Robotic Warehouses for E-Commerce: Evaluation, Operation, and Design

By

Yiduo Huang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2024

Robotic Warehouses for E-Commerce: Evaluation, Operation, and Design

Abstract

Robotic Warehouses for E-Commerce: Evaluation, Operation, and Design

by

Yiduo Huang

Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Zuo-Jun Shen, Chair

As e-commerce expands, warehouse systems face new challenges, leading to the development of robotic warehouses. These warehouses typically employ part-to-picker systems, where mobile robots and stationary human workers collaborate. To improve the performance of these systems, we proposed new performance evaluation models and operational strategies. We also proposed innovative designs aimed at fully robotic warehouses in the foreseeable future.

Performance prediction and evaluation are crucial in designing and operating robotic warehouses, especially given the highly stochastic and complex nature of robot traffic congestion. We introduced a new evaluation model that accounts for robot congestion by initially modeling the system as a closed queueing network (CQN) with blocking. Simulation observations led us to propose a congestion mechanism and simplify the system to a CQN without blocking. Demonstrating the asymptotic Poisson properties of robot arrivals enabled us to further approximate the simplified CQN as a transportation network. This approach allowed us to estimate congestion delays as closed-form functions of traffic flow. We integrated these estimated delays with the CQN model and developed an iterative algorithm to estimate the system throughput. Our numerical experiments confirmed that this method could accurately predict throughput under transportation congestion when the system was stable.

Effective real-time robotic warehouse operation requires strategic decisions regarding workstation assignments and collision-free robot path planning. To improve system efficiency, we developed an integrated method for task assignment and path planning, implemented in both offline and online phases. In the offline phase, based on our evaluation model, we estimated an approximated optimal steady-state traffic assignment, while the online phase guided robots according to offline traffic patterns using a decentralized and computationally efficient algorithm. The simulation results indicated that our method achieved 5-10% higher throughput and required much less computational time compared to current industrial implementations.

The advent of robotic arms has made fully robotic warehouses feasible. We proposed a new layout design that positions workstations, called internal workstations, equipped with robotic arms within the storage area to minimize transportation costs. We introduced a batching pool mechanism using special pods to collect and transport assembled totes in batches from internal to external workstations. This system was evaluated using an open queueing network (OQN), which led to a closed-form queue delay approximation. We found an upper bound for the relative error in the sojourn time estimation using this approximation and showed that our approximation is accurate. Using this approximation, we developed a location-allocation-queuing model, which can be transformed into mixed integer second-order conic programming (MISOCP) for efficient solving, to find the optimal workstation locations and pod-to-workstation allocation plans. This model demonstrated a significant reduction in transportation costs and an improvement in the robot machine time of 10-20% for large or deep systems in our simulations.

To my mother and father.

Contents

Contents **ii**

List of Figures **iv**

List of Tables **vi**

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 System Description . . . 2
- 1.3 Motivation and Objectives . . . 6
- 1.4 Structure of the Dissertation . . . 12

2 Literature Review **13**
- 2.1 A Review on Robotic Warehouse Evaluation . . . 13
- 2.2 A Review on Robotic Warehouse Operation . . . 14
- 2.3 A Review on Robotic Warehouse Design . . . 16

3 Evaluation of Robotic Warehouses **20**
- 3.1 Introduction . . . 20
- 3.2 System Description and Assumptions . . . 20
- 3.3 Approximation of the Steady-State . . . 21
- 3.4 Throughput Estimation . . . 32
- 3.5 Numerical Results . . . 35
- 3.6 Summary . . . 43

4 Operation of Robotic Warehouses **44**
- 4.1 Introduction . . . 44
- 4.2 The Offline Algorithm: Finding an Approximated Optimal Network Flow . . . 45
- 4.3 The Offline-Online Algorithm . . . 51
- 4.4 Discussions . . . 53
- 4.5 Numerical Results . . . 55
- 4.6 Summary . . . 60

5 New Design for Robotic Warehouses **62**
5.1 Introduction 62
5.2 System Design 62
5.3 Model 65
5.4 Solution Method 70
5.5 Numerical Results 73
5.6 Summary 80

6 Conclusions **81**

Bibliography **83**

A The Asymptotic Poisson Property **90**
A.1 Proof of Theorem 1 90
A.2 Simulation Study for Conjecture 1 90

B Proof of Theorems for Warehouse Design **94**
B.1 Proof of Proposition 3 94
B.2 Proof of Theorem 2 95

List of Figures

1.1 RSS robot and drop-off point . . . 3
1.2 Drop-off point [64] . . . 3
1.3 RSS workstation [18] . . . 4
1.4 RSS . . . 5
1.5 RMFS robot with a pod [25] . . . 6
1.6 RMFS grid-cell . . . 7
1.7 RMFS layout . . . 8

3.1 System . . . 22
3.2 Closed queuing network models . . . 23
3.3 Approximation of the closed queuing system . . . 25
3.4 Open queuing network models . . . 30
3.5 A buffer cell example . . . 31
3.6 CQN embedding the traffic delay . . . 33
3.7 Simulation setting: congestion mechanism . . . 36
3.8 Congestion in a grid-based AMR system . . . 37
3.9 Flow on different cells every 1000 timesteps . . . 38
3.10 Accuracy of the objective function . . . 38
3.11 Different layout used in throughput validation experiments . . . 40
3.12 Throughput validation results . . . 41

4.1 RSS model . . . 46
4.2 Cell node decomposition . . . 48
4.3 Flow chart of the offline-online algorithm . . . 52
4.4 Large system map . . . 55
4.5 Accuracy of the objective function . . . 56
4.6 Throughput for different settings . . . 57

5.1 KIVA-like RMFS systems . . . 63
5.2 New RMFS design . . . 64
5.3 OQN for fully robotic RMFS . . . 66
5.4 Validate the error bound . . . 74
5.5 Layout of scenario 1 . . . 76

5.6 Layout of scenario 6 77
5.7 Workstations location with respect to varying $\mathbb{E}[S_2]$ 78

A.1 CV of inter-arrival times vs E[G] where G is normally distributed with different CV_X 91
A.2 CV of inter-arrival times vs E[G] where G is uniformly distributed with different CV_X 91
A.3 CV of inter-arrival times vs E[G] where G is normally distributed with different ρ 92
A.4 CV of inter-arrival times vs E[G] where G is uniformly distributed with different ρ 92

List of Tables

3.1 Layout of throughput estimation validation experiment 41

4.1 Error, gap and improvement bound 56
4.2 Performance improvement 58
4.3 Computation time per simulation step in second 59

5.1 Scenarios of numerical experiments 75
5.2 Cost comparison between our new design and KIVA system 76
5.3 Operational cost comparison with respect to different demand densities 79

A.1 Simultion estimated CV for different G distributions with $\rho = 0.9$ 93

Acknowledgments

I would also like to extend my deepest gratitude to Professor Zuo-Jun Shen for his invaluable guidance and support throughout my research and the journey of completing my Ph.D. dissertation. His expertise and insightful feedback have been pivotal in shaping my academic endeavor. I am truly grateful for his mentorship and the opportunity to learn from such an esteemed academic figure. Thank you, Professor Shen.

I wish to thank Professor Bipan Zou at Zhongnan University of Economics and Law, who helped me with the numerical experiments in the design problems and improved my writing in this chapter. I also wish to thank Xiguang Wei and Geek Plus International Co., Ltd. Their industrial experience and data support are essential for completing this work. Thanks to committee members Professor Michael Cassidy and Assistant Professor Zeyu Zheng for their guidance and insight.

Chapter 1

Introduction

1.1 Background

A warehouse is a large building where goods are stored before they are distributed for sale or further processing. As a crucial node in the supply chain, a warehouse provides a central location for receiving, holding, and dispatching products. Warehouses are used by manufacturers, importers, exporters, wholesalers, transport companies, customs, and other entities involved in the movement of goods.

E-commerce, or electronic commerce, involves buying and selling goods or services using the Internet and transferring money and data to execute these transactions. The business-to-consumer (B2C) segment of e-commerce has seen a rapid increase in sales volume [53]). In 2023, e-commerce sales were estimated at \$ 1118.7 billion, accounting for 15.4% of total retail sales [56].

E-commerce heavily relies on a vast network of warehouses, especially fulfillment and sorting facilities. Platform-owned sellers on Amazon use fulfillment centers to manage their extensive inventory and ensure efficient product delivery to customers. An Amazon fulfillment center is a large warehouse facility where Amazon stores, picks, packs, and ships products sold on its platform. Their products are stored in these centers until an order is placed. The ordered item is then located, retrieved, packed, and shipped to the customer. In addition to platform operations, individual third-party sellers, which make up more than 60% of total sales on Amazon [52], often rely on postal and express companies like FedEx [26] and their sorting facilities. Unlike a fulfillment center, a sorting center does not keep inventory, but serves as a facility where packages are received, sorted, and dispatched to their final destinations.

The growth of e-commerce platforms such as Amazon and JD.com has introduced several challenges to traditional order fulfillment and sorting in warehouses, necessitating significant adaptations and innovations in warehouse systems. First, these platforms have increased order processing demands, dealing with various products, often in smaller quantities. Warehouses now must handle a broader range of items and process more individual orders (an average of 1.6 lines per order [13]) compared to the bulk shipments typical of traditional retail. Second, with the rise of same-day and next-day delivery options, e-commerce has created an expectation of rapid order fulfillment. This pressures warehouses to pick, sort, and ship products much faster. Third, e-commerce often experiences significant demand fluctuations, especially during holiday seasons or special sales events such as Black Friday, requiring warehouses to quickly and cost-efficiently scale up throughput capacity to handle these peaks.

