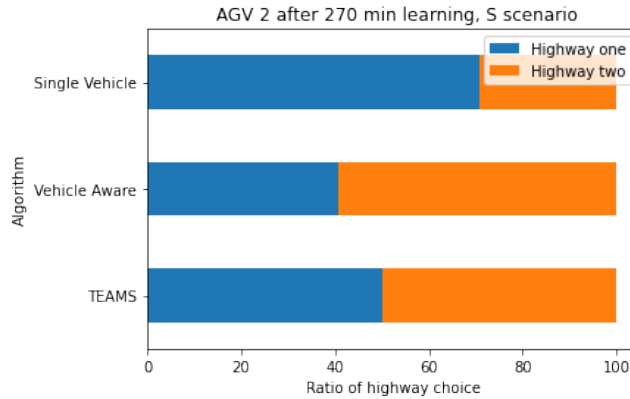


Fig. 14. Ratio of highway choice AGV 2 S scenario

8.3 KPI 2. Container flow: The lowest number of completed orders by an AGV.

Figure 16 shows the lowest number of completed orders by each AGV.

When the Single Vehicle algorithm is executed, there is a small difference between the AVGs. The lowest amount of orders per hour was achieved by AGV 2, which completed 250 movements per hour. This is 49.9 % of the total amount of orders .

The Vehicle Aware algorithm also results in a small difference between the AGVs: AGV 1 completed 264 movements per hour, which corresponds to 49.8 % of the total amount of orders.

When using TEAMS, both AGVs make an equal amount of movements per hour.

The target value for the KPI is to be within 15% of the value for TEAMS or higher. Seeing as this is the case, the KPI is achieved.

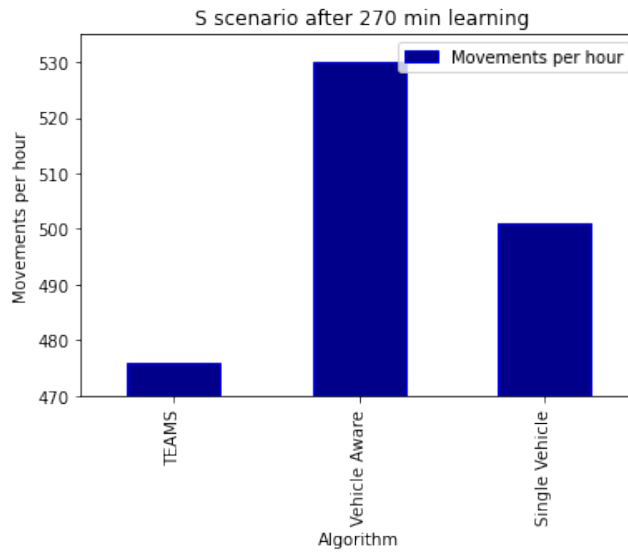


Fig. 15. Total number of movements per hour S scenario

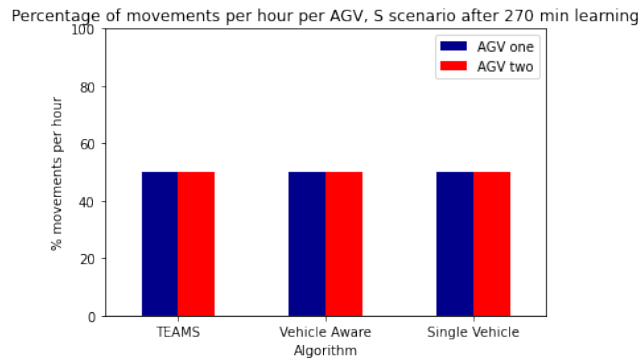


Fig. 16. Percentage of movements per hour per AGV, S scenario after 270 min learning

9 Results scenario B: the U-curve

In this section the performance of the VARLPPA (Vehicle Aware) is compared to the performance of the RLPPA (Single Vehicle) and TEAMS. The following subsections display the results of the ratio of highway choice and both Key Performance Indicators. The learning time is 4.5 hours and the emulation time is 60 minutes at a speed of 4x real time.

In this U-scenario, it is possible for the AGVs to choose the same route every time and never cross each other. Therefore, the researchers found a hypothesis for a best performing scenario, in which AGV 2 always takes the small U via highway 2, and

AGV 1 always takes the large U via highway 1. To test this hypothesis, a new scenario is added to the emulation. This scenario is called 'Forced Highway algorithm', and manipulates AGV 1 into taking the large U and AGV 2 into taking the small U. These results are described below.

