國立臺灣科技大學

工業管理系

碩士學位論文

學號：M11101801

使用深度 Q 學習優化路口交通控制於智動化揀貨系統

Optimizing Intersection Traffic Control in Robotic Mobile Fulfillment Systems Using Deep Q-Learning

研 究 生: Muhammad Hazdi Kurniawan

指導教授： 周碩彥 博士

郭伯勳 博士

中華民國 113 年 07 月

M11101801

碩士學位論文指導教授推薦書

Master's Thesis Recommendation Form

系所： Department/Graduate Institute	工業管理系 Department of Industrial Management
姓名： Name	Muhammad Hazdi Kurniawan Muhammad Hazdi Kurniawan
論文題目： (Thesis Title)	使用深度 Q 學習優化路口交通控制於智動化揀貨系統 Optimizing Intersection Traffic Control in Robotic Mobile Fulfillment Systems Using Deep Q-Learning

係由本人指導撰述，同意提付審查。

This is to certify that the thesis submitted by the student named above, has been written under my supervision. I hereby approve this thesis to be applied for examination.

指導教授簽章：
Advisor's Signature

周碩彥

共同指導教授簽章（如有）：
Co-advisor's Signature (if any)

郭伯勳

日期：
Date(yyyy/mm/dd)

2024 / 7 / 29

M11101801

碩士學位考試委員審定書

Qualification Form by Master's Degree Examination Committee

系所： Department/Graduate Institute	工業管理系 Department of Industrial Management
姓名： Name	Muhammad Hazdi Kurniawan Muhammad Hazdi Kurniawan
論文題目： (Thesis Title)	使用深度 Q 學習優化路口交通控制於智動化揀貨系統 Optimizing Intersection Traffic Control in Robotic Mobile Fulfillment Systems Using Deep Q-Learning

經本委員會審定通過，特此證明。

This is to certify that the thesis submitted by the student named above, is qualified and approved by the Examination Committee.

學位考試委員會

Degree Examination Committee

委員簽章：
Member's Signatures

郭伯勳　　

周碩彥　　許中雯　　

指導教授簽章：
Advisor's Signature　周碩彥

共同指導教授簽章（如有）：
Co-advisor's Signature (if any)　郭伯勳

系所（學程）主任（所長）簽章：
Department/Study Program/Graduate Institute Chair's Signature

日期：
Date(yyyy/mm/dd)　2024 / 7 / 29

ABSTRACT

The rapid development of Robotic Mobile Fulfillment Systems (RMFS) has significantly transformed the warehouse industry by integrating various activities and processes and enhancing the capabilities of individual robots. However, managing traffic flow and pathfinding for numerous autonomous robots within dynamic warehouse environments remains a critical challenge. This study explores the implementation of Deep Q-Learning (DQL) to optimize intersection traffic control for autonomous mobile robots in RMFS, aiming to reduce energy consumption and improve overall operational efficiency.

By creating virtual traffic lights controlled by DQL agents, this research addresses the frequent stop-and-go behavior and waiting times at intersections, which are significant contributors to energy consumption. The study involves designing and training a DQL model to manage intersection traffic dynamically. A comprehensive simulation environment mimicking actual RMFS layouts and operational dynamics was used to validate the proposed system.

The results indicate that the DQL-integrated system achieves a 3% reduction in total energy consumption compared to the baseline scenario. However, this improvement is accompanied by a significant increase in waiting times at intersections, highlighting the need for further refinement to balance energy efficiency with operational efficiency. Statistical analysis confirms the robustness of the DQL approach, emphasizing its potential to optimize energy consumption in RMFS environments.

Keywords: *Robotic Mobile Fulfillment Systems (RMFS), Deep Q-Learning (DQL), Traffic Control, Autonomous Mobile Robots, Energy Consumption, Intersection Management, Reinforcement Learning*

ACKNOWLEDGEMENT

First and foremost, I would like to extend my deepest gratitude to OpenAI for creating ChatGPT, a tool that has been nothing short of a lifesaver during my research and writing process. Without ChatGPT, there's a good chance this thesis might still be just a vague idea floating around in my head.

I am profoundly grateful to my advisor, Professor Shuo-Yan Chou, as well as Dr. Anindhita Dewabharata and Dr. Ferani Eva Zulvia, for their unwavering support, valuable suggestions, and constant encouragement. Their guidance and the numerous opportunities they provided have been vital to the successful completion of this thesis.

My sincere appreciation also goes to Professor Po-Hsun Kuo, my co-advisor, and Professor Yu-Ling Hsu, a member of my oral defense committee. Their thorough evaluation, insightful feedback, and consistent encouragement greatly enriched my research and contributed significantly to the quality of this work.

Finally, a big thank you to myself. Without my persistence, dedication, and perhaps a bit of stubbornness, I wouldn't have been able to complete this thesis. It's been a challenging journey, but I'm proud of what I've achieved.

Muhammad Hazdi Kurniawan

Taipei, August 2024

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

The development of Robotic Mobile Fulfillment Systems (RMFS), as demonstrated by industry giants such as Amazon, Swisslog, and Interlink, has transformed the warehouse business [1, 2]. These systems have improved rapidly, both horizontally, by integrating across many activities and processes, and vertically, by improving the capabilities and intelligence of individual robots. To illustrate, Figure ***1.1*** depicts several examples of different Autonomous Mobile Robots (AMRs) in the distribution centers [3]. In detail, from left to right are the robot MiR100 at Honeywell Safety & Productivity Solutions, then the CarryPick from Swisslog and the AutoStore RMFS system [3]. Instead, the focus of this work is based on the RMFS technology exemplified by the robots in Images 1 and 2. Nonetheless, AutoStore serves as a valuable reference to demonstrate the variety and innovation present in RMFS solutions within the industry. Another prominent example was when Amazon acquired Kiva Systems for 775 million US dollars [4].

Figure 1.1 RMFS examples [3]

As the whole warehousing sector moves ahead, it calls for more than just infrastructure expansion. With the huge number of autonomous robots traveling within the warehouse area, pathfinding and traffic flow are the next focus area for development and advancement. Consequently, the challenge of coordinating numerous autonomous robots in dynamic

environments has garnered considerable attention from researchers and industry stakeholders alike [5]. Specifically, since 2017, interest in this area has grown considerably [5]. Figure ***1.2*** illustrate the number of publications related to RMFS [5]. Addressing these concerns is critical for a variety of reasons. Effective traffic control and pathfinding tactics can significantly improve operational performance by reducing delays and avoiding deadlocks. Warehouses can improve the flow of robots to enable shorter lead time, help meet customers' expectation, resulting in higher customer satisfaction.

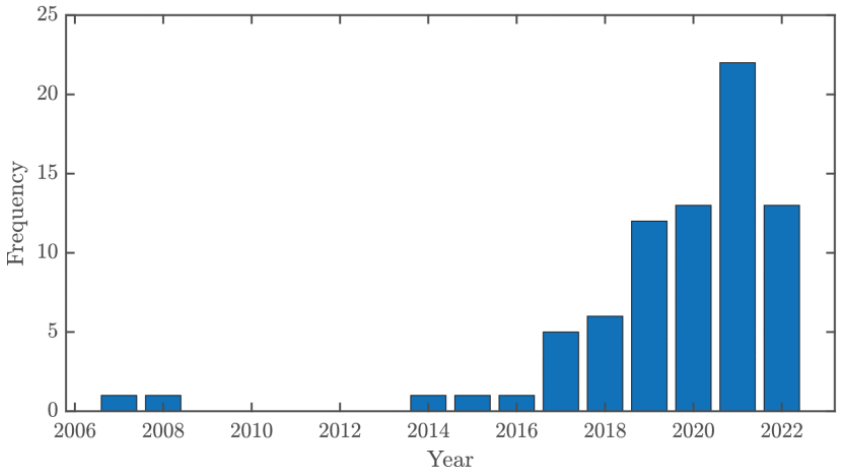

Figure 1.2 Number of publications related to RMFS [5]

This work will focus on the issue of intersection traffic control for many autonomous mobile robots, with an emphasis on energy consumption in dynamic warehouse environments. The goal is to create creative traffic control systems that not only manage the flow of robots efficiently, but also reduce energy consumption. By attaining these objectives, this study hopes to improve the entire operational performance of RMFS, ensuring that the increasing complexity and demands of modern warehousing are addressed with efficient, sustainable, and cost-effective solutions.

1.2 Objectives

This research's objectives can be described as follows:

- Implement of Deep Q-Learning (DQL) to optimize intersection traffic control for autonomous mobile robots in RMFS.
- Reduce energy consumption and improve overall operational efficiency.

1.3 Scope and Limitations

Taking into account all the previously mentioned circumstances, the following limitations of the research must be acknowledged:

- The study will confine to one intersection line closest to the picking station
- The simulations are conducted with the same order sequences. As long as the order throughput remains consistent within a similar timeframe and the waiting times are acceptable, the primary focus will be on optimizing energy consumption.

1.4 Organization of Thesis

The structure of this study includes five distinct chapters, each considering a specific aspect of the research. The structure of the study is as follows:

Chapter 1: Introduction. This chapter begins by explaining why this research is important and what prompted the interest in studying it. It establishes explicit objectives for the research and provides a glimpse of how the study will be organized.

Chapter 2: Literature Review. This chapter provides a complete assessment of the available literature on the research issue. The literature review provides a detailed summary of the status of the discipline, emphasizing key discoveries and contributions from prior studies.

Chapter 3: Methodology. This chapter elaborates on the research methods used in this study. It includes a full overview of the study's design, and analytical procedures. Furthermore, it provides a clear and simple description of the processes required to carry out the research, guaranteeing the study's conclusions are reliable and valid.

Chapter 4: Results and Discussion. This chapter examines the acquired data and reveals the study's findings. Additionally, it discusses these findings, taking into account both the observable outcomes and the statistical analysis.

Chapter 5: Conclusion and Future Work. This chapter summarizes the study's primary findings and examines how they may affect future work. Furthermore, it makes recommendations for future study projects.

CHAPTER 2

LITERATURE REVIEW

2.1 Robotic Mobile Fulfillment Systems (RMFS)

2.1.1 RMFS Decision Problems

Figure 2.1 A robot carrying a pod [1]

Robotic Mobile Fulfillment Systems (RMFS), a successful multi-robot warehouse system employing autonomous vehicles for efficient inventory movement. ***Figure 2.1*** given above display an example of a robot carrying a pod, which is typical and commonly seen in distribution centers [1]. This section will consider various decision-making issues commonly experienced within a Robotic Mobile Fulfillment System (RMFS). These systems often represent operational demands as pick or replenishment orders (as shown in ***Figure 2.2***). When these requests are received, pallets are methodically separated into smaller units, each with a specific Stock Keeping Unit (SKU). Individual units are carefully positioned on a pod to fulfill a replenishment order.

Recognizing the interconnected nature of these operational decision problems, it is crucial to cultivate a thorough comprehension of the current research in these fields. Numerous scholarly articles typically focus on addressing existing challenges or enhancing efficiency in one of these areas.