Traditional warehouse systems struggle to meet the needs of today's online shopping businesses. Typically, workers get a list of items from a depot and then collect them on different shelves. This method is particularly inefficient for small orders, as workers spend considerable time walking between the depot and the shelves, which does not directly contribute to order completion. This inefficiency can lead to fewer orders being processed unless changes are made, such as hiring more workers or grouping several orders, making it challenging to meet the fast delivery expectations of online shopping.

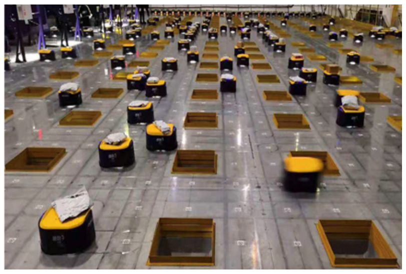
To address these challenges in order fulfillment, Amazon developed a robotic mobile fulfillment system (RMFS) called KIVA in 2012 [6]. This technology fundamentally transformed the operational dynamics of Amazon fulfillment centers. Unlike traditional warehouses where human workers travel throughout the warehouse to collect items, the KIVA robot system uses a fleet of mobile robots that bring shelves or "pods" containing necessary items directly to workers stationed at packing stations. This goods-to-person operation mode drastically reduces the time and physical strain involved in retrieving goods, thus increasing efficiency and productivity in order fulfillment. Unlike traditional automated warehouses that use complex conveyor belts requiring significant investments, the KIVA system is designed to be scalable, allowing Amazon to handle increasing package volumes, especially during peak periods like the holiday season, by simply adding more robots.

Inspired by the success of KIVA, mobile robots have also been implemented in sorting operations, called Robotic Sorting Systems (RSS) [76]. In these systems, robots navigate a grid-based roadmap, transporting parcels between loading stations and designated drop-off points. Each robot is loaded with a single parcel at one of the various workstations and transports it to a drop-off point that matches the parcel's delivery destination. Upon delivery, the robot deposits the parcel into a receptacle connected to the drop-off point and then returns to the queue for the next task. Notable implementations of RSS include Tompkins Robotics' tSort [55], Amazon's Xanthus and Pegasus robots [7], and Deppon Express [64]. To better understand the functioning of these systems, interested readers may view video demonstrations at `https://www.youtube.com/watch?v=4MH7LSLK8Dk` or `https://www.youtube.com/watch?v=EbLDXsEPHS8`.

1.2 System Description

The dissertation analyzes two types of robotic warehouses for e-commerce: RMFS and RSS. This section introduces how these systems work.

RSS


(a) Drop-off point [64]

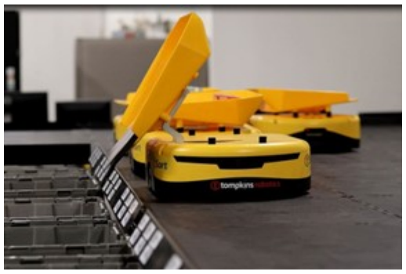

(b) RSS robot [64]

Figure 1.1: RSS robot and drop-off point

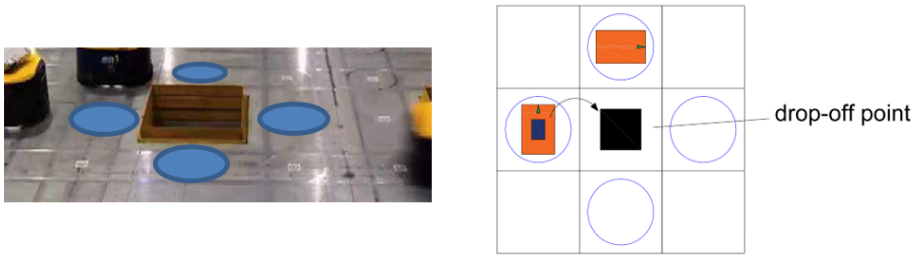


Figure 1.2: Drop-off point [64]

RSSs are sorting systems where robots transport parcels from loading workstations to drop-off points to sort parcels. The layout of the RSS studied in this paper is depicted in Figure 1.4. This system is designed to handle parcels labeled for various delivery destinations and transport them to corresponding drop-off points using mobile robots. Amazon [7] and Deppon Express [64] use systems with similar layouts. The robot moving area comprises non-overlapping square cells, allowing for two basic movements: transitioning from one cell to another and making a 90-degree turn. Moving between cells is assumed to take T_1 and turning takes T_2, with $T_2 > T_1$. In RSSs, as illustrated in Figure 1.1b, robots can carry or release one parcel at a time using a tilt tray.

Some cells are designated as drop-off points, each featuring a hole leading to a roll container on the lower floor, as shown in Figure 1.2. Each drop-off point, represented by solid black or red squares in Figure 1.4, corresponds to a specific delivery destination. Robots can approach the adjacent cell of a drop-off point to deliver a parcel using the tilt tray, but are prohibited from entering the drop-off cells. The time to drop a parcel is denoted as T_{drop}.

Designated workstation areas are represented by blue dashed rectangles in Figure 1.4, where human workers load empty robots. The parcel loading time is defined as T_{load}. Each

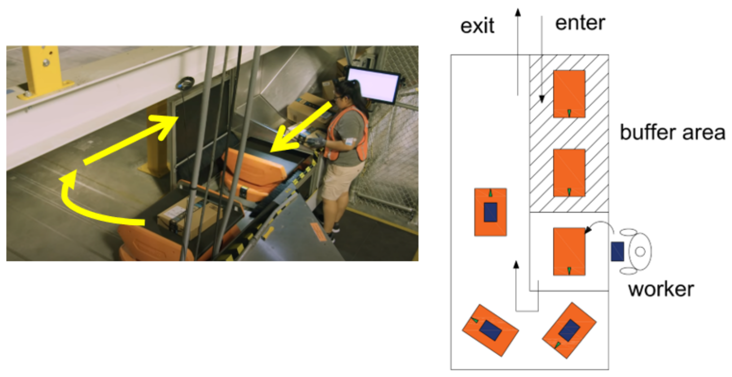


Figure 1.3: RSS workstation [18]

workstation includes an entrance, an exit, and a buffer area, as shown in Figure 1.3. Robots follow the yellow arrow depicted to enter the workstation, wait in the queue to load, and exit after being loaded by a worker.

In addition to the cell grid, a sorter system integrates conveyor belts along the perimeter of the facility, connecting the workstations to the parcel entrance, as detailed in Figure 1.4. On arrival, the new packages are transported by the sorter system to an assigned workstation.

The movement trajectory of a robot is outlined in four steps (indicated by dashed arrows in Figure 1.4):

1. loading at a workstation;

2. transporting the parcel to the designated drop-off point;

3. unloading the parcel at the adjacent cell of the drop-off point;

4. returning to a workstation to await the next parcel.

Cell traffic is unidirectional; for instance, if robots are allowed to move from south to north on the cell i, then movements from north to south on the same cell are prohibited, as noted in the ”allowed direction” in Figure 1.4. This unidirectional design, common in real-world RSSs and described in [7, 55, 64], helps prevent robot gridlocks.

RMFS

An RMFS is a part-to-worker fulfillment system in which robots move pods with items from the storage area to picking workstations, and human workers at workstations pick ordered items and assemble totes to be packed and shipped. The layout of the RMFS studied in this paper is illustrated in Figures 1.6 and 1.7. Similar to RSS, the robot moving area comprises non-overlapping square cells, as depicted in Figure 1.6. The workstations are strategically located along the periphery of the facility and the pods are stored in designated storage

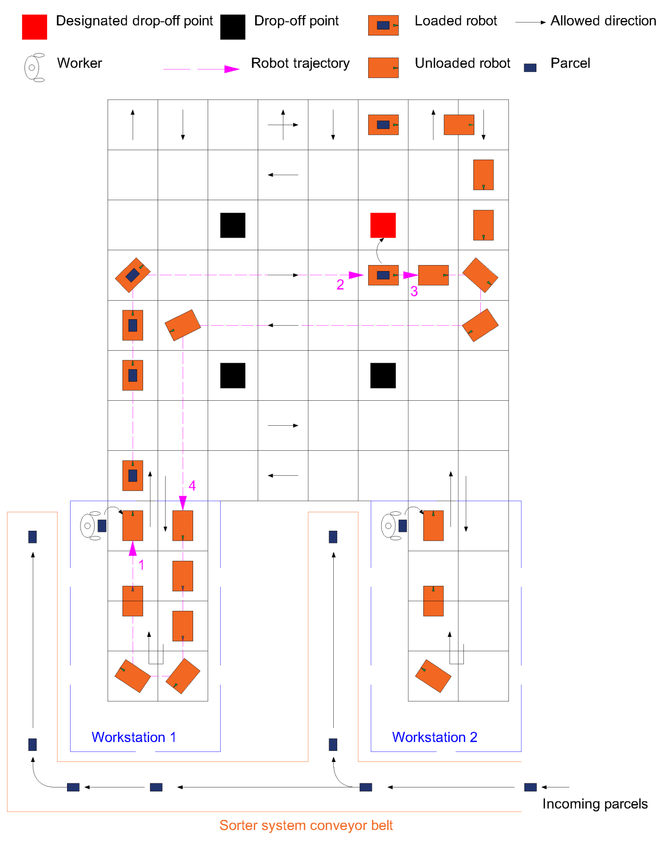


Figure 1.4: RSS

areas. Upon receiving a new order, robots transport pods containing all requested items to a workstation. A human worker picks the items from the pods and places them into a tote. The robots in the RMFS move the pod by lifting the pod, as shown in Figure 1.5.