9.1 Highway distribution

To see how the behaviour of the AGVs is influenced by the algorithms, this experiment has collected data regarding the choice of highway. Figure 17 and 18 show the behaviour of AGV 1 and AGV 2, respectively. Each of the graphs show the ratio of which highway is chosen by each AGV. The x axis shows the percentage of the choice of Highway one and Highway two. The y axis show the results for each algorithm.

AGV1 The Single vehicle learned a preference for highway two by 73.3 percent, the Vehicle Aware algorithm learned a preference for highway two by 67 percent, and TEAMS had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, TEAMS distributed the movements to both highways equally. The 'Forced Highway algorithm' is forced to travel via Highway 1.

AGV2 The Single vehicle learned a preference for highway two by 70 percent, the Vehicle Aware algorithm learned a preference for highway two by 93.3 percent, and TEAMS had assigned each highway to each direction: the AVG travelled west on highway 2 and east on highway 1. This way, TEAMS distributed the movements to both highways equally. The 'Forced Highway algorithm' is forced to travel via Highway 2.

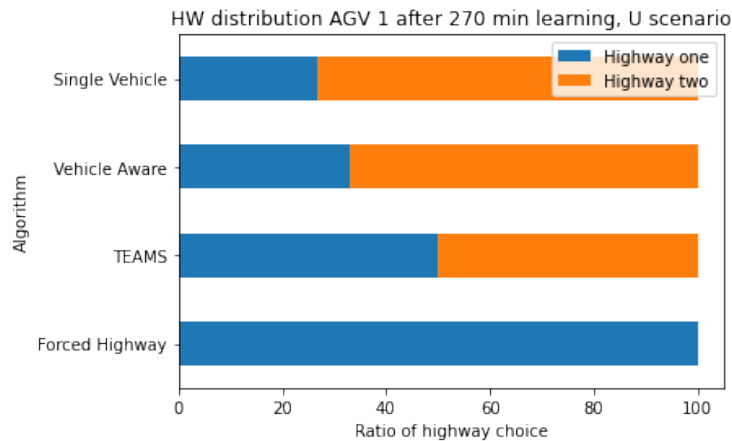


Fig. 17. Ratio of highway choice AGV 1 U scenario

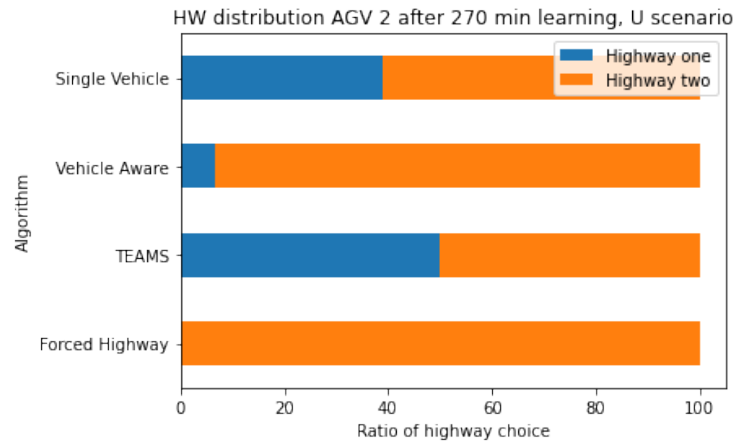


Fig. 18. Ratio of highway choice AGV 2 U scenario

9.2 KPI 1 Productivity: Total number of completed orders

To measure the productivity of each algorithm, the total number of completed orders (movements per hour) was measured. Figure 15 shows the total movements per hour for both AGVs for each algorithms after 270 minutes of learning.

The forced highway algorithm completed 504 movements per hour, TEAMS completed 473 movements per hour, the Single vehicle algorithm completed 494 movements per hour and the Vehicle Aware algorithm completed 518 movements per hour.

All algorithms are within 15% of the movements per hour, compared to TEAMS. The Vehicle Aware algorithm achieved the highest amount of movements per hour.