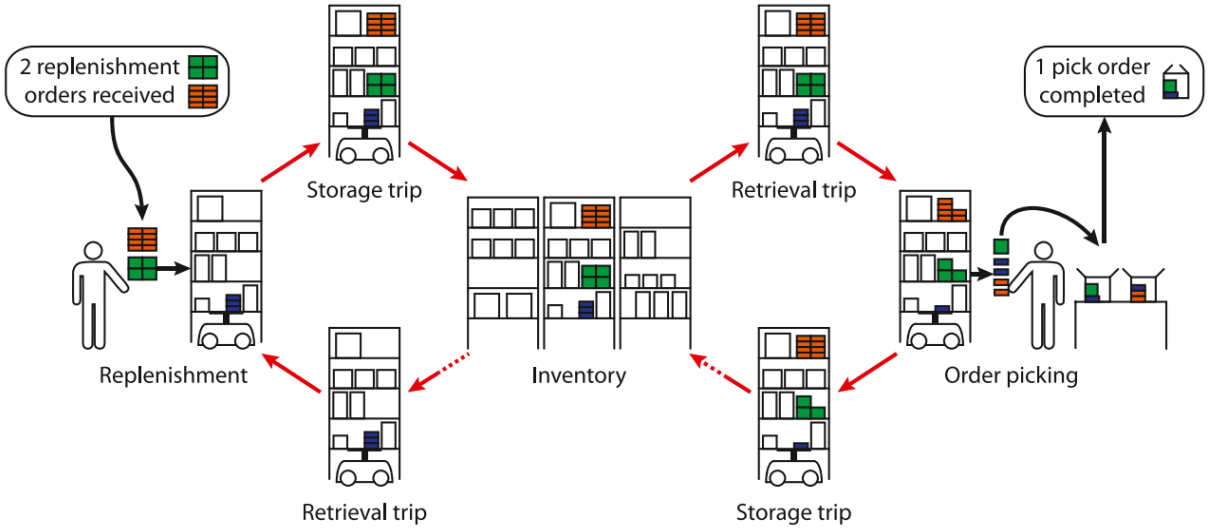

Figure 2.2 The internal storage/retrieval process in RMFSs [2]

According to Merschformann and partners, the RMFS operational decision problems can be classified into 4 main areas, which are listed as follows:

(1) **Order Assignment (OA):** OA decisions are classified into two types: assigning pick orders to pick stations (Pick Order Assignment or POA) and assigning replenishment orders to replenishment stations (Replenishment Order Assignment, or ROA).

(2) **Task Creation (TC):** Task Creation consists of two subproblems. The first is Pod Selection (PS), which identifies the selected pods for their storage trips. This subproblem includes Pick Pod Selection (PPS), which emphasizes the due times, and Replenishment Pod Selection (RPS), which employs an alternative approach. The second TC subproblem is Pod Storage Assignment (PSA), which determines the storage location for the pod's retrieval trip.

(3) **Task Allocation (TA):** This process assigns jobs to robots. Hence, a trip will be formed while a series of tasks are arranged for the robots.

(4) **Path Planning (PP):** Having sequential tasks inherently defines routes and provides input for Path Planning algorithms, which construct paths for robots to follow.

2.1.2 Previous research

First of all, item and pod allocation strategies are of significant emphasis. In their paper published in 2020, Kim and colleagues created a technique for efficiently assigning things to inventory pods [6]. Specifically, their article ensures frequently ordered products are placed in the same pod by presenting an efficient heuristic approach for assigning items to pods in an RMFS [6]. A data-driven approach for zone clustering and storage site assignment was proposed by Keung and partners in 2021, which met their purpose of improving operational efficiency in the RMFS [7]. Another method emerged in 2020 when Li and colleagues developed a high-density storage arrangement to increase space usage [8]. In detail, they utilize a cluster approach to temporal association analysis with the aim of identifying highly correlated items for storage in the same rack.

Additionally, a new turnover-rate-based decentralized storage policy (TRBDSP) is presented to eliminate AMR blockage. Ultimately, the findings suggest that the novel method can significantly enhance order-picking efficiency while lowering AMR energy consumption [8]. Related to this, Cechinel et al. propose an island model algorithm for multi-robot systems that takes into account journey time, energy consumption, robot payload, and order deadlines [9]. Consequently, their results reveal that the method produces a greater number of complete solutions, simultaneously meeting the deadlines and demonstrating energy consumption reductions [9].

Several approaches for order-picking have been proposed. Meller and Pazour devised a heuristic for SKU allocation in an A-frame automated system to improve order fulfillment efficiency in pharmaceutical distribution centers, which has been considered one of the early studies [10]. Furthermore, the split order concept was introduced as a new method to serve the purpose of enhancing overall efficiency [11]. Xie and colleagues further explained that this approach allows portions of an order to be selected at various stations [11]. Another approach was employed by Yuan and partners, in which a mathematical model was implemented with a two-stage hybrid algorithm technique to investigate pod correlation and pod placements in the storage arrangement [12]. Experimental results of their research later suggest that the proposed pod assignment model and algorithm optimization approach can improve order-picking efficiency [12].

By merging order and robot scheduling to improve picking efficiency, Allgor and colleagues particularly focus on the approach utilized by the Amazon Fulfillment Centers, which was the four-phase decomposition heuristic [13]. Most recently, in 2024, SASORL algorithm was introduced, and it is a reinforcement learning-based solution for optimizing order allocation and sequencing that demonstrated considerable reductions in travel distance and processing time [14].

Regarding path planning decision problems, also known as pathfinding problems, are path arrangement problems in which a group of robots must find collision-free paths with different starting and ending points [15]. When addressing this problem, critical elements such as acceleration, deceleration, and turning time must all be considered. If there is no longer a collision-free path, the robot must wait for one to become available. Henceforth, traffic control is inextricably tied to path planning problems since traffic control solutions can help minimize the delay time for the robots. Since 1968, Hart, Nilsson, and Raphael invented the A* approach, and its versions are widely employed to solve the pathfinding problem [16]. Several scholars have developed and constructed several extensions of A* to address its shortcomings and improve overall efficiency [17-19].

Later in 2013, Sharon and colleagues proposed the Increasing Cost Tree Search (ICTS) to find optimal solutions using a two-level search method [20]. Finally, the Conflict-Based Search (CBS) algorithm is another commonly employed search-based algorithm [21]. The mechanism of CBS approach is to divide a multi-agent pathfinding problem into multiple limited single-agent pathfinding challenges [21]. More recently in 2019, Li and colleagues proposed two new viable heuristics that take into account the pairwise relationships between agents and empirically show improvement in success rates and run-time [22]. In addition, there are papers implementing reinforcement learning with decentralized execution to solve this problem [23-27]. These papers will be discussed further in the following section, which will focus on the application of RL in RMFS research.

2.1.3 Traffic Control

As mentioned earlier, traffic control is intricately linked to the path planning stage in RMFS as a traffic control solution can reduce the deadlocks and delay time and subsequently enhance the system's overall efficiency. In 1988, traffic management regulations were established to navigate AMRs on a 2D grid while their motions were coordinated and linked by a control center [28]. This

policy is nearly optimal even for large numbers of AMRs and has since set a solid foundation for general RMFS traffic control and management.

Utilizing a decentralized mechanism, Bourbakis developed a formal modeling of a generic traffic priority language in 1997 [29]. This traffic control solution does not require a central control to coordinate; the robots can move freely and are referred to as autonomous. For this reason, Bourbakis's approach is considered helpful in navigating dynamic contexts. Similarly, Wang and Premvuti have employed distributed robotic systems (DRS) in their paper to set up traffic regulation [30]. Their system lets the AMRs dynamically create their own unique targeted route, which is unknown to other robots, yet the clock, memory, and ground support are shared and synchronized [30].

Delays typically occur at intersections while all the robots are moving. Minimizing the delay time at intersections is the shared target of traffic control in urban and warehouse contexts. Firstly, in the urban traffic context, Dunne and Potts, as the pioneers in this discipline, used rule-based solutions [31]. In their paper, the intersectional traffic signal will respond to a control algorithm based on the number of vehicles requiring service. The proposed linear control algorithm has proven stable in undersaturated situations [31]. With the same target of minimizing the total delay, a computer control scheme was introduced by Ross and colleagues [32]. Their aim differs from previously mentioned research as it concentrates on addressing the residual queues at a specific critical intersection [32]. In particular, reinforcement learning has greatly benefited real-time traffic signal control systems [33, 34]. These papers will be discussed later in the following section.

Since there are certain similarities between intersections in warehousing and urban contexts, we can refer to and learn from their approach to solving intersectional problems. Nonetheless, traffic flow between these two environments is different by nature, so the possible issues are also distinctive. Within the RMFS, intersections pose a significant barrier to maximizing the system's overall efficiency and energy usage. One of the outstanding and relatable research was in 2009, written by Teja and colleagues [35]. Their research assigns an intersection agent to govern intersectional traffic flow by providing priority to robotic agents entering the intersection [35].

To conclude, understanding different approaches to solving intersectional delay time is valuable, and therefore, the fundamental purpose of this research is to control the permissible

directions at crossings by using virtual traffic lights and developing and training a learning model to manage intersection traffic dynamically.

2.2 Deep Q Network (Reinforcement Learning)

2.2.1 Reinforcement Learning

This section examines and discusses the use of Reinforcement Learning (RL) in this study. Sutton and Barto characterize RL as an agent learning optimal behavior by interacting with its environment and receiving feedback as reward signals for each action [36, 37]. Based on this feedback, the agent then refines its control policy, which is appropriate for adjusting traffic signals to dynamic traffic scenarios. Several advantages highlight the efficacy of an RL method. RL overcomes this issue by allowing traffic lights to learn and alter their timings autonomously based on real-time observations, which is critical for ensuring adequate traffic flow [38-43]. In general, this adapt-and-learn strategy has the potential to minimize congestion while also improving travel times and providing environmental advantages.

Similar implementations have been applied to RMFS traffic control over the years. Implementing RL within RMFS has received much attention because of its effectiveness in controlling the dynamic and unpredictable nature of path planning and traffic circumstances [23-27, 33, 34, 42, 44]. Particularly in traffic control, Jin and Ma let each traffic signal learn and adapt to traffic conditions and make optimal timing decisions based on perceived system states [34]. The intelligent control method uses reinforcement learning with multiple-step backups to update each traffic signal's knowledge online [34].

Likewise, some articles use reinforcement learning to tackle path-planning difficulties. First, Chen and colleagues created a unique two-agent collision avoidance system in 2017 that employed deep reinforcement learning, effectively shifting the computational load from online execution to offline training [24]. They also proposed a systematic approach for cases with more than two actors and an extended formulation that considers kinematic limitations [24]. Sartoretti and colleagues presented a novel solution to multi-agent pathfinding problems in 2019 by combining distributed reinforcement learning with imitation learning from a centralized expert planner [26]. Another work published by Long and colleagues contributed by presenting a multi-scenario, multi-stage deep reinforcement learning system that uses the policy gradient approach to establish the optimal collision avoidance strategy [25].

2.2.2 Deep Q-Learning

Q-learning determines the value of performing a specific action in a given condition, increasing the overall reward over time. It is one of the Reinforcement Learning approaches that does not need models. Watkin proposed the convergence theorem for Q-learning in 1989 [45], and later on presented it again and proved it in his publish with Dayan in 1992 [46]. In their paper in 2019, a Q-Learning flow chart was depicted, in which Chang and colleagues demonstrated the agent obtaining states and rewards from the environment based on its actions [47]. It calculates rewards for activities in each state through a Q-table and is updated using a method that weighs immediate rewards against expected future rewards. Eventually, an optimal policy is learned, enabling it to take the best action possible in each state to attain its goals.