The pods are organized into blocks within the storage area, each occupying one cell and allowing enough clearance for the unloaded robots to pass underneath. To load a pod, a robot approaches it from one side, locks onto it, lifts it using its loading mechanism, and then transports it to a workstation, as illustrated on the right part of Figure 1.7. The loaded robots are restricted to aisles between the pod blocks as they cannot travel directly under other pods.

At workstations, robots with pods queue according to the first-come, first-served (FCFS) principle. Each human worker at the workstation retrieves items from the nearest loaded pod corresponding to a fulfillment order and places these items into a tote. These assembled totes are then transferred to conveyors for rapid shipment to other facilities for packing and consolidation, as depicted in Figure 1.7. The placement of workstations on the periphery of

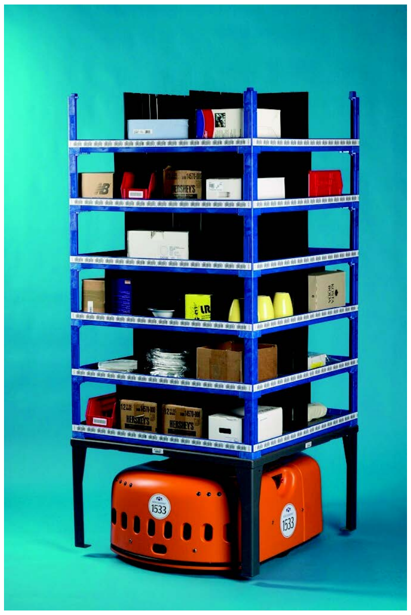


Figure 1.5: RMFS robot with a pod [25]

the facility facilitates easy access to these conveyors.

The movement trajectory of a robot in the RMFS can be summarized in three steps (indicated by arrows in Figure 1.7):

1. moving to a pod;
2. lifting the pod and transporting it to a picker at a workstation;
3. waiting at the workstation for item retrieval and then returning the pod to the storage area.

1.3 Motivation and Objectives

The transition from conventional to robotic sorting and fulfillment systems presents several distinct advantages. First, robotic systems provide greater layout and operational flexibility. Workstations, drop-off points, and storage pods can be strategically distributed across the warehouse, eliminating the need for fixed conveyor systems. This modularity enables rapid adaptability to fluctuating demand patterns. Second, robotic systems produce substantial cost savings, with capital investments that are 40 to 50 percent lower than those required for traditional tilt tray or cross belt systems [55]. In addition, these systems significantly reduce the need for manual labor, offering further cost savings.

E-commerce companies strive to maximize efficiency in cost, throughput, space utilization, and service level within their robotic warehouses. Challenges arise on two distinct levels:

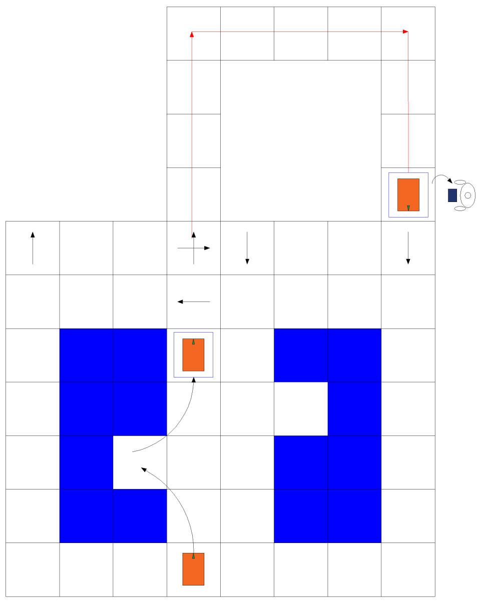


Figure 1.6: RMFS grid-cell

strategic and operational. At the strategic level, decisions related to system design, such as determining optimal locations for workstations, storage areas, and drop-off points. These decisions crucially influence the long-term performance of the system. At the operational level, the focus shifts to task assignment, route planning, and scheduling for robots and human workers, which substantially affects system efficiency.

Performance evaluation plays a crucial role in robotic warehouse design and operation. It involves analyzing the warehouse's efficiency and provides insights into the effectiveness of a particular design or operational strategy relative to established benchmarks, identifying areas for improvement. Furthermore, an accurate performance evaluation framework enables designers to efficiently assess various design options, facilitating the refinement of the design space at the preliminary stages of the design process. Mathematically, a performance evaluator is an objective function in system design and operational problems.

Consequently, constructing and managing a proficient robotic warehouse necessitates:

- A fast and accurate evaluation model: Developing an evaluation model capable of forecasting the performance of the robotic warehouse system.
- Efficient operation strategy: Formulating operational strategies aimed at improving robot efficiency.
- New designs: Innovating system designs and the corresponding design algorithms.

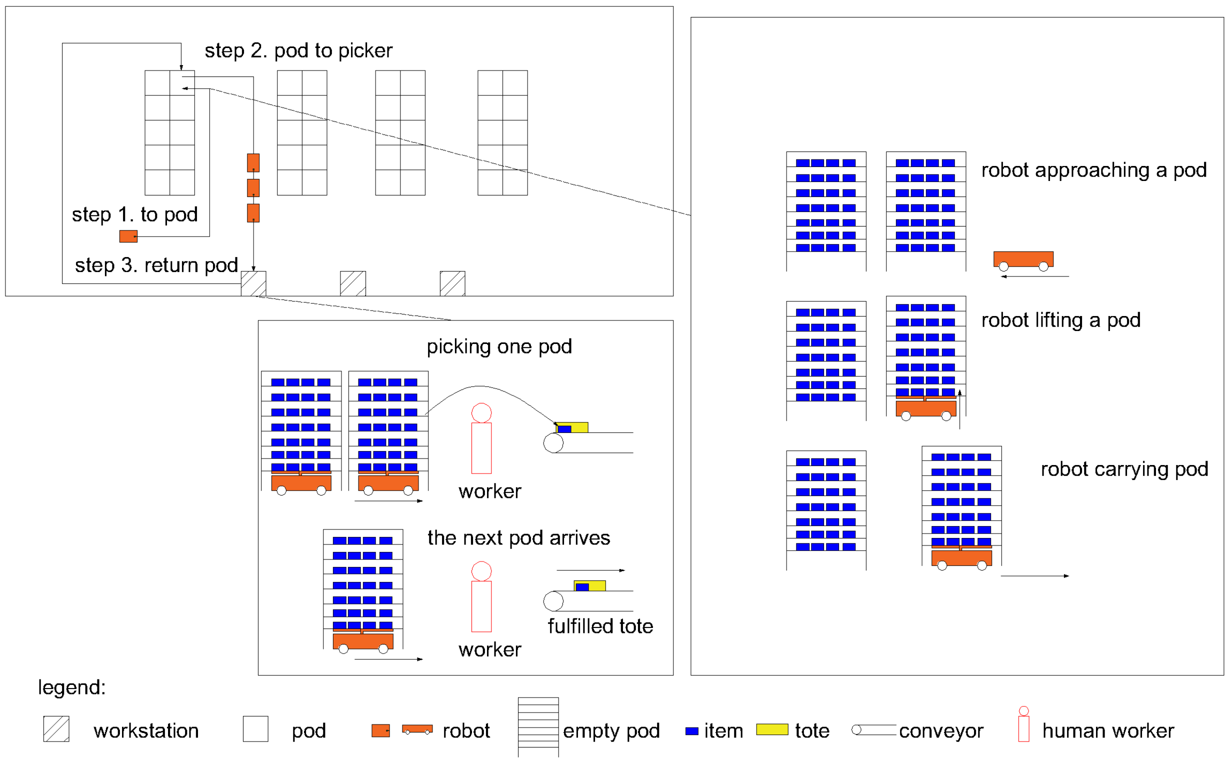


Figure 1.7: RMFS layout

Evaluation

Despite the widespread deployment of robot-based systems in recent years, a theoretical evaluation model for general congestion remains elusive. Typically, when order demand increases, managers can respond by enlarging the robot fleet and adding more workstations. However, an expanded fleet size could exacerbate congestion and delays, particularly when the system approaches saturation. This underscores the need for a precise evaluation model that considers the layout and performance metrics of each workstation and robot.

Numerical simulations are commonly utilized to assess system performance in industrial applications for robotic warehouses. However, the extensive computation time required for these simulations presents a challenge when used as an objective function in operational and design optimization. Therefore, there is a pressing need for a theoretical model that can estimate the robot machine time and system throughput. A closed-form theoretical evaluation model would streamline design optimization and provide crucial insights into the system's operational dynamics.

A significant challenge in modeling flexible robotic warehouses, especially Robotic Sorting Systems (RSS), is accounting for congestion and blocking effects. Robotic warehouses typically feature a grid-based roadmap with hundreds of cells, where vehicle movements are not restricted to specific aisles or corridors. Congestion occurs when multiple robots compete for the same cell along their routes. Unlike automated guided vehicle (AGV) container terminals, each cell in an RSS can act as an intersection and is a potential conflict point, as queues can develop from any direction within the system. Consequently, a mid-sized system could have hundreds or even thousands of potential conflict points, complicating the resolution of congestion issues. Due to this complexity, most researchers have omitted congestion from their theoretical system evaluation models.

This dissertation aims to provide a theoretical performance evaluation for robotic warehouses, considering traffic congestion and blocking effects without relying on time-consuming simulations. This model will also serve as a foundation for addressing operational and design challenges in the field. Given the system layout and robot parameters, we will build a model to estimate the robot congestion delay and the system throughput. We will model the system as closed queuing network (CQN) with blocking and then approximate it using a traffic flow network. Congestion can be estimated as the link delay function. We will develop an iterative algorithm that combines CQN and link delay to estimate throughput.