9.3 KPI 2. Container flow: The lowest number of completed orders by an AGV.

Figure 20 shows the lowest number of completed orders by each AGV. When the Single Vehicle algorithm is executed, the lowest amount of orders per hour was achieved by AGV 1, which completed 226 movements per hour. This is 45.7 % of the total amount of orders. In case of the Vehicle Aware algorithm, AGV 1 completed 234 movements per hour, which corresponds to 45.2% of the total amount of orders. When using TEAMS, AGV 1 completed 221 movements, which corresponds to 44.7 % of the total. Lastly, in the Forced Highway scenario, AGV 1 completed 218 movements per hour, which is 43.3% of the total.

As described above, each scenario shows the same behaviour of AGV 1 achieving fewer movements per hour than AGV 2. This behaviour can be explained by the fact that AGV 1 has to drive the large U which is a longer route and can therefore complete fewer movements per hour with respect to AGV 2. The target value for the KPI is to be within 15% of the value for TEAMS or higher. Seeing as this is the case, the KPI is achieved.

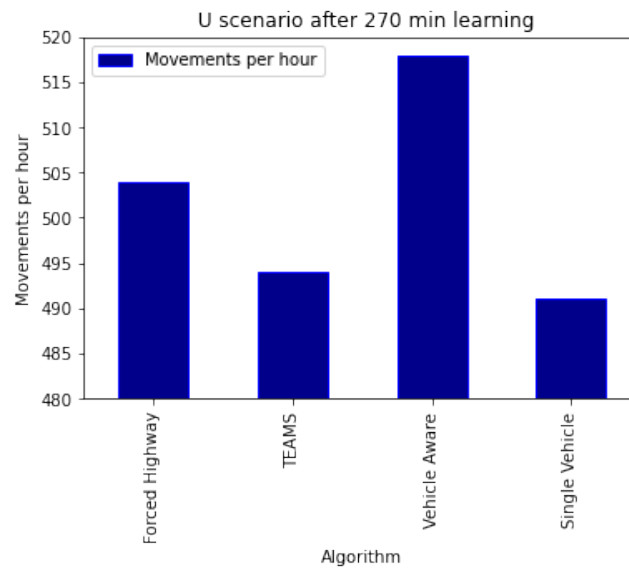


Fig. 19. Total number of movements per hour U scenario

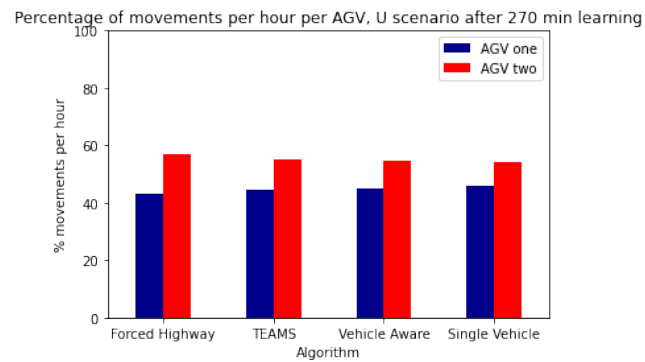


Fig. 20. Percentage of movements per hour per AGV, U scenario after 270 min learning

10 Evaluation of the algorithm

This section provides an analysis of the proposed algorithm.

10.1 How does the algorithm compare to the state of the art machine learning path planning, in terms of the KPIs?

The most innovative element of the VARLPPA, compared to the design of the OPABRL, is the vehicle aware element: at every state of the VARLPPA, the state of the other

vehicles is measured and taken into account, whereas the OPABRL is only used for the path planning of a single vehicle. In order to determine the added value of this element, the experiments discussed above have compared the VARLPPA with and without the vehicle aware element. These algorithms were called 'Vehicle Aware' and 'Single Vehicle', respectively.

KPI 1: Productivity The Vehicle Aware algorithm outperformed the Single Vehicle algorithm in both scenarios, in terms of KPI 1: movements per hour. In Scenario A, the performance was increased by 5.8% which is 29 movements per hour, and in scenario B, the performance was increased by 4.9%, which is 24 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm performed equally as well as the Single Vehicle algorithm in both scenarios, in terms of KPI 2: The lowest number of completed orders by an AGV. In both scenarios, the value of the Vehicle Aware showed a negligible difference between both AGVs.

10.2 How does the algorithm compare to the path planning algorithm industry used algorithm (TEAMS), in terms of the KPIs?