With this mechanism, Q-Learning is appropriate for a dynamic environment as it does not involve prior awareness of the environment. Particularly, RMFS layout has such a dynamic nature, which means it can easily be changed, even the batches sometimes different for all warehouse, so one single model for all warehouse is not a sufficient selection. Furthermore, Q-learning seeks to identify the optimal policy to maximize the predicted future reward while simultaneously obeying a different policy [48]. Q-learning has been widely used in traffic signal control in both urban and robotic contexts due to its benefits and remarkable advantages [23, 33]. In urban traffic control, Jin and Ma proposed an adaptive signal control system based on a Q-Learning algorithm and a group-based phasing technique [33]. Simulation results indicate that RL-based techniques outperform fixed-time signal regulation by a significant margin, independent of demand levels [33].

Notably, deep Q-Learning is also adopted in EMFS traffic control, specifically in Ma and colleagues published in 2021 [23]. It is explained that instead of mapping each state-action pair to the corresponding value, deep Q-Learning utilizes a deep neural network to map its states and actions [23]. In detail, a single agent's perspective is employed to treat each agent separately and train the model, and subsequently, the final taught policy is applied to each agent during decentralized execution [23]. Ultimately, it is shown that there is a high success rate with a low average step in an empirical evaluation with a multiple-obstacle environment.

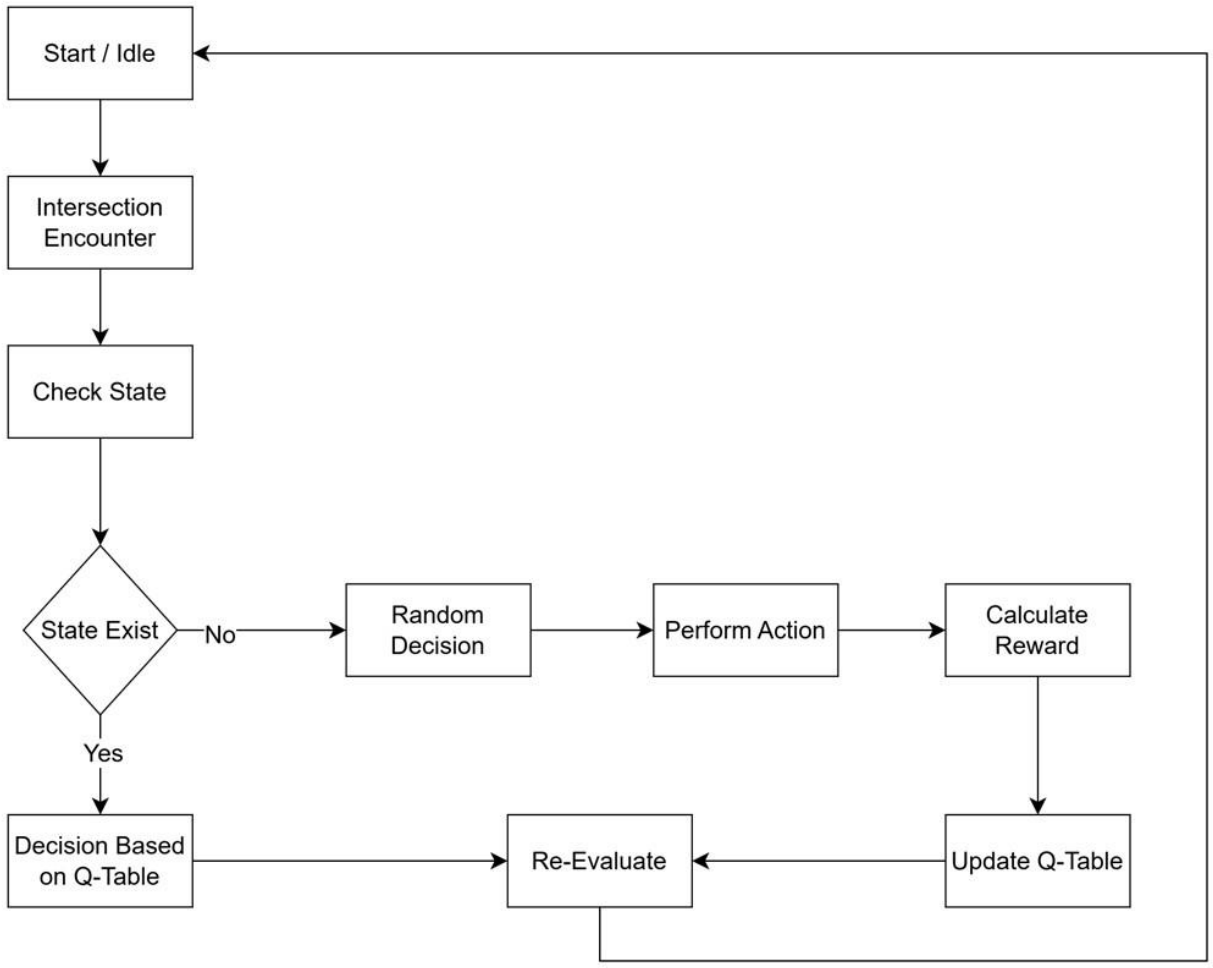

Figure 2.3 Deep Q-learning flow chart

Finally, this paper will employ Deep Q-Learning to manage the permissible intersectional directions, resulting in virtual trssaffic signals. Figure ***2.3*** illustrate the deep Q-learning flow chart which will be implemented in this paper. The primary goal is to reduce the stop-and-go behavior and the waiting time. In addition, minimizing overall energy consumption inside the RMFS will also be taken into consideration. Henceforth, this study will design and train a Deep Q-Learning model to dynamically manage intersection traffic while emphasizing energy savings.

CHAPTER 3
METHODOLOGY

3.1 Problem Definition

In the Robotic Mobile Fulfillment System (RMFS), the management of intersections is critical for optimizing both system throughput and energy consumption. Traditionally, intersections operate without advanced decision-making models, relying on a simplistic priority system where robots with higher priority or closer proximity proceed first. This approach can lead to suboptimal traffic flow, particularly in cases where deadlock occurs or where certain directions, such as vertical lanes that can only accommodate two robots, become bottlenecks, causing inefficiencies elsewhere in the system.

The challenge arises from the lack of dynamic control over intersection traffic, which often results in unnecessary idling, increased stop-and-go movements, and higher energy consumption. These inefficiencies highlight the need for a more intelligent system to manage traffic at intersections.

The primary goal of this research is to implement Deep Q-Learning to optimize the control of allowed directions at intersections, effectively creating adaptive virtual traffic lights. By dynamically adjusting the flow of robots based on real-time conditions and priority levels, the model aims to reduce energy consumption while maintaining or improving throughput. This research explores how Deep Q-Learning can be leveraged to make intersection management more energy-efficient without compromising the system's operational efficiency.

3.2 System Overview

3.2.1 Process Flow

In this section, we will explore the detailed process flow within the RMFS. The following sequence diagram (Figure ***3.1***) provides a comprehensive overview of the interactions between various components, including the Order, Pod (Storage Area), Robot, Picking Station, and Replenishment Station.

The RMFS process flow can be divided into several key stages, each involving different interactions and tasks, and are explained as follows:

(1) **Order Arrival:** When an order arrives, the process is triggered by assigning the order to a suitable pod within the storage area. The system identifies the appropriate pod to fulfill the order's requirements.

(2) **Order to Pod Assignment:** The selected pod undergoes sequencing to ensure the items can be picked efficiently. Sequencing involves arranging the items in a manner that optimizes the picking process. Once sequenced, the pod is assigned to a robot.

(3) **Pod to Robot Assignment:** The robot is routed to the storage area to pick up the assigned pod. The robot navigates through the warehouse to reach the pod's location. After picking up the pod, the robot is assigned to a picking station where the items will be picked.

(4) **Robot Routing:** The robot routing process involves assigning a route to the robot for navigation within the warehouse. This step is essential for directing the robot to the correct location, whether picking up a pod, transporting it to a station, or returning it to storage. The robot uses this routing information to navigate efficiently and avoid obstacles.

(5) **Robot to Station Assignment:** At the picking station, the robot delivers the pod to the designated station assigned to the order. The operator at the picking station picks the required items from the pod. The robot holds the pod in place while the operator retrieves the necessary items for the order. Once the items are collected, the robot returns to the storage area to return the pod.

(6) **Robot to Storage Assignment:** After the items are picked, the robot returns to the storage area to return the pod to its designated place. This involves navigating the warehouse and ensuring the pod is correctly placed back in storage.

(7) **Replenishment *(If required)*:** If the pod requires replenishment, the system checks its status to determine whether it needs restocking. The pod is assigned to a robot to route to the replenishment station if replenishment is needed. The operator replenishes the pod with the required items at the replenishment station. After replenishment, the robot routes the replenished pod back to the storage area, completing the cycle.

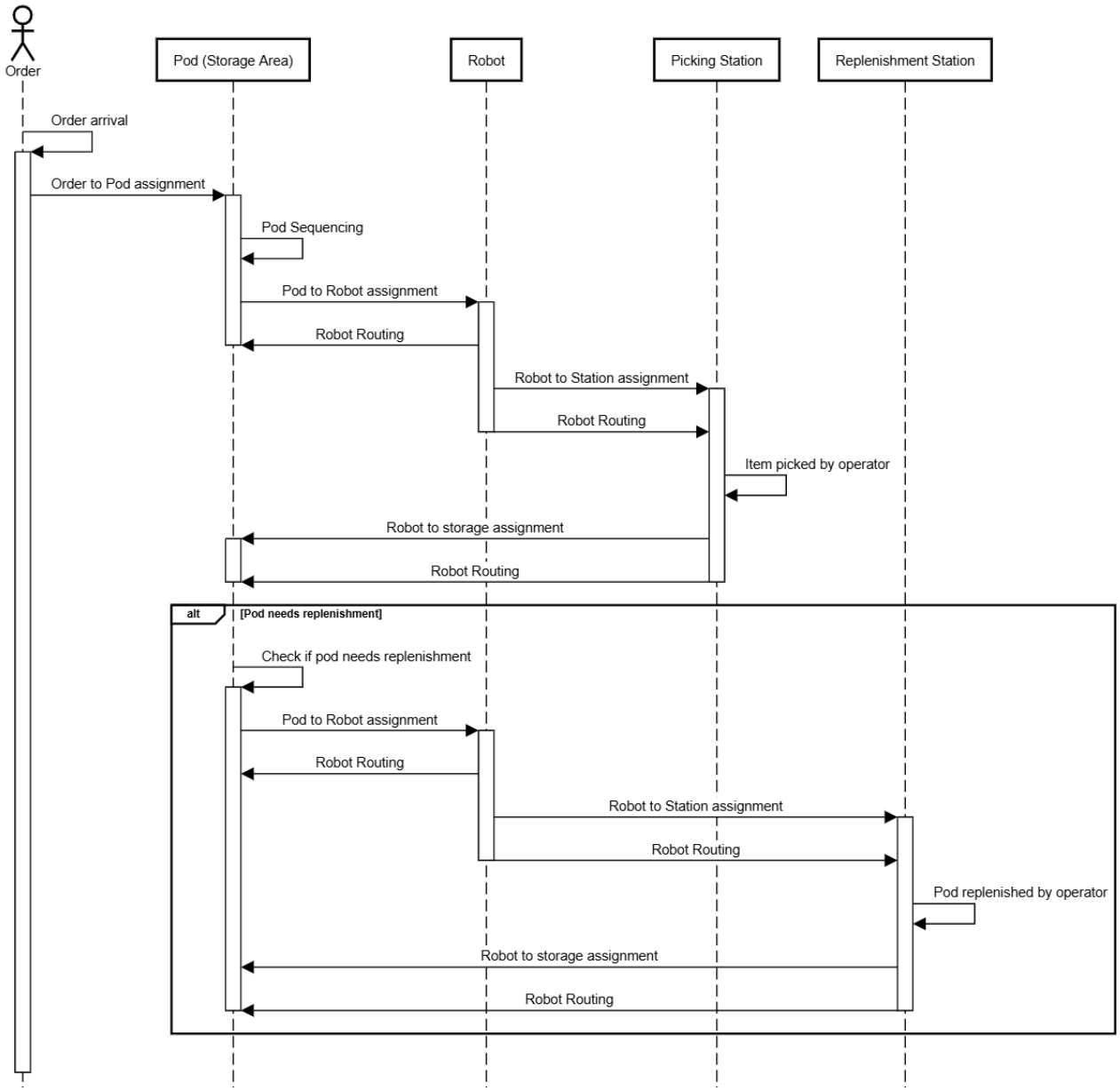

Figure 3.1 RMFS Process Flow Sequence Diagram

3.2.2 System Architecture

In this section, we will discuss the system architecture of our simulation, illustrated in Figure 3.2. NetLogo is the leading software used for simulation, responsible for visualization and output display [49]. All the functions are executed in Python due to its advantages in handling optimization algorithms, a task where NetLogo faces some challenges despite its strengths in agent-based modeling [50]. The ability of agents to communicate with each other in NetLogo is beneficial for solving various problems, but Python better handles the optimization capabilities required for this simulation.