Operation

Controllers must address two key operational challenges in an RSS: *assignment* and *path-planning*. The assignment involves two subproblems: (1) parcel-to-workstation assignment and (2) robot-to-workstation assignment, commonly referred to as *dispatching* in the literature [27]. Upon the arrival of a new parcel or order, it must be assigned to an appropriate workstation; similarly, a robot returning from a drop-off point/ pod storage area requires reassignment to a workstation. Additionally, the controller must plan a collision-free path for the robot, consisting of a sequence of grid cells from its current location to its intended destination.

Multi-robot path-finding is a well-explored area with two primary types of algorithms: centralized and decentralized. In a centralized system, a controller monitors all robots in real time and assigns spatial-temporal paths to each, ideally preventing any stops if robot speeds are precisely controlled. However, implementing a centralized algorithm in large systems is impractical for two reasons. First, finding an optimal solution for general multi-robot path planning is NP-hard [31], and many heuristics suffer from the curse of dimensionality. Second, no robot controller is flawless, and control errors tend to accumulate, which means precalculated optimal plans can easily be disrupted. Therefore, maintaining a high communication frequency among all robots is essential, a task that becomes unfeasible in large systems, even with advanced communication technologies.

In contrast, a decentralized controller requires a careful design to prevent increased congestion. Allowing robots to individually decide their paths without cooperation may lead to competition for bottleneck areas, resulting in gridlocks. Robots should cooperate, not compete, to improve performance. In addition, cooperation between robots and other system components is crucial. Task assignments that disregard path planning can lead to extended travel distances and weaving traffic patterns between different parts of the system.

In this study, our aim is to develop a method that simultaneously plans and assigns workstations for robots and parcels in a robotic warehouse. The pathfinding algorithm must be fast, decentralized, and capable of handling a system with hundreds of robots. Path-planning and task-assignment decisions must be made online and must cooperate with each other and all other parts of the system. Using the total transportation cost from our evaluaiton model as the objective, we can minimize the total cost by finding the optimal steady-state traffic flow assignment offline. Given the optimal flow assignment, we can plan the paths and assign tasks to robots according to the path-flow split ratio in online operation.

Design

Integrating new technologies into established systems can significantly improve performance. For instance, incorporating mobile robots into fulfillment systems has led to innovative designs like Amazon's KIVA, which has transformed the landscape of e-commerce warehouses. Current systems use human workers to pick up or load items because robotic arms, which can accurately identify the desired items and pick them, are too expensive for now. With the advent of advanced and cheaper robotic arms, it is now feasible to envision robotic warehouses operating entirely without human labor, using cheaper and more advanced robotic arms, prompting further innovations in system design.

In KIVA-like systems, workstations are traditionally placed in the periphery to facilitate access to conveyors and maintain separation between human employees and robots. With the potential replacement of human labor by robotic arms, there is an opportunity to reposition workstations into the warehouse's interior, significantly reducing travel distances. This result is supported by simulations in [62]. However, internal workstations are isolated within the storage area and lack access to system input/output conveyors, necessitating a new design to realize a fully robotic warehouse.

This shift to a new design with interior workstations introduces challenges, such as determining the optimal placement for these workstations and assessing the possible performance improvement through this new layout. Taking into account the queuing delay in transportation time, this dissertation aims to develop a location-allocation-queuing model that identifies optimal workstation locations in a fully automated robotic warehouse and compares the results with those of KIVA-like systems.

In this study, our objective is to provide a new design for fully automated robotic warehouses to reduce robot transportation costs and to find the optimal location and type of workstations inside the warehouse based on our location-allocation-queueing model. Our idea is to put robotic arm-equipped workstations inside storage areas (interal workstations) to reduce the transportation cost. We will use special pods as pools for the assembled totes at the internal workstation. Once a special pod is filled, it will be transported to one of the external workstations to be shipped. To find the optimal locations of the internal and external workstations, we model the system as an open-queueing network and build a mixed-integer programming (MIP) model. The MIP can be approximated as a second-order conic programming (SOCP) and solved efficiently. For large-scale problems, we will develop Lagrangian Relaxation procedures with SOCP subproblems.

Summary of Objectives

In summary, the primary objectives of this dissertation were:

1. To develop a theoretical performance evaluation model for robotic warehouses that accounts for traffic congestion and blocking effects. The input is the system layout and robot parameters, and the output is the estimated traffic delay and system throughput.

2. To develop an integrated path-finding and task-assignment strategy for robotic warehouses, optimizing both system performance and computational efficiency. The input is the system layout. We will first run the offline algorithm to find the optimal steady-state traffic assignment. Then we can operate the system while assigning paths and tasks to robots online.

3. To create a new design for robotic warehouses using advanced technologies coupled with an optimization model to identify the optimal design in this innovative setting. The new design will deploy internal workstations to pick up items and assemble totes inside storage areas. The assembled totes are stored in special pods and will be sent to external workstations once the speical pods are full. The assembled totes will be batched at internal workstations, so the total transportation cost can be greatly reduced. For the new design and the KIVA design, we will also develop a location-allocation-queueing model to find the optimal workstation location design. The input is the system map, the candidate locations of the workstation, the budget of the workstation, and the robot parameters. The output can find the optimal workstation locations that can minimize the total transportation cost.

Additionally, the theoretical contributions of this research to the grid-based robot transportation system can be summarized as follows:

- A closed-form estimator for robot travel and waiting times for robotic warehouses, considering the system layout, fleet parameters, and traffic flow distribution.

- Proof of the asymptotic Poisson properties of interarrival times in light traffic cells within a grid-based mobile robot warehouse, with a numerical demonstration of a similar result for heavy-traffic cells, albeit with a non-strict proof.

- Application of a traffic flow network to approximate a large closed queueing network with complex blocking scenarios.

- Development of an approximation method for location problems that incorporate queueing congestion, complete with a proven error bound.

- A reformulation of the location-allocation-queue problem using second-order conic programming.

1.4 Structure of the Dissertation

This dissertation is organized as follows: In Chapter 2, we reviewed the related literature on the evaluation, operation, and design of robotic warehouses. Chapter 3 introduced our evaluation model for robotic warehouses. Chapter 4 presented our integrated assignment and path-finding operation strategy. Chapter 5 described our new warehouse design and the solution to the location problem within this new design. Chapter 6 summarized the findings.

Chapter 2

Literature Review

2.1 A Review on Robotic Warehouse Evaluation

Robotic warehouses, including Robotic Mobile Fulfillment Systems (RMFS) and Robotic Sorting Systems (RSS), have been extensively studied in recent years. Boysen et al. [12] and Azadeh et al. [9] have conducted notable reviews on robotized warehouses and similar systems.

Before the advent of RMFS and RSS, most robotic warehouses operated as autonomous vehicle storage and retrieval systems (AVS/RS). In these systems, robots functioned as vehicles within a traditional Automated Storage and Retrieval System (AS/RS). AVS/RS can be segmented into different tiers, with autonomous vehicles moving within one tier or transitioning to another via a lift. Compared to RMFS, AVS/RS lacks flexibility, as vehicle movement is limited to physical aisles. Theoretical evaluation methods for AVS/RS were based mainly on queueing networks, including open queueing networks (OQN) [30] and semi-open networks (SOQN) [24] [15]. The straightforward layout of the AVS/RS facilities, with rule-based routing assumptions, allowed each aisle/rack section to be modeled as a server with finite capacity. The researchers evaluated the blocking effect using capacitated queueing networks [9][45].

RMFS features a more flexible parts-to-picker system, where a robot retrieves a pod from the storage area and transports it to a picker workstation. After the picking process, the robot returns the pod to the storage area. Theoretical evaluation models for RMFS were typically based on SOQN or closed queueing network (CQN). SOQN for RMFS was initially developed by Lamballais, Roy, and De Koster [35]. In their model, a robot was synchronized with the order and picked up the order at a workstation. The transportation delay was modeled as an infinite capacity delay node, where travel times were the shortest path travel times without blocking. This SOQN model was extended to evaluate various operating policies, including inventory allocation [36], assignment rules [75], battery changing [74], and zone assignment strategies [47]. In addition to queueing network models, simulation-based studies such as those of Merschformann and Lamballais et al. [40] also provided design insights for

RMFS. However, most theoretical evaluation models of RMFS overlooked the blocking effect and transportation congestion due to the multitude of potential conflict points, making it challenging to represent RMFS with a blocking queueing network accurately.

RSS shares a structure similar to that of RMFS. Instead of retrieving items from storage pods, robots in an RSS pick items at workstations and drop them at designated drop-off points without the need for an "inventory" or "storage area". All the RMFS evaluation models mentioned above can be applied to RSS with minor modifications. Zou et al. [76] developed a queueing network model for RSS that considers the congestion effect. They modeled the blocking effect at drop-off points using queue nodes representing blocked vehicles in their CQN and developed an iterative procedure to estimate the CQN with updated blocking delays at each iteration. In addition to blocking at drop-off points, congestion delays elsewhere were calculated using traffic flow delay functions derived from simulation data based on BPR-like flow-delay relations [42]. The functional form of their delay function, based on the Underwood model [57], was adopted from [46]. Xu et al. [64] studied the optimization of parcel-to-workstation assignment using an OQN to evaluate RSS performance.

The article by Zou et al. [76] is the most closely related to our evaluation model, as we also examined RSS traffic congestion and blocking effects. However, their study focused solely on developing a theoretical model for blocking during the unloading process, estimating en-route traffic congestion delays using empirical data-based formulas derived from simulations. In real-world systems, these two types of congestion (near the drop-off point and on route) operate under similar mechanisms and should be addressed within the same theoretical framework. Our research developed a theoretical delay-flow model capable of theoretically explaining both types of congestion. Our approach can also be readily extended to other systems with different functions, such as RMFS and AGV-based airport baggage handling systems [27], while their method is specific to parcel handling processes and is limited to solving RSS-related problems.