KPI 1: productivity The Vehicle Aware algorithm outperformed the in the industry used algorithm (TEAMS) in both scenarios, in terms of KPI 1: movements per hour. In Scenario A, the performance was increased by 11.3% which is 54 movements per hour, and in scenario B, the performance was increased by 9,5%, which is 45 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm performed equally as well as the Single Vehicle algorithm in both scenarios, in terms of KPI 2: The lowest number of completed orders by an AGV. In both scenarios, the value of the Vehicle Aware algorithm was within 15% of the value for the algorithm currently used in the industry (TEAMS).

10.3 Hypothesis evaluation: How does the algorithm compare to the path planning of the Forced Highway algorithm, in terms of the KPIs?

In the U scenario the hypothesis is tested whether the Forced Highway outperforms the proposed algorithm.

KPI 1: productivity The Vehicle Aware algorithm outperformed the Forced Highway algorithm, in terms of KPI 1: movements per hour. The performance was increased by 2.8% which is 14 movements per hour.

KPI 2: Container flow The Vehicle Aware algorithm outperformed the Forced Highway in terms of KPI 2: The lowest number of completed orders by an AGV. The lowest number of completed orders for Vehicle Aware was AGV 1 with 226 movements per hour, and the lowest number for the Forced Algorithm was 218 movements per hour.

10.4 Discussion

Vehicle behaviour The vehicle behaviour influences the results of the machine learning algorithm. For example, the AGVs in the TEAMS emulator are programmed to wait for other vehicles to pass when they are crossing their path. This is shown in figure 21. AGV 2 waits for AGV 1 to pass. In this case, the more efficient solution would be for AGV 1 to choose highway 1 instead of highway 2, so the AGVs never have to cross. However, since AGV 2 just waits for its turn, AGV 1 doesn't have to brake, which would indicate that highway 1 would be the better option, and thus never learns the optimal policy.

A solution would be to take the waiting time into account, so the action is influenced by the waiting time. This is a topic for further research.

Effect on other vehicles It is not possible for an AGV to learn the influence it has on the other AGVs. Each AGV learns the best route for itself based on the environment state. Although it has some information about the other AGVs' position and goal, as long as it is not disturbed by other AGVs, it will never learn to take another route. An example is given in figure 21. Further research is necessary to improve the algorithm.

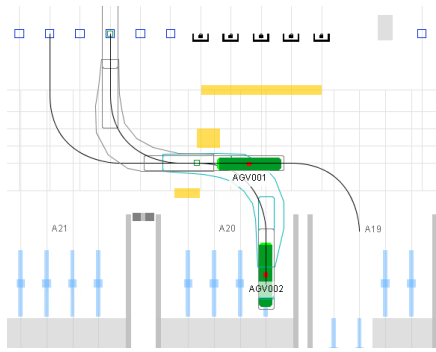


Fig. 21. Influence of the AGV behaviour

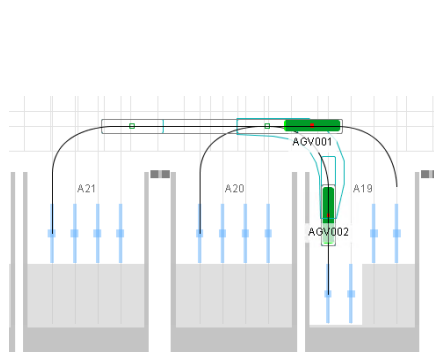


Fig. 22. Influence of the AGV behaviour

The size of the terminal The size of the terminal during this research is limited due to the scope of this research. Therefore, further research can be done to indicate if the scale of the environment influences the performance of the algorithm.

Learning from a real world terminal The model which has been used during the experiments is based on the data delivered by the manufacturers of the vehicles. Although this data is very close to reality, there is still some variation. Further research can be done by connecting the algorithm to a real terminal without executing the highway choice.

11 Conclusion

This thesis has provided insight into how machine learning can be beneficial to path planning in container terminals.

In automated container terminals, the Automated Guided Vehicles (AGVs) are guided by a path planning algorithm. In some cases, the layout leads to exceptional situations where case-specific solutions have to be found by the engineers to optimize the behaviour of the AGVs. Tuning or even building new software for each single situation requires a lot of manpower. Therefore there is potential in the integration of machine learning in software for the path planning systems in automated container terminals.

This thesis has proposed the machine learning algorithm VARLPPA. This algorithm uses the Epsilon-Greedy Monte Carlo Control method. The algorithm consists out of nine phases: initialization, checking availability of an AGV, translation of the measurement to the cost matrix, calculation of the action value, observation of the current state, action selection, route calculation, communication with the emulator, and when the set time is reached, storage of data.