NetLogo and Python interact seamlessly, with NetLogo calling functions and receiving feedback directly from Python. This interaction allows NetLogo to send arrays as input to Python functions and receive results in the same format. However, when dealing with large amounts of information, using a file or database for data transfer is more convenient. In this simulation, Excel is used to store and transfer data. This method facilitates data management and helps maintain a history of the entire simulation process. It is important to note that NetLogo does not have direct access to Excel; the Python functions handle all transactions involving Excel.

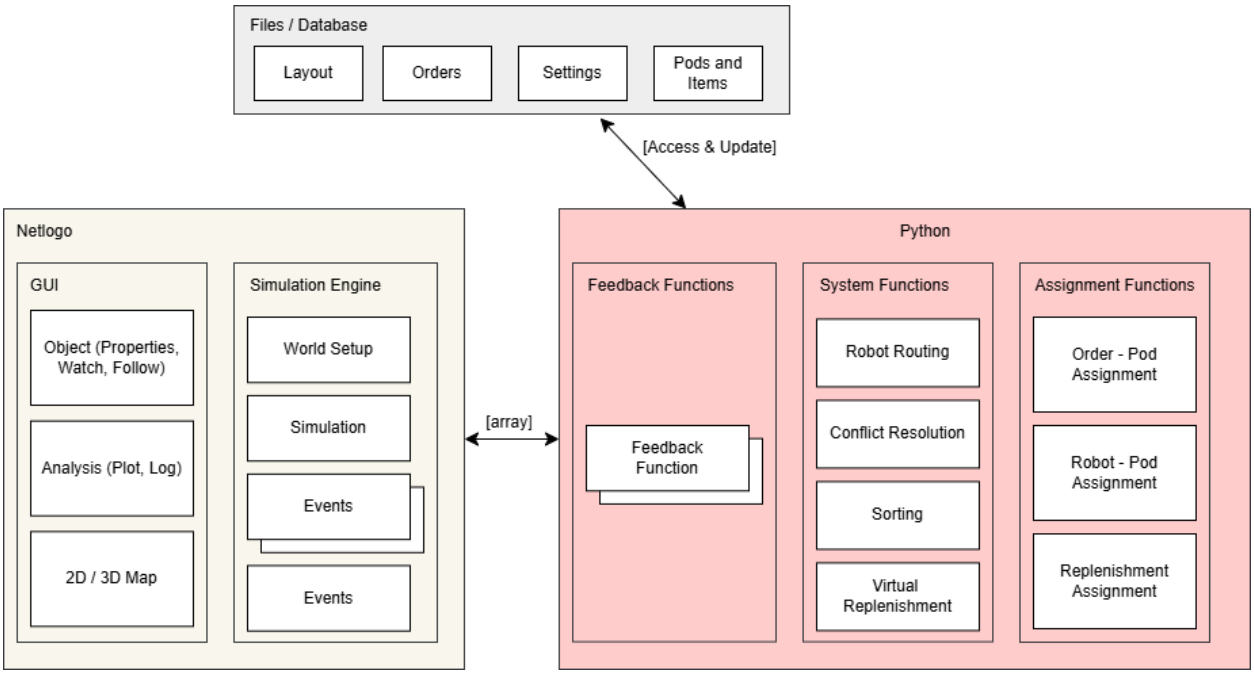

Figure 3.2 RMFS System Architecture Diagram

The Graphical User Interface (GUI) in NetLogo is crucial in visualizing the entire order-picking process, including the movement of AMRs and pods. The simulation output is displayed

on the GUI as plots or numerical values, which are updated automatically. Additionally, the GUI provides detailed agent information, accessible by selecting an agent with a click. This information is also updated in real-time as the simulation progresses, allowing users to monitor and analyze the behavior and performance of individual agents effectively.

This robust system architecture, depicted in Figure 3.2, ensures seamless integration between NetLogo and Python. NetLogo's strengths in agent-based modeling and visualization, combined with Python's capabilities in optimization and data handling, result in a comprehensive solution for simulating and optimizing the order-picking process in a warehouse environment.

3.2.3 Simulation

This section provides an in-depth discussion of the virtual traffic light system implemented within the Robotic Mobile Fulfillment System (RMFS). This system is pivotal in managing robotic traffic at critical intersections to optimize flow and reduce energy usage.

- **System Design:**

The virtual traffic light system consists of a decentralized network of control nodes, each located at strategic intersections within the RMFS. These nodes use real-time data from robots to make traffic decisions dynamically.

- **Layout**

The simulation layout represents an e-commerce warehouse, featuring a picking station, a replenishment station, and a storage area. The storage area contains pods, empty storage locations, and aisles for AMR movement. **Figure *3.3*** illustrates the simulation layout in NetLogo.

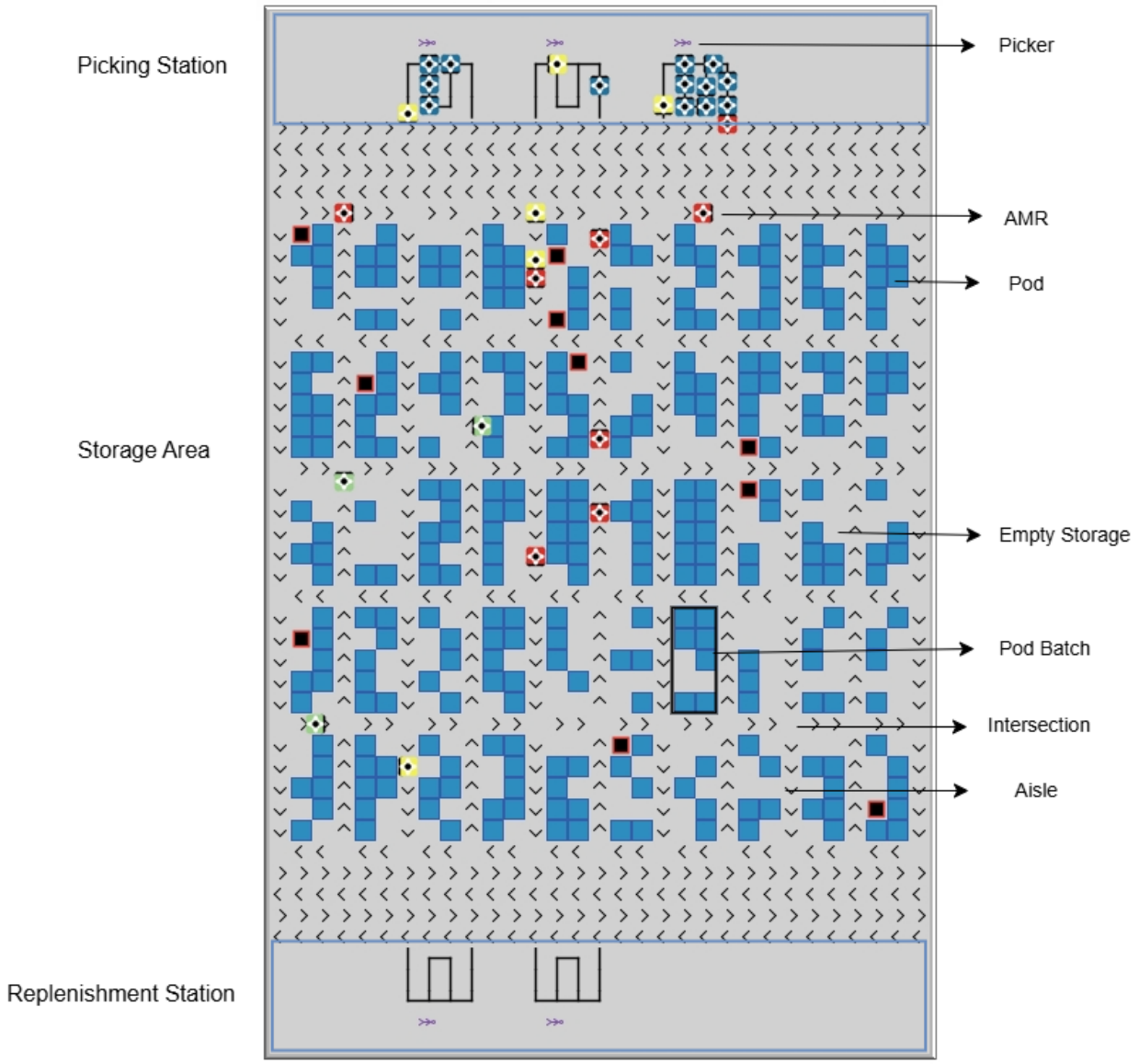

Figure 3.3 Simulation Layout

At the top of the layout are picking stations, each with a picker and a queuing track. Below the picking stations is the storage area, which houses pods containing various items. Pods with orders to be picked by AMRs are marked with different shapes. There are also empty storage locations where AMRs can place pods after picking.

The aisles are designated as one-way, but there are no directional constraints below the pods. AMRs in the aisles can either move forward or stop. At intersections, AMRs can turn based on the direction of the next aisle they need to navigate. Replenishment stations are at the bottom of the layout and have yet to be considered in the current simulation.

- **Operational Mechanism:**

Each equipped intersection's virtual traffic node communicates directly with approaching robots, processing data such as robot location, intended path, and operational urgency to dictate traffic flow dynamically. This method reduces the frequent stopping of robots, thereby saving energy.

- **Energy Efficiency Considerations:**

The core objective of implementing the virtual traffic light system is to minimize energy consumption across the RMFS. The strategic placement of Deep Q-Learning agents at high-impact intersections significantly reduces robot idle times and unnecessary stop-and-go movements, which are significant contributors to energy wastage.

- **Integration with RMFS:**

This system is seamlessly integrated with the overarching RMFS control system, which oversees all robot movements. This integration ensures that the traffic management system aligns with the warehouse's operational needs, facilitating a balanced and efficient workflow.

- **Simulation Environment:**

To validate the effectiveness of the virtual traffic light system, simulations are conducted in an environment that mimics the actual RMFS layout and operational dynamics. These simulations test the system under various traffic conditions to ensure reliability and robustness.

3.3 Reinforcement Learning Framework

This chapter outlines the reinforcement learning framework implemented to manage intersection control in an RMFS using Deep Q-Learning. The goal is to reduce energy consumption by optimizing the stop-and-go and waiting times of robots at intersections. This section details the state representation, action space, and reward function used in the Deep Q-Learning model.