2.2 A Review on Robotic Warehouse Operation

Effective operation of a robotic warehouse system requires addressing two fundamental issues: robot/order assignment and path planning. Although there are many studies, the prevailing literature often investigates these problems separately.

Assignment Problem in Robotic Warehouse Systems

There were two main research streams on the assignment problem for robotic warehouses. One stream focused on steady-state analysis, utilizing queueing networks to evaluate order and robot assignment strategies or to construct their objective functions. For example, Lamballais et al. [35] developed a semi-open queueing network and proposed a zone-based rack-to-storage assignment strategy for an RMFS, which they extended to include order replenishment and inventory management [36]. They found that spreading stock-keeping units (SKUs) among racks would improve throughput. Roy et al. [47] compared different robot-to-workstation assignment rules using closed queueing networks. They found that a pooled robot system outperforms a dedicated robot strategy, and assigning robots to the least congested zone will improve the performance of multi-zone systems. For RSS, Zou et al. [76] developed a CQN model to predict system throughput. Xu et al. [64] optimized parcel-to-workstation assignment as an integer programming problem, with the throughput estimated using an open queueing network. Queuing network-based approaches provided simple, closed-form solutions to evaluate system performance, helping to identify optimal assignment strategies. However, these methodologies often rely on assumptions of steady-state conditions and memoryless arrival processes, which may not align with the complexities of real-world systems. Our research follows this stream, using assumptions that may not hold for all system layouts. We will introduce an indicator to help managers decide on the applicability of our method to their systems without the need for a preliminary implementation.

Another research stream addressed transient-state systems with finite time horizons, formulating the assignment problem as mixed-integer programming (MIP) with binary variables indicating specific assignments. Boysen et al. [11] optimized robot visit sequences and order fulfillment to minimize total robot visits in an RMFS. Weidinger et al. [60] optimized rack-to-storage positions to minimize travel distances, proposing "shortest-path storage" assignment rules comparable to optimal MIP solutions. Wang et al. [59] examined rack-to-workstation assignment under uncertainties in human picker performance using stochastic dynamic programming. Unlike queueing-based models, MIP-based models accommodated dynamic and nonsteady-state systems with fewer assumptions but suffered from the curse of dimensionality, making them impractical for large-scale industrial applications.

One of the similar problem settings to our RSS assignment problem can be found in [64]. We share a similar roadmap and model parameters. However, they only considered the assignment rule of robots to workstations for RSS, while our approach is for integrated assignment and path-planning, and our model can also be extended to other problems like RMFS.

Multi-Agent Path Finding (MAPF)

Numerous studies on MAPF for mobile robots or automated guided vehicles (AGVs) have been conducted (for reviews, see [8, 65, 43, 20]). Cao's review [16] classified approaches into centralized and decentralized. Centralized methods control all robots in real time and can find optimal collision-free solutions in small-scale systems. However, path planning for multiple robots is NP-hard [69], and even with heuristics, it becomes computationally unfeasible in real-time for large systems. Decentralized methods, which scale well with system size, allow each robot to resolve conflicts using local information [see [66, 71, 16]]. While faster and suitable for real-time large-system use, they may not avoid gridlocks and often yield suboptimal solutions.

A new method used traffic flow to help avoid congested areas, combining centralized and decentralized approaches: the upper-level flow problem was centralized to optimize system performance, while path planning was decentralized for fast, real-time computation. Fransen et al. [27] and Digani et al. [23, 22, 21] have developed hierarchical strategies for multi-AGV systems, optimizing sector-level paths, and coordinating robots within sectors. Similarly, our research adopted a flow-based strategy, centralizing the offline flow problem and decentralizing the online algorithm, but our model formulation differed significantly from these previous works.

2.3 A Review on Robotic Warehouse Design

The research problem investigated in the new design of the RMFS falls within the scope of the location allocation queueing problem [1], which involved decisions on the location and capacity of the facilities and the assignment of customers/demands to these facilities. The objective was to minimize the total cost, which included the costs of locating facilities, establishing capacity, and the travel and waiting costs for customers accessing services. This section reviews two streams of literature: studies on robotic mobile fulfillment systems (RMFS) and those on the location-allocation-queuing problem.

Studies on RMFS Layout

Introduced by Amazon in 2012 [41], the RMFS has since been rapidly adopted by numerous e-commerce and logistics companies, attributed to the efficiency and cost benefits of its innovative parts-to-picker operation mode. The academic community has extensively explored the design and operational policy analysis of this system. Initial studies, such as the pioneering work by Lamballais et al. [35], focused on the location of workstations within RMFS. They constructed a semi-open queueing network (SOQN) to estimate performance and found that the placement of the workstation significantly affects the system throughput. Workstations located on the longer side of a nonzoned storage area enhance throughput capacity more effectively than those on the shorter side. On the contrary, Wu et al. [62] explored various layout scenarios, including moving the pick station into the storage area, and demonstrated the superiority of internal over external stations using a similar SOQN approach. Furthermore, Azadeh et al. [9] addressed the design optimization problem by determining the optimal system shape and the locations of the workstation. Li et al. [37] utilized a simulation platform to evaluate RMFS performance and the impact of different task fulfillment processes, positioning workstations equally spaced or facing the aisles at the bottom of the system. Yang et al. [67] developed an integer programming model to minimize the total travel distance of robots, investigating near-optimal workstation location patterns in traditional and flying-V layouts.

Further studies within RMFS have focused on the assignment of orders, robots, and racks to workstations. Yuan et al. [70] developed an open queueing network for system performance estimation using dedicated and shared robot-to-workstation assignment rules. Zou et al. [75] designed a neighborhood search algorithm to optimize robot workstation assignments.

Boysen et al. [11] studied batching and sequencing of orders at picking stations, employing a simulated annealing-based heuristic to solve these decision problems. Xie et al. [63] optimized rack and order sequencing in RMFSs, showing that integrating these decisions and allowing for order splitting can increase system throughput by up to 46%. The joint optimization of these processes was further explored by Yang et al. [68] and Zhuang et al. [73], the latter also considered workload balance and rack conflicts among multiple workstations.

Unlike previous studies, Wang et al. [59] and Sheu et al. [50] investigated the impact of human factors on RMFS operating decisions. Specifically, Wang et al. [59] examined the fluctuation of the picking state of human workers in the assignment of racks to pickers. They developed a stochastic dynamic programming model to address this problem and designed an approximate dynamic programming-based branch-and-price method to solve the model. The results showed that picking time can be reduced by about 10% compared to solutions that do not consider schedule-induced fluctuations in the picking states. Sheu et al. [50] studied the assignment of racks to picking stations, considering the cumulative fatigue of human workers. They developed a discrete-time nonlinear dynamic stochastic model to address this problem. A real case study shows that the application of the proposed robot-picker coordination system can reduce the accumulated fatigue of the picker by 54% at the expense of reducing the efficiency of the picker by 15%. Zhen et al. [72] considered a combination optimization problem that involves the batching of orders, the sequencing of orders and batches to pods (movable racks that store SKUs) and workstations, and the assignment of robots to jobs. They formulated the model for such a comprehensive problem and designed a two-layer revolving algorithm for solution. Kumar et al. [34] investigated the effect of perceived workload of human workers on system performance using an SOQN. Their result showed that the workload-dependent service rate significantly affected system performance, suggesting that increasing robots may not always be beneficial. For a comprehensive review of RMFS studies, we refer to [10].

The above studies on RMFS implicitly assumed workstations located on the perimeter of the system, except for [62], for two reasons. First, safety regulations prevent human workers from walking in the middle area of the system, where a collision with a robot is possible. Second, a conveyor is normally placed adjacent to the workstation to facilitate the transfer of picked orders outside the system for further packaging and delivery. This constraint can be relaxed in RMFS using workstations with robotic arms and swarm robots. On the one hand, the robotic arm can replace the human worker [54], eliminating the risk of accidents with robots and allowing them to be located in the middle of the system. On the other hand, different types of robots can jointly perform tasks [49], with one part transporting the pods to the workstation to pick up orders and the other part transferring the shelves with picked orders outside the system.

Studies on the Location-Allocation-Queuing Problems

A location-allocation-queuing problem involves allocating several facilities among candidate locations, determining their service capacity, and assigning customers/demands to these facilities. The objective of this problem is generally to minimize the total costs, including the costs of setting up the facilities (fixed and capacity-related) and the service costs (travel and waiting). This problem has numerous applications in various contexts, such as the location of emergency facilities, public service facilities, and manufacturing facilities.

Early studies focused on the formulation of this problem. For example, Agnihotri et al. [4] explored the problem of assigning customers to service facilities to minimize the total cost of accessing facilities and waiting for service. They constructed an optimization model and developed a Lagrangian relaxation-based heuristic solution procedure to solve the model. Marianov et al. [38] extended the set covering problems to locate the minimum number of facilities and servers needed to provide service within a limited waiting time for customers. They designed a heuristic concentration method to solve the model. The above two studies considered facility-related costs [38] or customer-related costs [4]. Abolian et al. [1] addressed both in the location-allocation queuing problem, designing a special-purpose algorithm to find the optimal solution for minor problems that iteratively optimizes the location-allocation and assignment decisions, respectively. A simulated annealing heuristic was also developed.