The algorithm was tested in terms of scientific academic innovation and the value for the industry by multiple experiments.

The proposed Machine learning algorithm has been proven to be capable of learning a policy which can outperform the in the industry used algorithm, as well as the algorithm with the single vehicle states used in state of the art reinforcement learning path planning algorithms. By observing the position and destination of the current AGV, as well as the position, orientation and destination of the surrounding AGVs, the algorithm is capable to find a policy that is able to plan paths efficiently.

The results show that the algorithm which includes the Vehicle Aware element performs better regarding the KPIs than the algorithm that does not. The Machine Learning can increase the efficiency of path planning for automated guided vehicles at a container terminal, provided that the states are chosen well.

The thesis has shown that it is possible to increase the efficiency of path planning in a container terminal by applying machine learning and vehicle awareness to the path planning algorithm.

Bibliography

- Rina Dechter and Judea Pearl. Generalized Best-First Search Strategies and the Optimality of A. *Journal of the ACM (JACM)*, 32(3):505–536, 1985. ISSN 1557735X. <https://doi.org/10.1145/3828.3830>.
- František Duchon, Andrej Babinec, Martin Kajan, Peter Beno, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified A star algorithm for a mobile robot. In *Procedia Engineering*, volume 96, pages 59–69, 2014. <https://doi.org/10.1016/j.proeng.2014.12.098>.
- Stuart Eiffert, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. Path Planning in Dynamic Environments using Generative RNNs and Monte Carlo Tree Search. Technical report, 2020.
- Ariel Keselman, Sergey Ten, Adham Ghazali, and Majed Jubeh. Reinforcement Learning with A* and a Deep Heuristic. Technical report, 2018. URL <http://arxiv.org/abs/1811.07745>.
- Xiao Huan Liu, De Gan Zhang, Hao Ran Yan, Yu Ya Cui, and Lu Chen. A New Algorithm of the Best Path Selection Based on Machine Learning. *IEEE Access*, 7:126913–126928, 2019. ISSN 21693536. <https://doi.org/10.1109/ACCESS.2019.2939423>.
- Changhyeon Park and Seok Cheol Kee. Online local path planning on the campus environment for autonomous driving considering road constraints and multiple obstacles. *Applied Sciences (Switzerland)*, 11(9), 2021. ISSN 20763417. <https://doi.org/10.3390/app11093909>.
- Ryuichi Shibasaki. *Data [2] container shipping demand for the present and future*. Elsevier Inc., 2021. ISBN 9780128140604. <https://doi.org/10.1016/b978-0-12-814060-4.00009-5>. URL <https://doi.org/10.1016/B978-0-12-814060-4.00009-5>.
- Valentyn N. Sichkar. Reinforcement learning algorithms in global path planning for mobile robot. *2019 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2019*, pages 10–14, 2019. <https://doi.org/10.1109/ICIEAM.2019.8742915>.
- Richard S.Sutton and Andrew G.Barto. *Reinforcement Learning: An Introduction*, volume 4. 2020. ISBN 9780262039246. URL <http://marefateadyan.nashriyat.ir/node/150>.
- Tianqi Zha, Lei Xie, and Jiliang Chang. Wind farm water area path planning algorithm based on A and reinforcement learning. *ICTIS 2019 - 5th International Conference on Transportation Information and Safety*, (2017):1314–1318, 2019. <https://doi.org/10.1109/ICTIS.2019.8883718>.
- Hong Mei Zhang and Ming Long Li. Rapid path planning algorithm for mobile robot in dynamic environment. *Advances in Mechanical Engineering*, 9(12): 2017, 2017. ISSN 16878140. <https://doi.org/10.1177/1687814017747400>. URL <https://us.sagepub.com/en-us/nam/>.
- Zhanying Zhang and Ziping Zhao. A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm. *International Journal of Smart Home*, 8(3):75–86, 2014.

ISSN 1975-4094. <https://doi.org/10.14257/ijsh.2014.8.3.07>. URL
<http://dx.doi.org/10.14257/ijsh.2014.8.3.07>.

Luowei Zhou, Pei Yang, Chunlin Chen, and Yang Gao. Multiagent reinforcement learning with sparse interactions by negotiation and knowledge transfer. *IEEE Transactions on Cybernetics*, 47(5):1238–1250, 2017. ISSN 21682267. <https://doi.org/10.1109/TCYB.2016.2543238>.