3.3.1 Action Space

The action space defines the set of possible actions that the Deep Q-Learning agent can choose from to control the traffic lights at the intersection. Actions are 0: None or all directions (no restriction); 1: Allow vertical movement only, 2: Allow horizontal movement only.

3.3.2 State Representation

The state representation is crucial in defining the environment for the reinforcement learning agent. In this implementation, the state captures various attributes of the intersection and the robots present, and the state variable along with descriptions and possible values are shown in the following ***Table 3.1***:

Table 3.1 State Variables

State Variable	Description	Possible Values
allowed_direction_code	Current allowed direction at the intersection	0 (None), 1 (Vertical), 2 (Horizontal)
duration_since_last_change	Time since the last direction change (in ticks)	0 to ∞
delivering_pod_horizontal	Number of robots delivering pods to stations horizontally	0 to 5
returning_pod_horizontal	Number of robots returning pods to storage areas horizontally	0 to 5
taking_pod_horizontal	Number of robots taking pods horizontally	0 to 5
delivering_pod_vertical	Number of robots delivering pods to stations vertically	0 to 2
returning_pod_vertical	Number of robots returning pods to storage areas vertically	0 to 2
taking_pod_vertical	Number of robots taking pods vertically	0 to 2
connected_intersection_code	Allowed direction code of connected intersections	0 (None), 1 (Vertical), 2 (Horizontal)
connected_intersection_robots	Number of robots at connected intersections	0 to 2 (Vertical), 0 to 5 (Horizontal)

- **Allowed Direction Code:** This variable is fundamental as it determines the current traffic flow at the intersection. Knowing the allowed direction is crucial for the RL agent to decide when to change the traffic direction to optimize robot movement and reduce waiting times.
- **Duration Since Last Change:** This variable helps the RL agent avoid frequent changes in direction, which can lead to increased waiting times and energy consumption. By considering

the duration since the last change, the agent can make more informed decisions about when to switch directions, thus optimizing the flow.

- **Delivering, Returning, and Taking Pods Horizontally (Combined):** Combining these variables into a single state helps simplify the model while capturing the essential aspect of horizontal traffic. However, keeping them separate initially allows the agent to prioritize different actions (e.g., delivering might be more critical than returning or taking), enabling more nuanced decision-making.
- **Delivering, Returning, and Taking Pods Vertically (Combined):** Similar to the horizontal movement, combining these variables simplifies the model. It provides a clear understanding of vertical traffic while allowing the agent to consider the combined impact of different robot actions, helping to optimize the intersection flow.
- **Connected Intersection Code:** Understanding the directions allowed by connected intersections provides a broader context for the RL agent. It helps in coordinating traffic flow across multiple intersections, preventing bottlenecks and ensuring smoother transitions.
- **Connected Intersection Robots:** The traffic at connected intersections affects the current intersection. The agent can make more informed decisions to optimize the overall traffic flow by considering the number of robots at these intersections.

3.3.3 Reward Function

The reward function is responsible for computing the feedback for the agent based on the current state and actions of the robots at the intersection. This function integrates the rewards for passing robots and current robots to optimize energy consumption by minimizing waiting times and stop-and-go instances. **Calculate Reward for Passing Robot** will include the following, which are demonstrated in Algorithm 1:

- Initialize Reward: Start with a reward value of 0.
- Get State Multiplier: Retrieve a multiplier based on the robot's state, such as its priority or task.
- Calculate Previous Averages: Determine the previous average waiting time and stop-and-go instances for the direction in which the robot is moving.
- Track Robot Data: Update the intersection's data with the robot's information.
- Calculate Current Averages: Determine the new average waiting time and stop-and-go instances after including the robot's data.

- Reward Improvement: If the current average waiting time or stop-and-go instances are lower than the previous averages, increase the reward based on the improvement.
- Reward for Passing: Add a base reward for the robot successfully passing the intersection.
- Return Reward: Return the calculated reward for the passing robot.

Algorithm 1: Calculate Reward for Passing Robot

```
INITIALIZE reward TO 0
SET robot_state_multiplier TO get_state_multiplier(robot)
SET previous_average_wait TO intersection.calculate_average_waiting_time(direction)
CALL intersection.track_robot_intersection_data(robot, direction)
SET current_average_wait TO intersection.calculate_average_waiting_time(direction)
SET current_average_stop_n_go TO intersection.calculate_average_stop_n_go(direction)
IF current_average_wait < previous_average_wait:
    SET wait_diff TO previous_average_wait - current_average_wait
    ADD wait_time_reward * wait_diff * robot_state_multiplier * multiplier TO reward
IF current_average_stop_n_go < previous_average_stop_n_go:
    SET stop_go_diff TO previous_average_stop_n_go - current_average_stop_n_go
    ADD stop_n_go_reward * stop_go_diff * robot_state_multiplier * multiplier TO reward
ADD passing_robot_reward * robot_state_multiplier * multiplier TO reward
RETURN reward
```

Calculate Reward for Current Robot will include the following, in Algorithm 2:

- Initialize Reward: Start with a reward value of 0.

- Get State Multiplier: Retrieve a multiplier based on the robot's state, such as its priority or task.
- Calculate Waiting Time: Determine the total waiting time for the robot since it entered the intersection.
- Calculate Averages: Get the average waiting time and stop-and-go instances for the robot's direction of movement.
- Compare Waiting Time: If the robot's waiting time exceeds the average, decrease the reward based on the difference.
- Compare Stop-and-Go: If the robot's stop-and-go instances exceed the average, decrease the reward based on the difference.
- Return Reward: Return the calculated reward for the current robot

Algorithm 2 Calculate Reward for Current Robot

```
1  INITIALIZE reward TO 0
2  SET robot_state_multiplier TO get_state_multiplier(robot)
3  SET total_waiting_time_current_robot TO current_tick -
   robot.current_intersection_start_time
4  SET average_waiting_time TO intersection.calculate_average_waiting_time(direction)
5  SET total_stop_n_go_current_robot TO robot.current_intersection_stop_and_go
6  SET average_stop_n_go TO intersection.calculate_average_stop_n_go(direction)
7  IF total_waiting_time_current_robot > average_waiting_time:
8      SET wait_diff TO total_waiting_time_current_robot - average_waiting_time
9      ADD wait_time_penalty * wait_diff * robot_state_multiplier * multiplier TO reward
10 IF total_stop_n_go_current_robot > average_stop_n_go:
11     SET stop_go_diff TO total_stop_n_go_current_robot - average_stop_n_go
12     ADD stop_n_go_penalty * stop_go_diff * robot_state_multiplier * multiplier TO
       reward
13 RETURN reward
```

Handle No Robot at Intersection: In addition to calculating rewards for individual robots, the function also checks the overall state of the intersection. If the allowed direction is set (indicating the traffic light is open), but no robots are present at the intersection, a minor penalty

of -0.1 is applied. This discourages the agent from keeping the traffic light open unnecessarily, preventing potential infinite values in the reward calculation when no robots benefit from the open direction.

Algorithm 3 Main Reward Calculation Function

```
1  INITIALIZE reward TO 0
2  FOR each_robot IN intersection.previous_vertical_robots:
3      ADD calculate_reward_for_passing_robot(each_robot, intersection, "vertical", 2) TO
       reward
4  FOR each_robot IN intersection.previous_horizontal_robots:
5      ADD calculate_reward_for_passing_robot(each_robot, intersection, "horizontal", 1)
       TO reward
6  FOR each_robot IN intersection.vertical_robots.values()
7      ADD calculate_reward_for_current_robot(each_robot, intersection, "vertical", 2,
       tick) TO reward
8  FOR each_robot IN intersection.horizontal_robots.values():
9      ADD calculate_reward_for_current_robot(each_robot, intersection, "horizontal", 1,
       tick) TO reward
10 IF intersection.allowed_direction IS NOT None AND intersection.robot_count() == 0:
11     ADD open_intersection_penalty TO reward
12 RETURN reward
```

3.3.4 Energy Consumption Calculation

In this simulation, the Total Energy Consumption (TEC) of the RMFS robot is calculated by summing the energy used during various phases of the robot's movement and operations, which include acceleration, deceleration, constant speed travel, rotating, lifting, and lowering. This comprehensive approach ensures that all aspects of the robot's energy consumption are accounted for, providing a reliable measure of the system's efficiency.

The formula (1) shows the equation for TEC used in this simulation, which is:

$$\boldsymbol{TEC = E_{acc} + E_{dec} + E_{const} + E_{rot} + E_{lift} + E_{lower}} \quad \textbf{(1)}$$

In which:

- Energy during Acceleration (E_{acc}) is as follows, where m_A is the mass of the AMR, m_P is the mass of the pod, α^+ is the acceleration, k_{ir} is the inertia resistance coefficient, g is the gravitational constant, k_r is the rolling friction coefficient, v_{agv} is the average velocity, and t_{move} is the time spent moving.

$$\boldsymbol{E_{acc} = (m_A + m_P) \cdot (\alpha^+ \cdot k_{ir} + g \cdot k_r) \cdot v_{agv} \cdot t_{move}} \quad \textbf{(2)}$$

- The energy consumed during deceleration is similar to the energy during acceleration but with the direction of force being opposite. Energy during Deceleration (E_{dec}):

$$\boldsymbol{E_{dec} = (m_A + m_P) \cdot (\alpha^- \cdot k_{ir} - g \cdot k_r) \cdot v_{agv} \cdot t_{move}} \quad \textbf{(3)}$$

- Energy during Constant Speed (E_{const}) is as follows, in which v_{const} is the constant velocity:

$$\boldsymbol{E_{const} = (m_A + m_P) \cdot g \cdot k_r \cdot v_{const} \cdot t_{move}} \quad \textbf{(4)}$$

- Energy during Rotation (E_{rot}):

$$\boldsymbol{E_{rot} = E_{krot} + E_{frot}} \quad \textbf{(5)}$$

  - Kinetic energy during rotation (E_{krot}) shown below, where l is the length, w is the width, θ is the rotation angle, and t_{rot} is the time spent rotating:

$$\boldsymbol{E_{krot} = \frac{1}{6} \cdot (m_A + m_P) \cdot l^2 + w^2 \cdot \frac{\theta^2}{t_{rot}^2}} \quad \textbf{(6)}$$

- Frictional energy during rotation (E_{frot}), where r is the radius of rotation:

$$\boldsymbol{E_{frot} = (m_A + m_P) \cdot g \cdot k_r \cdot r \cdot \theta} \quad \textbf{(7)}$$

- Energy during Lifting (E_{lift}) is as follows, in which k_h is the deviation coefficient for height:

$$\boldsymbol{E_{lift} = m_P \cdot g \cdot k_h} \quad \textbf{(8)}$$

- Energy during Lowering (E_{lower}):

$$\boldsymbol{E_{lift} = m_P \cdot g \cdot k_h} \quad \textbf{(9)}$$

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experimental Design

4.1.1 Simulation Parameters

This section provides a detailed explanation of the key parameters used in the simulation. The parameters are crucial for ensuring the consistency and reliability of the simulation results.