Subsequently, several variations of the location-allocation queuing problem are investigated. Abolian et al. [2] additionally considered the elastic demand with respect to distance in the location-allocation queueing problem with no fixed coverage radius of facilities. They took profit maximization as the objective function. An exact algorithm was designed to solve small problems that successively improve the upper and lower bounds. An ascending heuristic was proposed for large-size problems. Castillo et al. [17] studied two scenarios for the setting of capacity of the location-allocation queuing problem. One used a single server with varying service rates at each facility, and the other adjusted the number of servers (with fixed capacity). The optimal service rates in two scenarios were explicitly derived, resulting in a tractable optimization model. A Lagrangian relaxation approach was used to solve the models. Rahmati et al. [44] constructed the location-allocation queuing problem as a multiobjective model, where the first objective was to minimize the total setup cost and capacity cost of the system, and the second objective minimized the total expected travel time and waiting time of customers. Multi-objective evolutionary algorithms were used to solve the model. Abolian et al. [3] considered a public facility location-allocation queuing problem to maximize the number of people accessing the services. A two-stage solution method was designed to solve the model, where the first stage optimized the customer allocation with the given facility location and server capacity, and the second stage optimized the controlled decisions in the first stage. The nonlinear part of the objective function was transformed by a piecewise linear approximation with maximum relative error ϵ, the so-called ϵ-optimal algorithm. [48] studied an idle vehicle repositioning problem, which implements the location-allocation queuing problem. A Lagrangian relaxation approach was designed to solve the model and a simulation experiment was conducted to validate the performance of the proposed algorithm.

Some studies specialized in solving the location-allocation queueing problem model. Vidyarthi et al. [58] focused on solving the location-allocation queueing problem. They transformed the non-linear waiting cost expression into a linear form, with the expense of adding additional variables and constraints. The linearized model was solved by an iterative algorithm that updates the upper and lower bounds of the objective function until the gap converges. Ahmadi et al. [5] modeled the location-allocation queueing problem as different mixed-integer second-order cone programs and investigated their performance.

Chapter 3

Evaluation of Robotic Warehouses

3.1 Introduction

Our study aimed to provide a theoretical performance evaluation for RSS and RMFS, considering traffic congestion and blocking effects without running time-consuming simulations. Given a robotic warehouse layout and the robot fleet parameters, our aim was to estimate the average robot traffic delay and system throughput (the number of parcels/orders that can be processed in one hour).

We analyzed the congestion in robotic warehouses and divided them into congested areas and non-congested cells, then represented the warehouse as a closed queuing system. Subsequently, we proposed heavy-traffic approximations for congested areas and light-traffic approximations for non-congested cells. Under our assumptions, we proved that the arrival at each non-congested cell is asymptotic Poisson and showed that the inter-arrival time coefficient of variance (CV) at congested areas converges to 1. Using these approximations, cell delays could be approximated as the waiting time of an M/G/1 queue for congested areas and as the residual service time of an alternating renewal process for non-congested cells. To evaluate RSSs' throughput, we created a closed queueing network (CQN) that embeds the traffic flow network. We developed an algorithm that solves the CQN and iteratively estimates the delay in the traffic flow network.

We introduced how we approximated the complicated system and found a closed-form solution for traffic delay in Section 3.3. We developed an algorithm to predict system throughput in Section 3.4. These results were numerically validated using simulation in Section 3.5. We summarized the findings in Section 3.6 and provided some ideas for future research.

3.2 System Description and Assumptions

We developed our model based on RSS in Section 1.2. This evaluation model can also be extended to RMFS.

Let $\{W_1, \ldots, W_{n_W}\}$ be the set of workstations, $\{D_1, \ldots, D_{n_D}\}$ the set of drop-off points, and $\{C_1, \ldots, C_N\}$ the set of cells that are neither workstation cells nor drop-off points. In the example in Figure 3.1a, we have two workstations $n_W = 2$ and 4 drop-off points $n_D = 4$. There are R robots circulating in the system. Let the traffic flow from cell C_i to C_j be $v_{i,j}$. $v_{i,j}$ is defined as the average number of robots traveling from C_i to C_j in one time step under the steady state.

Assume that enough parcels are waiting to be sorted so that the system runs at its maximum capacity. Usually, packages that arrive the previous day will be stored and sorted the next day. Therefore, the robots will be immediately assigned to a new parcel and a new workstation after releasing the previous parcel; that is, there will be no idle robot. R robots circulate in the system to deliver parcels from workstations to drop-off points.

After running for a while, we assume that the system can reach a steady state. We also assume that cells are first come, first served (FCFS): When two robots from C_i and C_j, respectively, want to enter C_k, the robots that arrived first will be allowed entry into C_k, and the other robot will stop and wait until C_k is clear.

A robot can either (1) get loaded at one of the workstations, move to one of the drop-off points, and drop the parcel or (2) go back to one of the workstations after dropping the parcel. We define the traffic of loaded robots from workstations to drop-off points as *forward flow*, and the traffic of empty robots from the drop-off points to the workstations as *backward flow*.

3.3 Approximation of the Steady-State

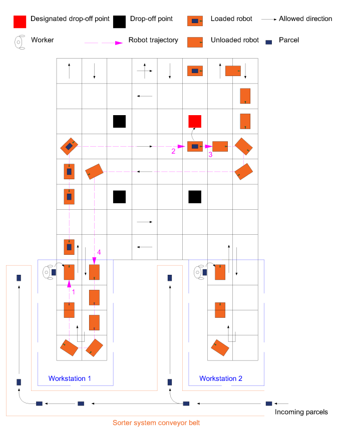
This section will discuss four models, each of which is an approximated version of the previous one.

- (M1). Closed queueing network (CQN) with blocking.
- (M2). Closed queueing network (CQN) without blocking.
- (M3). Transportation network without buffer cells.
- (M4). Transportation network with all cells.

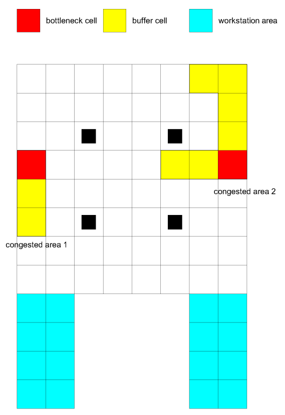
Based on our description, we can model the system as a closed queuing network where each cell is a first-come-first-served (FCFS) server with a buffer capacity of one (M1). This network is difficult to solve due to robot blocking, so we need further approximations. According to our observation, under a steady-state operation, some areas were prone to congestion, and queues from one congested area cannot spill to other congested areas (otherwise, these two areas should be combined as one). In each congested area, one cell usually served as a bottleneck, while the others served as buffers. We can model each congested area (several connected cells) as one FCFS server with an infinite buffer and each non-congested cell as an infinite server (IS) node. Therefore, the system can be considered a CQN without blocking

(M2) with some assumptions. Furthermore, we proved that for non-congested cells in a large system with many robots, the robot arrival process was asymptotic Poisson and numerically demonstrated similar results for congested areas. Consequently, we can decompose the CQN into individual queues. Due to the traffic flow conservation, the decomposed system is a transportation network whose link-delay function is either the waiting time of an M/G/1 queue for congested areas or the residual service time of an alternating renewal process for non-congested cells (M3). In addition, since the delay caused by buffer cells is trivial, we can add the buffer cells back to the transportation network as non-congested cells so that we don't need to specifically identify congested areas before running the algorithm, which gives us (M4). Using (M4), we can estimate the total delay in the system as closed-form functions of traffic flows.

Closed Queuing Network with Blocking (M1)



(a) A snapshot of the system



(b) Congested area

Figure 3.1: System

Consider the CQN in Figure 3.1a. Since the system is always busy, each cell is FCFS, and there is a finite number of robots in the steady state, the RSS can be modeled as a CQN with blocking shown in Figure 3.2a (M1). Each cell is a server with a service capacity of 1, that is, at most one robot in the server at one time.

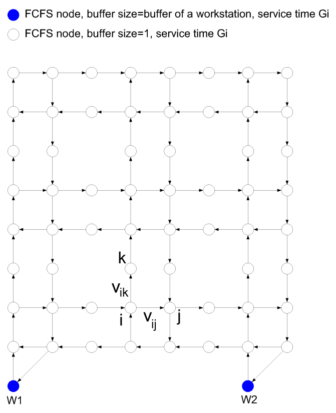
G_j is the service time on C_j. G_j is a mixture of different distributions. Consider the task set $\mathcal{K} := \{\text{go through}, \text{make a turn}, \text{drop item}, ...\}$, where each type of task $k \in \mathcal{K}$ takes T_k time units to complete. For example, if the travel time is deterministic, $T_{\text{go through}} = 2T_1$, $T_{\text{make a turn}} = 2T_1 + T_2$, $T_{\text{drop item}} = 2T_1 + T_{drop}$, and $T_{\text{pick item}} = 2T_1 + T_{load}$ (we add $2T_1$ since the robot must enter and leave the cell). Define the task-specific flow as v_j^k and the total incoming flow of one cell $v_j = \sum_{k\in\mathcal{K}} v_j^k$. Then, the first two moments of G_j are:

$$\mathbb{E}[G_j] = \sum_{k\in\mathcal{K}} \frac{v_j^k}{v_j}\mathbb{E}[T_k] \tag{3.1}$$

$$\mathbb{E}[G_j^2] = \sum_{k\in\mathcal{K}} \frac{v_j^k}{v_j}\mathbb{E}[T_k^2]. \tag{3.2}$$

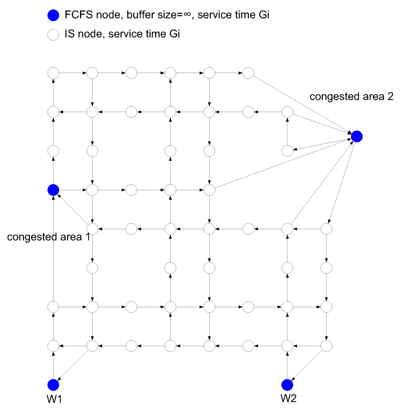
Note that the blocking type is neither before nor after service: if one robot blocks two cells and gets service from two cells simultaneously, the problem is difficult to analyze even if we know the traffic assignment on each link.