Table 4.1 Simulation Parameters

Parameter	Value
Simulation	
Run Length	20 hours
Iteration	4 (Include Baseline)
Layout	
Inventory Area	500
Inventory capacity	300
Charging Port	10
Empty Storage Area	190
Pod Batch	2 x 5 blocks
Items	
SKU on pod	5
SKU variations	1000 type
Order	
Total	180
SKU frequency / popularity	Exponential distribution, $\lambda = 10$
Interarrival Times	Poisson distribution, $\lambda = 1$

Table 4.2 Simulation Parameters *(continue)*

Parameter	Value
AMR Movement	
Robot acceleration / deceleration	1 m/s^2
Robot max velocity	1.5 m/s^2
Full turning time	2.5 s
Lift / store pod time	3 s
Robots in the system	20
Replenishment	
Total Station	2 stations
Safety Stock	50% of total items on a pod
Time to replenish pod	20 s
Queuing per station	9 AMRs
Picking	
Total Station	3 stations
Order capacity	6 bins
Queueing per station	9 AMRs

4.1.2 Virtual Traffic Light Setup

Not all intersections within the RMFS are governed by the Deep Q-Learning-based traffic light system. To optimize the deployment of our resources and maximize impact, the Deep Q-Learning agents are selectively placed at intersections that experience high traffic volumes or are critical to the efficiency of the robot paths. Figure 4.1 shows these intersections.

The diagram highlights intersections equipped with virtual traffic light agents and illustrated the connectivity between these intersections. This visualization helps understand how traffic data from connected intersections influence decision-making processes, enhancing the system's overall efficiency.

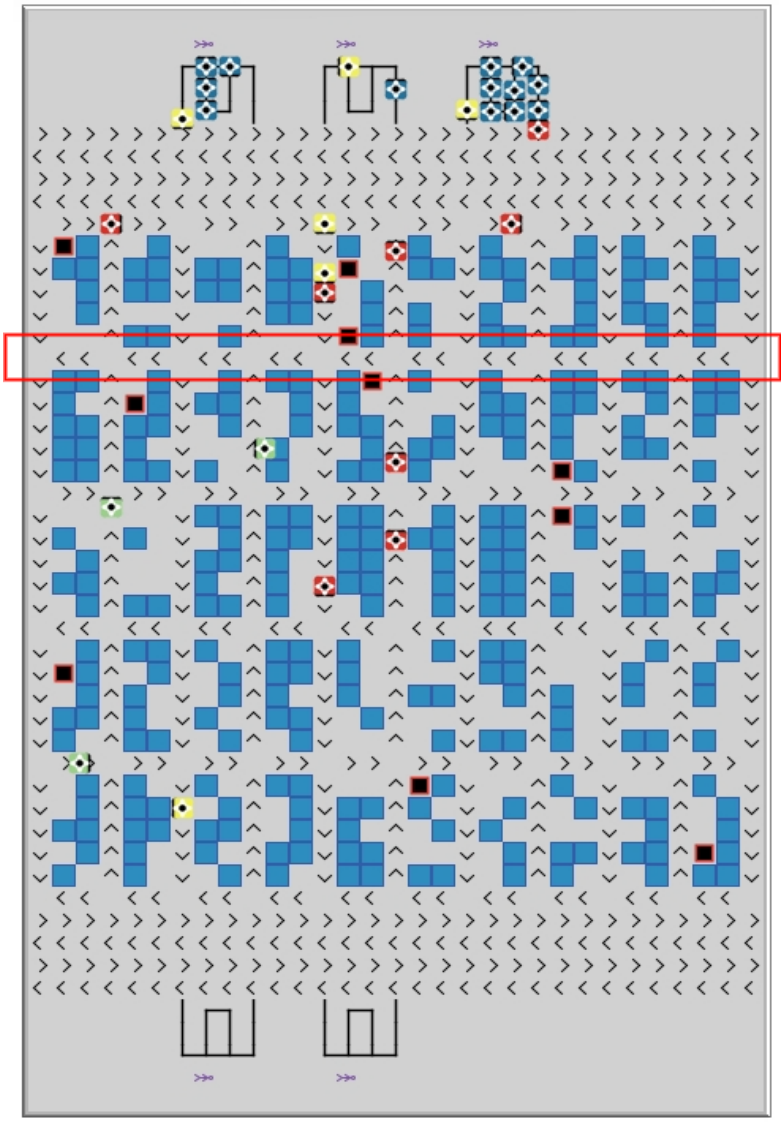

Figure 4.1 Intersections with Reinforcement Learning Agent

4.2 Results and Analysis

4.2.1 Baseline Scenario Result

In the baseline scenario, the RMFS operates without reinforcement learning (RL) intervention or traffic lights. Instead, a priority-based system is used to manage intersections. The priority rules are as follows:

- Robots carrying pods to the picking station have the highest priority.
- Robots returning pods to the storage area have the next highest priority.
- Robots without pods have the lowest priority

The simulation environment includes a specific number of robots, orders, SKUs, and pods as described in Table ***4.1*** and Table ***4.2***.

The performance of the RMFS in the baseline scenario is measured over a simulation period, focusing on key performance metrics, as shown in Table ***4.3*** below.

Table 4.3 Baseline Scenario Metrics

Metric	Baseline Scenario
Total energy consumption	228,977.27 J
Total waiting time at intersections	2,346.22 seconds

4.2.2 RL-Integrated Scenario Results

In the RL-integrated scenario, the RMFS employs a Deep Q-Learning algorithm to manage traffic lights at intersections. The RL model is trained to optimize traffic flow by dynamically adjusting traffic light timings based on real-time traffic conditions. The simulation environment and parameters remain consistent with those described in Chapter 4.1.

Table 4.4 Comparison of Matrix between Baseline and Deep Q-Learning

Metric	Baseline	Deep Q-Learning	Improvement
Total energy consumption	228,977.27 J	222,057.1 J	-3.0%
Total waiting time at intersections	2,346.22 seconds	3,713.4 seconds	+58.3%

Table 4.5 Comparative Analysis of RL Performance Across Different Scenarios

Metric	Scenario 1		Scenario 2		Scenario 3	
	Baseline	**Deep Q-Learning**	**Baseline**	**Deep Q-Learning**	**Baseline**	**Deep Q-Learning**
Total energy consumption	228,977.27 J	222,057.1 J	291,365.3 J	299,418 J	432,688.5 J	421,052.6 J
Total waiting time at intersections	2,346.22 seconds	3,713.4 seconds	2,934.4 seconds	4,127.2 seconds	4,438.95 seconds	5,014.5 seconds

Table 4.5 illustrates the comparative analysis of the RMFS performance under three different scenarios, each employing the Deep Q-Learning algorithm versus a baseline scenario.

- Total Energy Consumption: Across all scenarios, the RL algorithm shows varying impacts on energy consumption. In Scenario 1, the RL model slightly reduces energy consumption by approximately 3%. However, in Scenario 2, the energy consumption increases by around 2.77%, suggesting that the RL model may prioritize other metrics, such as traffic flow, in a medium-traffic environment. In Scenario 3, the RL model again reduces energy consumption by approximately 2.68%, demonstrating its effectiveness in handling high-traffic volumes efficiently.
- Total Waiting Time at Intersections: The waiting time at intersections increases significantly in all scenarios when using RL. This is particularly noticeable in Scenario 1, where the waiting time rises by about 58.3%. Similar trends are observed in Scenarios 2 and 3, where the waiting time increases by 40.65% and 12.99%, respectively. These results suggest that while the RL model is optimizing for energy consumption and other factors, it may do so at the cost of increased waiting times at intersections.

This analysis highlights the trade-offs involved in applying Deep Q-Learning in different traffic scenarios. The RL model appears to be more effective in environments with higher traffic volumes, where the marginal gains in energy efficiency are more pronounced. However, the increased waiting times indicate that further tuning of the model may be necessary to balance energy savings with intersection efficiency.

4.2.3 Training Iterations

The RL model was trained in three separate iterations to evaluate its performance and consistency. Each iteration used the same set of orders, SKUs, pods, and other parameters to ensure consistency and comparability. Each iteration aimed to run for a total of around 12,000 steps, with 1 second of simulation time equating to 4 steps. However, the simulation might stop earlier if no orders are being processed. This setup ensures that each training run is conducted under identical conditions, allowing for a precise analysis of the model's performance across different iterations.

Table 4.6 Training Iterations

Metric	Iteration 1	Iteration 2	Iteration 3
Total energy consumption	227,001.26 J	226,658.02 J	222,057.1 J
Total waiting time at intersections	2,451.78 seconds	2,354.21 seconds	3,713.4 seconds

Description of Training Iterations:

- ***Iteration 1 (0 to 12,000 steps):*** During the first iteration, the RL model focused on learning the basic navigation and intersection management strategies. The primary goal was to establish a baseline understanding of minimizing energy consumption while maximizing order throughput. The model encountered various intersection scenarios throughout these steps, gradually improving its decision-making process. The total energy consumption was recorded at 227,001.26 J, with a total waiting time at intersections of 2,451.78 seconds. Despite being the initial run, the model completed 180 orders.
- ***Iteration 2 (12,001 to 24,000 steps):*** Building on the knowledge gained from the first iteration, the second iteration aimed to refine the model's strategies for handling intersections and reducing waiting times. The model applied the insights from the previous steps to optimize its routes and decision-making processes further. This iteration

saw a slight decrease in total energy consumption to 226,658.02 J and a reduction in waiting time at intersections to 2,354.21 seconds. Consistently, the model managed to complete 180 orders, indicating its reliability and efficiency improvements over the previous iteration.

- ***Iteration 3 (24,001 to 36,000 steps):*** In the third and final iteration, the RL model focused on fine-tuning its strategies to achieve the lowest possible energy consumption and intersection waiting times. The model utilized the cumulative knowledge from the earlier iterations to make more informed decisions, resulting in a significant reduction in total energy consumption to 222,057.1 J. However, this iteration experienced an increase in total waiting time at intersections to 3,713.4 seconds, suggesting a trade-off between energy efficiency and waiting time. Notably, the model attempted to hold as many robots as possible at intersections, which is evidenced by the observation that up to 5 robots were queuing at a single intersection, a scenario not observed in Iteration 1 and 2. This behavior likely contributed to the increased waiting time, which was 58.3% higher than the baseline in Iteration 1. Despite this, the model consistently completed 180 orders, demonstrating its robustness and capacity to handle varying conditions.

Table 4.7 Queuing Robot per Iteration

Queuing Robot	Baseline	Iteration 1	Iteration 2	Iteration 3
0	541	546	535	528
1	1651	1636	1650	1616
2	94	100	97	122
3	1	5	5	15
4	0	0	0	5
5	0	0	0	1

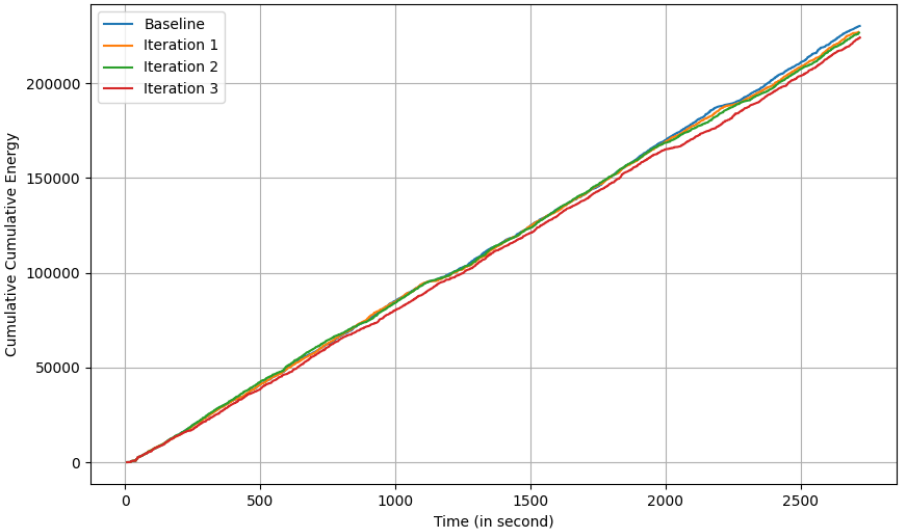

Figure 4.3 Cumulative Energy Consumption

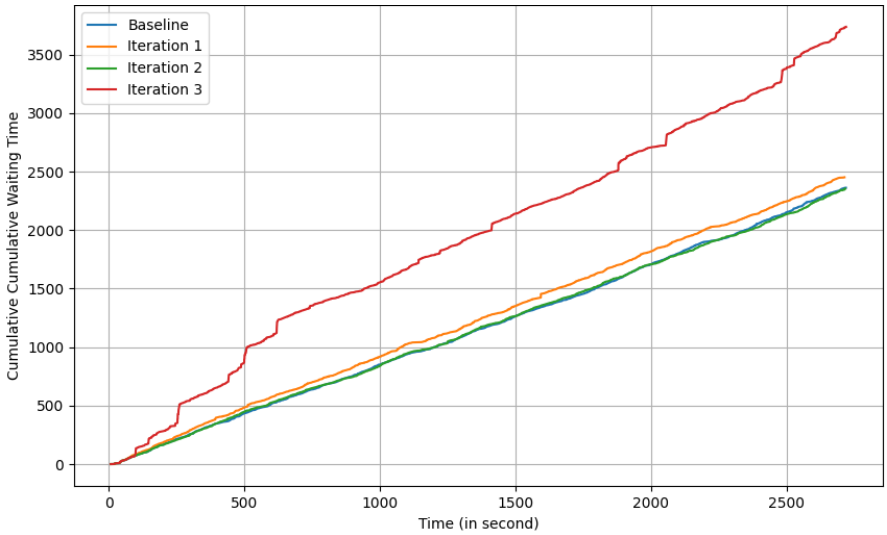

Figure 4.2 Cumulative Waiting Time

4.3 Statistical Analysis

In this section, we perform a detailed statistical analysis of the energy consumption and waiting times for both the baseline and RL-integrated scenarios. The data is paired or related because both scenarios use the same orders, SKUs, and other parameters. This allows us to use appropriate non-parametric tests to compare the two scenarios.

4.3.1 Normality Tests

To determine the appropriate statistical tests for comparing energy consumption and waiting times, we first assessed the normality of the data using visual and statistical methods. We used histograms and Q-Q plots to visually inspect the distribution of the energy consumption and waiting time data.

For Energy Consumption:

- Histograms: The histograms for both the baseline and Deep Q-Learning scenarios display the frequency distribution of the energy consumption values. Both histograms show a highly skewed distribution with a large number of observations clustering towards the high end of the energy consumption spectrum.

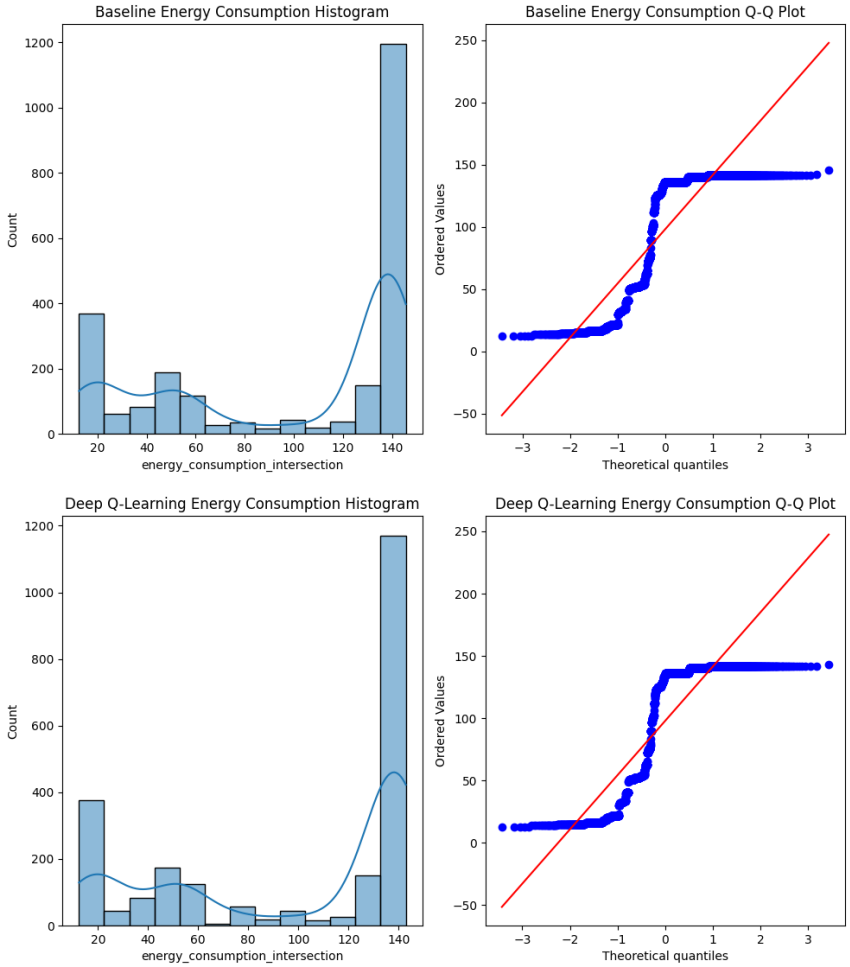

Figure 4.4 Histogram and Q-Q Plot for Energy Consumption

- Q-Q Plots: The Q-Q plots compare the quantiles of the energy consumption data with the theoretical quantiles of a normal distribution. Significant deviations from the red

reference line in both plots, especially at the tails, indicate that the data does not follow a normal distribution.

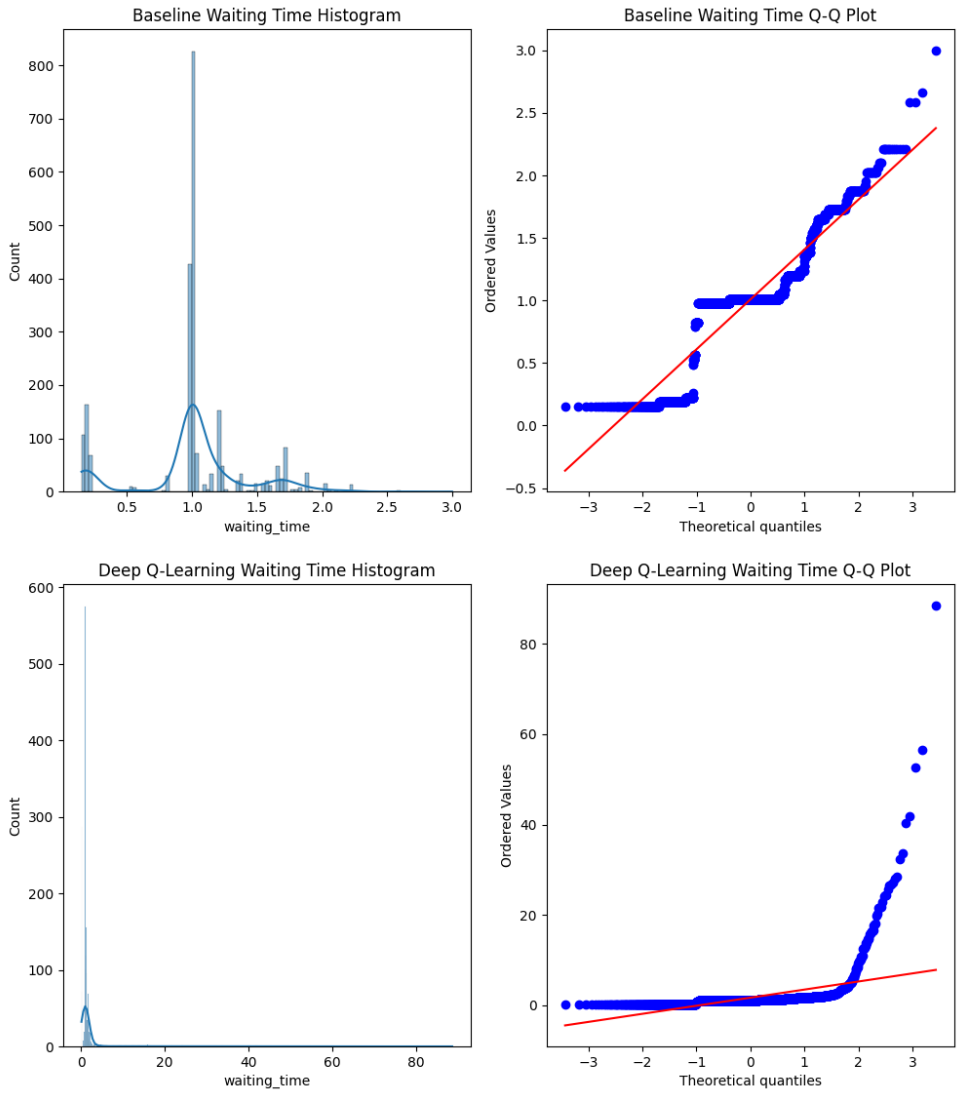

Figure 4.5 Histogram and Q-Q Plot for Waiting Time

For Waiting Time:

- Histograms: The histograms for the baseline and Deep Q-Learning scenarios show the frequency distribution of the waiting times. Both histograms indicate a skewed distribution, similar to the energy consumption data.
- Q-Q Plots: The Q-Q plots for waiting times also show significant deviations from the normal distribution reference line, particularly at the tails.

Given the visual indications of non-normality from the histograms and Q-Q plots, it is necessary to confirm these observations using statistical tests.

To statistically confirm the non-normality suggested by the visual inspections, we employed the Shapiro-Wilk test and Kolmogorov-Smirnov test. These tests provide a quantitative basis for determining whether the data deviates from a normal distribution.

Table 4.8 Energy Consumption Normality Test

Scenario	Shapiro-Wilk		Kolmogorov-Smirnov	
	Statistic	**P-Value**	**Statistic**	**P-Value**
Baseline	0.757	0.0	0.286	0.000
Deep Q-Learning	0.761	0.0	0.276	0.000

Table 4.9 Waiting Time Normality Test

Scenario	Shapiro-Wilk		Kolmogorov-Smirnov	
	Statistic	**P-Value**	**Statistic**	**P-Value**
Baseline	0.846	0.0	0.669	0.0
Deep q-learning	0.227	0.0	0.666	0.0

Interpretation of Results:

- Shapiro-Wilk Test: The p-values for both scenarios and both variables were 0.0, leading us to reject the null hypothesis of normality. This suggests that the energy consumption and waiting time data are not normally distributed.
- Kolmogorov-Smirnov Test: The p-values for both scenarios and both variables were 0.0, further confirming that the data significantly deviates from a normal distribution.

Based on the assumption of non-normality indicated by the visual inspections and confirmed by the Shapiro-Wilk and Kolmogorov-Smirnov tests, the Wilcoxon Signed-Rank test will be conducted to compare the energy consumption and waiting times between the baseline and RL-integrated scenarios.

4.3.2 Non-Parametric Tests

Given the non-normal distribution of the data, we use the Wilcoxon Signed-Rank Test to compare the energy consumption and waiting times between the baseline and RL-integrated scenarios.