Closed Queuing Network without Blocking (M2)



● FCFS node, buffer size=buffer of a workstation, service time Gi
○ FCFS node, buffer size=1, service time Gi
k
v_ik
i
v_ij
j
W1
W2

(a) M1



● FCFS node, buffer size=∞, service time Gi
○ IS node, service time Gi
congested area 2
congested area 1
W1
W2

(b) M2

Figure 3.2: Closed queuing network models

Due to the complicated blocking process in (M1), (M1) is difficult to analyze. We will analyze the blocking mechanism and classify cells into different types and areas. To avoid spillovers, we merge cells in one area into one node in the queueing network, where some cells serve as bottleneck servers and others serve as buffers. Therefore, queues that spill from a bottleneck are contained in one congested area without interfering with other parts of the system, so we can model the system as a CQN without blocking (M2).

Consider a grid-based robot system. In steady state, for each cell C_i we can define *cell flow* $v_i = \sum_{j:C_j \text{is neighbor of } C_i} v_{i,j}$, and *cell occupation* $\xi_i = \lim_{T\to\infty}\int_0^T \frac{1(C_i \text{ is occupied at time t})}{T}dt$ as the proportion of time that C_i is occupied. We say C_i is congested if $\xi_i > \xi_0 > 0$, where ξ_0 is a threshold (we use $\xi_0 = 0.4$ in our simulations), indicating a persistent queue or a spillover of the queue in cell C_i. We define a random variable G_i as the free-flow service time for one robot in the cell C_i. Like a queuing system server, cell utilization ρ_i can be defined $\rho_i = v_i\mathbb{E}[G_i]$, with $\mathbb{E}[G_i]$ given in (3.1). Note that ξ_i can only be estimated by simulation, while ρ_i can be estimated based on flow distribution.

Based on industry practice experience with Geekplus [28] and our simulations, we notice two types of congested cells. The congestion in the first type of cells, *bottleneck cells* is caused by flow and large ρ_i. They are mostly free of spillovers from other cells. The congestion in the second type of cells, *buffer cells* is mainly caused by spillover from other cells, and their own ρ_i is small. (See Figure 3.1b).

One bottleneck cell will cause congestion in many cells due to its queue spillover. We call a set of connected congested cells a *congested area*. In one congested area, at least one bottleneck cell serves as the exit and the server, while the other congested but non-bottleneck cells serve as the buffer. In a steady state, each congested area can be modeled as a queueing node, where the number of servers is the number of bottleneck cells. This system itself is very complicated, since the service time, the server that a customer uses, and the customer priority all depend on the trajectory of each customer. We define congested cells, congested areas, and bottleneck cells as follows:

Definition 1 (A congested cell). *Cell C_i is a congested cell if $\xi_i > \xi_0$, where ξ_0 is the threshold parameter. In our experiment, we use $\xi_0 = 0.4$.*

Definition 2 (A bottleneck cell). *Let the expected robot number waiting to pass cell C_i be $\mathbb{E}[Q_i]$. If $\mathbb{E}[Q_i] > 1$, C_i is a bottleneck cell.*

Definition 3 (A congested area). *Consider a directed graph*

$$\mathcal{G}_c = (\mathcal{V}_c, \mathcal{A}_c) = (\{C_i : C_i \text{ is a congested cell}\}, \{(C_i, C_j) \in \mathcal{A} : C_i \text{ is not a bottleneck cell}\}).$$

A connected component of the undirected version of $\mathcal{G}_c$ is a congested area.

Definition 4 (A non-congested cell). *Cells outside congested areas are non-congested cells.*

Definition 5 (A buffer cell). *Non-bottleneck cells inside congested areas are buffer cells.*

Remark 1. *The definitions of a congested cell and area rely on ξ_i. ξ_i cannot be estimated without simulation or real-world experiments. The shape of the congestion area is not a priori.*

Remark 2. *The definitions of a bottleneck cell are based on $\mathbb{E}[Q_i]$. According to our assumptions, conjecture and theorem, $\mathbb{E}[Q_i]$ can be estimated given the flow assignment. (See Assumption 1) Therefore, we can identify bottleneck cells based on flow assignment.*

We make the following assumptions for the system to simplify (M1) to (M2):

Assumption 1. *There is, at most, one bottleneck cell in each congested area.*

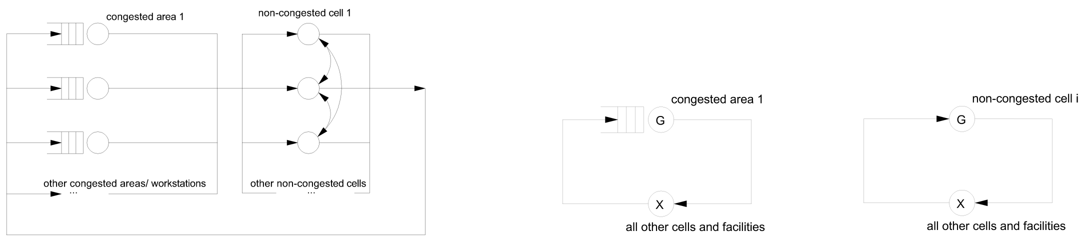
Assumption 2. *There is no blocking in non-congested cells, and the queues in congested areas will not spill over.*

It is difficult to analyze the blocking-unblocking process inside each congested area if there is more than one bottleneck. In a real-world system, if there is more than one bottleneck in one congested area, it indicates that the spillover from two cells meets and causes a very complex cell-blocking process. Such a system is prone to deadlocks, and operators will reduce the number of robots to avoid such scenarios. Therefore, it is not common for a well-designed, stable system to have multiple bottlenecks in the same congested area, and these two assumptions hold for general robotic warehouses.

With Assumptions 1 and 2, if congested areas are known, we can combine all cells in one congested area as one node to build a closed queueing network without blocking (M2) (Figure 3.2b). A congested area has enough buffer (Assumption 1) and can be modeled as an FCFS node. The non-congested cells can be modeled as infinite server (IS) nodes since we assume no blocking (Assumption 2). The service time is G_i if C_i is the bottleneck of the congested area. Note that the travel time on the buffer cells is part of the queue waiting time.

(M2) greatly simplifies (M1) but is still a closed queueing system with general service time. There is no closed-form solution for such systems. We will further approximate (M2).

Transportation Network without Buffer Cells (M3)


congested area 1
non-congested cell 1
other congested areas/ workstations
...
other non-congested cells
...
congested area 1
G
X
all other cells and facilities
non-congested cell i
G
X
all other cells and facilities

(a) M2 as a closed network (b) Finite system (c) Infinite system

Figure 3.3: Approximation of the closed queuing system

To simplify (M2), we want to "open" the CQN and approximate it with a transportation network with a closed-form link cost function. We will introduce new assumptions and show asymptotic Poisson properties to make the approximation possible. Since the arrival process at each node is asymptotic Poisson, we can approximate the waiting time at each node in the CQN (M2) as a traffic flow network (M3), where the link delay in (M3) is the queue delay in (M2), as long as the incoming flows at each node are the same and adding a source and a sink to preserve flow conservation (M3).

New Assumptions and the Re-organized Queuing Network

Since congested areas are separated by non-congested cells (otherwise, we can merge two congested areas as one), (M2) can be re-organized as the closed queuing network in Figure 3.3a.

Consider one congested area with bottleneck cell C_i and two consecutive arrivals in the congested area of the same robot. Assume the robot's path after leaving C_i and entering this congested area again is $\{C_{i_1}, C_{i_2}, ..., C_{i_j}, \text{The congested area with bottleneck } C_{i_{j+1}}, C_{i_{j+2}} + ...\}$, where $C_{i_1}, C_{i_2}, ...$ are non-congested cells. Let X_i be the time that the robot spent outside the congested area between two arrivals. Let Y_i be the time spent inside a congested area with bottleneck C_i. We have

$$X_i = G_{i_1} + G_{i_2} + ... + G_{i_j} + Y_{i_{j+1}} + G_{i_{j+2}} + ...$$

where G_{i_j} is the service time of the non-congested cell C_j, and $Y_{i_{j+1}}$ is the time spent in the congested area with bottleneck $C_{j_{i+1}}$.

Similarly, we can define X_i as the time the robot spent outside C_i between two consecutive arrivals at the non-congested cell of the same robot. When analyzing one congested area or one cell, we can re-organize the model in Figure 3.3a to the system in Figure 3.3b for a congested area, and Figure 3.3c for a non-congested cell. We introduce the following two assumptions:

Assumption 3. *The service time in the congested area in Figure 3.2b (or a non-congested cell in Figure 3.2b) G and the service time outside the congested area (or a non-congested cell) X are independent and identically distributed (i.i.d.), respectively. G is much greater than X with high probability.*

The service time G is i.i.d. because the spillover of other cells does not block the bottleneck cell. Otherwise, these two congested areas should be counted as one larger congested area. The service time G depends only on the robot's operation in the bottleneck cell.

After leaving the congested area, the robot trajectories became unpredictable. In a large system, robots leaving the same cell will quickly split into different areas, so they will likely not interfere with each other, and the robots are identical. Except for the congested area (or non-congested cell), the system is big enough to accommodate all the robots. Therefore, we can assume X is i.i.d., and the node X is an infinite-server node.