Energy Consumption:

- The Wilcoxon Signed-Rank Test results for energy consumption indicate a W value of 981105.5 and a P-value of 0.657. This P-value indicates no statistically significant difference in energy consumption between the baseline and RL-integrated scenarios.
- The normality tests showed that the energy consumption data is not normally distributed for both scenarios. The Wilcoxon Signed-Rank Test indicated no significant difference in energy consumption between the baseline and RL-integrated scenarios

Waiting Time:

- The Wilcoxon Signed-Rank Test results for waiting time indicate a W-value of 823688.5 and a P-value of 0.0. This P-value indicates a statistically significant difference in waiting times between the baseline and RL-integrated scenarios.
- The normality tests indicated that the waiting time data is not normally distributed for both scenarios. The Wilcoxon Signed-Rank Test showed a significant difference in waiting times, suggesting that the RL integration has a considerable impact on intersection waiting times.

CHAPTER 5
CONCLUSION

5.1 Conclusion

This study explored the implementation of Deep Q-Learning (DQL) for managing virtual traffic lights at intersections within a Robotic Mobile Fulfillment System (RMFS). The primary aim was to optimize robot movement and reduce energy consumption by dynamically adjusting traffic light timings based on real-time traffic conditions. The experimental design included various parameters such as simulation run length, iteration count, layout specifications, and robot movement characteristics. The study provided a comprehensive approach to calculating Total Energy Consumption (TEC), accounting for phases of robot movement, including acceleration, deceleration, constant speed travel, rotation, lifting, and lowering.

The results indicated that the DQL-integrated system achieved a 3% reduction in total energy consumption compared to the baseline scenario, highlighting the potential of reinforcement learning in enhancing energy efficiency within RMFS environments. However, this improvement in energy consumption was accompanied by a significant increase in waiting times at intersections, which rose by 58.3%. Despite the increased waiting times, the total number of orders completed remained unchanged, demonstrating that the DQL system-maintained order throughput while optimizing energy usage. Essentially, the DQL system showed energy reduction without compromising order throughput.

The statistical analysis involved normality tests (Shapiro-Wilk and Kolmogorov-Smirnov) and non-parametric (Wilcoxon Signed-Rank Test) to compare energy consumption and waiting times between the baseline and DQL-integrated scenarios. The results confirmed the non-normal distribution of the data. Although the statistical tests did not show a significant difference in energy consumption between the two scenarios, the DQL system did contribute to energy savings. There was a significant difference in waiting times, emphasizing the impact of the DQL system on intersection management.

The study also examined the performance of the DQL model across three training iterations, each focusing on refining the model's strategies for handling intersections and minimizing energy consumption. The iterations demonstrated consistent order completion rates, underscoring the

robustness and reliability of the DQL approach in managing robot traffic within RMFS environments. It is important to note that all iterations used the same parameters to ensure consistency and comparability, providing a reliable basis for evaluating the DQL model's performance.

Overall, the study highlights the potential of using reinforcement learning techniques like DQL to optimize energy consumption in RMFS. However, it also points to further refinement to balance energy efficiency with operational efficiency, particularly in reducing waiting times at intersections. The results suggest that while the DQL system effectively reduces energy consumption, future efforts should focus on fine-tuning the model to mitigate the increase in waiting times without compromising order throughput.

5.2 Future Research

Future research should address the trade-offs identified in this study, particularly the increased waiting times at intersections. One potential area of exploration is integrating multi-agent reinforcement learning to optimize traffic flow further and reduce bottlenecks. Additionally, incorporating advanced prediction models to anticipate traffic patterns and dynamically adjust intersection priorities could enhance overall system efficiency. Further studies should also explore the scalability of the DQL approach in more extensive and complex RMFS environments, considering different robot densities and varying layout configurations. Another promising direction is the application of hybrid models combining DQL with other optimization techniques to balance energy consumption and waiting times more effectively. Tuning the DQL model further to find the optimal parameters that reduce waiting times while maintaining or improving energy efficiency is crucial. Lastly, the environmental impact of reduced energy consumption and its potential benefits in sustainable logistics should be investigated to provide a holistic view of the system's advantages.

REFERENCE LISTS

1. Wurman, P., R. D'Andrea, and M. Mountz, *Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses.* AI Magazine, 2008. **29**: p. 9-20.
2. Merschformann, M., et al., *Decision rules for robotic mobile fulfillment systems.* Operations Research Perspectives, 2019. **6**: p. 100128.
3. Chauhan, A., B. Brouwer, and E. Westra, *Robotics for a Quality-Driven Post-harvest Supply Chain.* Current Robotics Reports, 2022. **3**.
4. Guizzo, E., *Amazon Acquires Kiva Systems for $775 Million*. 2012, IEEE Spectrum.
5. Benavides-Robles, M.T., et al., *Robotic Mobile Fulfillment System: A Systematic Review.* IEEE Access, 2024. **12**: p. 16767-16782.
6. Kim, H.-J., C. Pais, and M. Shen, *Item Assignment Problem in a Robotic Mobile Fulfillment System.* IEEE Transactions on Automation Science and Engineering, 2020. **PP**: p. 1-14.
7. Keung, K.L., C.K.M. Lee, and P. Ji, *Data-driven order correlation pattern and storage location assignment in robotic mobile fulfillment and process automation system.* Advanced Engineering Informatics, 2021. **50**: p. 101369.
8. Li, X., et al., *Storage assignment policy with awareness of energy consumption in the Kiva mobile fulfilment system.* Transportation Research Part E: Logistics and Transportation Review, 2020. **144**: p. 102158.
9. Cechinel, A.K., et al., *Multi-robot Task Allocation Using Island Model Genetic Algorithm ⁎⁎This study was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.* IFAC-PapersOnLine, 2021. **54**(1): p. 558-563.
10. Meller, R. and J. Pazour, *A Heuristic for SKU Assignment and Allocation in an A-Frame System.* IIE Annual Conference and Expo 2008, 2008.
11. Xie, L., et al., *Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems.* European Journal of Operational Research, 2021. **288**(1): p. 80-97.
12. Yuan, R., H. Wang, and J. Li. *The Pod Assignment Model and Algorithm in Robotic Mobile Fulfillment Systems*. in *2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. 2019.
13. Allgor, R., T. Cezik, and D. Chen, *Algorithm for Robotic Picking in Amazon Fulfillment Centers Enables Humans and Robots to Work Together Effectively.* INFORMS Journal on Applied Analytics, 2023. **53**(4): p. 266-282.
14. Perumaal Subramanian, S. and S. Kumar Chandrasekar, *Simultaneous allocation and sequencing of orders for robotic mobile fulfillment system using reinforcement learning algorithm.* Expert Systems with Applications, 2024. **239**: p. 122262.
15. Stern, R., *Multi-agent path finding–an overview.* Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures, 2019: p. 96-115.
16. Hart, P.E., N.J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths.* IEEE transactions on Systems Science and Cybernetics, 1968. **4**(2): p. 100-107.

17. Standley, T. *Finding optimal solutions to cooperative pathfinding problems*. in *Proceedings of the AAAI conference on artificial intelligence*. 2010.
18. Goldenberg, M., et al., *Enhanced partial expansion A.* Journal of Artificial Intelligence Research, 2014. **50**: p. 141-187.
19. Wagner, G. and H. Choset, *Subdimensional expansion for multirobot path planning.* Artificial intelligence, 2015. **219**: p. 1-24.
20. Sharon, G., et al., *The increasing cost tree search for optimal multi-agent pathfinding.* Artificial intelligence, 2013. **195**: p. 470-495.
21. Sharon, G., et al., *Conflict-based search for optimal multi-agent pathfinding.* Artificial intelligence, 2015. **219**: p. 40-66.
22. Li, J., et al. *Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search*. in *IJCAI*. 2019.
23. Ma, Z., Y. Luo, and H. Ma. *Distributed Heuristic Multi-Agent Path Finding with Communication*. in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.
24. Chen, Y.F., et al. *Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning*. in *2017 IEEE international conference on robotics and automation (ICRA)*. 2017. IEEE.
25. Long, P., et al. *Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning*. in *2018 IEEE international conference on robotics and automation (ICRA)*. 2018. IEEE.
26. Sartoretti, G., et al., *Primal: Pathfinding via reinforcement and imitation multi-agent learning.* IEEE Robotics and Automation Letters, 2019. **4**(3): p. 2378-2385.
27. Liu, Z., et al. *Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments*. in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. IEEE.
28. Grossman, D.D., *Traffic control of multiple robot vehicles.* IEEE Journal on Robotics and Automation, 1988. **4**(5): p. 491-497.
29. Bourbakis, N.G., *A traffic priority language for collision-free navigation of autonomous mobile robots in dynamic environments.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1997. **27**(4): p. 573-587.
30. Wang, J. and S. Premvuti. *Fully Distributed Traffic Regulation and Control for Multiple Autonomous Mobile Robots Operating in Discrete Space*. in *Distributed Autonomous Robotic Systems*. 1994. Tokyo: Springer Japan.
31. Dunne, M.C. and R.B. Potts, *Algorithm for traffic control.* Operations Research, 1964. **12**(6): p. 870-881.
32. Ross, D., R. Sandys, and J. Schlaefli, *A computer control scheme for critical-intersection control in an urban network.* Transportation Science, 1971. **5**(2): p. 141-160.
33. Jin, J. and X. Ma, *Adaptive Group-based Signal Control by Reinforcement Learning.* Transportation Research Procedia, 2015. **10**: p. 207-216.
34. Jin, J. and X. Ma, *A group-based traffic signal control with adaptive learning ability.* Engineering Applications of Artificial Intelligence, 2017. **65**: p. 282-293.
35. Teja, V.A., D.V.K. Viswanath, and K.M. Krishna. *A mixed autonomy coordination methodology for multi-robotic traffic control*. in *2008 IEEE International Conference on Robotics and Biomimetics*. 2009.

36. Sutton, R.S. and A.G. Barto, *Reinforcement learning: An introduction.* Robotica, 1999. **17**(2): p. 229-235.
37. Sutton, R.S. and A.G. Barto, *Reinforcement learning: An introduction*. 2018: MIT press.
38. Zheng, G., et al. *Learning phase competition for traffic signal control*. in *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019.
39. Wang, T., J. Cao, and A. Hussain, *Adaptive Traffic Signal Control for large-scale scenario with Cooperative Group-based Multi-agent reinforcement learning.* Transportation research part C: emerging technologies, 2021. **125**: p. 103046.
40. Vidali, A., et al. *A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management*. in *Woa*. 2019.
41. Liu, D. and L. Li, *A traffic light control method based on multi-agent deep reinforcement learning algorithm.* Scientific Reports, 2023. **13**(1): p. 9396.
42. Jiang, Q., et al., *Multi-agent reinforcement learning for traffic signal control through universal communication method.* arXiv preprint arXiv:2204.12190, 2022.
43. Chen, C., et al. *Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.
44. Wiering, M.A. *Multi-agent reinforcement learning for traffic light control*. in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 2000.
45. Watkins, C.J.C.H., *Learning from delayed rewards.* 1989.
46. Watkins, C.J.C.H. and P. Dayan, *Q-learning.* Machine Learning, 1992. **8**(3): p. 279-292.
47. Chang, F., et al., *Charging Control of an Electric Vehicle Battery Based on Reinforcement Learning*. 2019. 1-63.
48. Clifton, J. and E. Laber, *Q-learning: Theory and applications.* Annual Review of Statistics and Its Application, 2020. **7**: p. 279-301.
49. Wilensky, U., *NetLogo*. 1999, Center for Connected Learning and Computer-Based Modeling, Northwestern University: Evanston, IL.
50. Pedregosa, F., et al., *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011. **12**: p. 2825-2830.