In a large system, a robot must go through many other congested areas and complete different tasks before returning to the same cell. Therefore, a robot will spend much longer in node X than G.

Assumption 4. *All robots in one congested area or non-congested cell go through the bottleneck cell first-come-first-served (FCFS).*

Many robot path conflict and deadlock solving algorithms, like [8], and the algorithm used by [28], allow robots to reserve cells in advance. If one robot has reserved the bottleneck cell, the next robot has to wait. Therefore, the bottleneck area is usually FCFS in grid-based robot systems.

According to Assumptions 1, 3, and 4, the system that focuses on one of the congested areas can be represented as the CQN in Figure 3.3b where the congested area G is a G/G/1 queue, and the remaining part of the system X is a GI/GI/∞ queue. The system focusing on one non-congested cell can be represented in Figure 3.3c where the non-congested cell G is a GI/GI/∞ queue, and the remaining part of the system X is a GI/GI/∞ queue.

Asymptotic Poisson Arrival

Focusing on one non-congested cell (Figre 3.3c), we can show the arrival is an asymptotic Poisson for a large system with numerous robots.

Theorem 1. *For the CQN in Figure 3.3c with two infinite server nodes G, X, under the assumption 3, with a fixed desired flow rate v, and robot number* $R = \lceil v(\mathbb{E}[X] + \mathbb{E}[G]) \rceil$*, random service time G at the first node and X at the second node, as* $X \to \infty$ *in probability, the arrival process at the first node is asymptotic Poisson*(v).

The proof is given in Appendix Section A.1.

We can anticipate a similar result for congested areas as in Theorem 1. According to Assumptions 1, 3, and 4, the system that focuses on one of the congested areas can be represented as the CQN in Figure 3.3b where the congested area G is a G/G/1 queue, and the remaining part of the system X is a GI/GI/∞ queue.

Conjecture 1. *For the CQN in Figure 3.3b. Assume that the service time at the first node is G, G is bounded, and that the service time at the second node X has finite variance. With a fixed desired flow rate v at the first node, and robot number* $R = \lceil v(\mathbb{E}[X] + \mathbb{E}[G]) \rceil$, *as* $X \to \infty$ *in probability, the coefficient of variance (CV) of inter-arrival times at the first node* $\to 1$.

A nonstrict proof and numerical evidence supporting the Conjecture 1 are given in Appendix Section A.2.

From (M2) to (M3)

Based on our analysis on (M2) (Conjecture 1 and Theorem 1), every node in the closed queuing system has approximate Poisson independent arrival. In addition, flow conservation holds for all types of robots. Therefore, we can use a transportation network model (M3) to evaluate the total delay. (M3) includes the same nodes and links as (M2), and also includes one source node S and a sink node T (Figure 3.4a).

To represent the operation of the system and preserve flow conservation, we introduce two types of flow: forward flow and backward flow. The forward flow originates from the source node S and ends at the sink node T, representing the flow of parcels through the system. The backward flow originates from T and ends at S, representing the flow of empty robots returning to the workstations. Let the set of acyclic routes from S to T be $\mathcal{R}_F$, and the set of routes from T to S be $\mathcal{R}_B$, and $\mathcal{R} = \mathcal{R}_F \cup \mathcal{R}_B$. Let the forward flow intensity on $r \in \mathcal{R}$ be f_r^F and the backward flow intensity be f_r^B. For simplicity, let $f_r = f_r^F$ if $r \in \mathcal{R}_F$ and $f_r = f_r^B$ if $r \in \mathcal{R}_B$.

The path flow can uniquely determine the arc flow. Letting $v_{i,j}$ be the flow on the arc i, j, we have

$$v_{i,j} = \sum_{r \in \mathcal{R}_F} f_r^F \delta_{ijr} + \sum_{r \in \mathcal{R}_B} f_r^B \delta_{ijr}, \tag{3.3}$$

where $\delta_{ijr} = 1$ if link (i, j) is in path r.

In the next subsection, we will derive the link delay function from Equations (3.4) and (3.6). Note that the two-direction arcs do not represent the robot flow. We use them to make sure forward or backward paths connect source and sink nodes. We will impose high travel costs for specific flow directions to prevent forward flow from returning to the source or backward flow from returning to the sink. The actual robot movement path only contains normal cell nodes and workstation nodes. For special nodes, let $c_{i,j}^F$ be the cost for forward flow and $c_{i,j}^B$ be the cost for backward flow. We define $c_{i,source}^F = \infty$, $c_{i,source}^F = 0$, $c_{source,i}^F = 0$, $c_{source,i}^B = \infty$ if i is a workstation, $c_{i,j}^F = 0$, $c_{i,j}^B = \infty$, $c_{j,i}^F = \infty$, $c_{j,i}^B = 0$ if j is a drop-off point and C_i is a cell next to j, and $c_{i,j}^F = \infty$, $c_{j,i}^B = \infty$ if j is a workstation and C_i is a cell next to j. Therefore, no forward flow is allowed to enter the source, enter a workstation, or leave a drop-off point. No backward flow is allowed to leave the source, leave a workstation, or enter a drop-off point.

If $v_{i,j}$ in (M3) is the same as $v_{i,j}$ in (M2) for all i, j nodes, the total delay calculated from (M3) is an approximation of the total delay in (M2). (M3) is simple enough with closed-form delay functions. However, congested areas are difficult to identify without running simulations, so we must further approximate the system, as shown in (M4).

Heavy Traffic: Delay for Congested Areas

We define c_{ij} as the waiting time for robots entering from node i to j in (M3). Using Conjecture 1, the arrival can be approximated as Poisson for a large system. The FCFS node representing one congested area can be approximated as an M / G / 1 queueing system in heavy traffic, with G_j as the service time. Using Kingman's formula for the G/G/1 queue under heavy traffic, we have:

$$c_{ij} = \frac{v_j \mathbb{E}[G_j^2]}{2(1 - v_j \mathbb{E}[G_j])}, \tag{3.4}$$

where $v_j = \sum_{i:C_i \text{ is } C_j\text{'s neigbor}} v_{i,j}$ is the inflow of C_j. We approximate the CV of inter-arrival times $CV_j \approx 1$ using Conjecture 1 because the service time in one congested area is bounded, and the service time in the remaining part of the system is much greater than G_j in a large system between two arrivals at C_j for one robot. The righthand side of Equation (3.4) is the expected waiting time in an M/G/1 queue using Kingman's formula. G_j is the service time when C_j is free of spillovers from other cells given in (3.1) and (3.2).

The length of the queue in a congested area Q_j is an important identifier of a congested area according to Definition 2. Using Little's law, the queue length can be estimated as

$$\mathbb{E}[Q_j] = \frac{v_j^2 \mathbb{E}[G_j^2]}{2(1 - v_j \mathbb{E}[G_j])} + v_j \mathbb{E}[G_j] \tag{3.5}$$

Light Traffic: Delay for Non-Congested Cells

For non-congested cells, although we ignore the blocking in (M2) to obtain the Poisson arrival properties (Theorem 1 and Conjecture 1). There could still be blocking near non-congested cells. To reduce the approximation error, we need to estimate the delay in non-congested cells with the following assumption:

Assumption 5. *If cell C_j is not a bottleneck, with a probability of 1, queue length ≤ 2. i.e., the blocking that occurred in C_j must be one-on-one (one robot blocks another, no spillover). In addition, the service time outside the cell is much longer than G_j in probability.*

Conditioned that the queue length is less than 2, the only collision possible is one-on-one blocking with no spillovers: for cells with an expected queue length of less than 1, the queue length is less than 2 with high probability, so this assumption is valid. Using Theorem 1, the arrival at a non-congested C_j is approximately Poisson. Let $G_{i,j}$ be the service time in C_j observed by robots from C_i. The cell is altering between two statuses: blocked and cleared. The lifetime of blocked status observed by robots entering C_j from C_i is the service time $G_{i,j}$, and the lifetime of one cycle (blocked and then cleared) is the inter-arrival time of robots at the cell. The block-clear process is an alternating renewal process, and the waiting time for C_i is the remaining lifetime of the blocking time when a robot arrives. Using Poisson arrivals see time averages (PASTA) and renew theorem, the expected link delay is:

$$c_{i,j} = \frac{v_j}{2}\mathbb{E}[G_{i,j}^2]. \tag{3.6}$$

Due to the "platooning effect," $G_{i,j}$ differs from G_j. Considering two robots arriving from C_i to C_j next to each other, if the first robot occupied C_j for T_k, the second robot's waiting time can be at most $T_k - 2T_1$, where T_1 is the time used to move one cell, because these two robots also need to leave C_i, and the minimum departure interval is $2T_1$ to avoid collision. Therefore, conditioned that the previous robot also enters from C_i,

$$\mathbb{E}[G_{i,j}^2|\text{The previous robot also from } C_i] = \mathbb{E}[(G_j - 2T_1)^2]$$

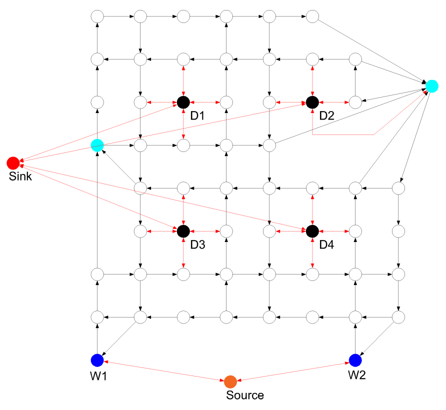
Therefore, we have

$$\mathbb{E}[G_{i,j}^2] = \mathbb{E}[(G_j - 2T_1)^2]\frac{v_{i,j}}{v_j} + \mathbb{E}[(G_j)^2]\frac{v_j - v_{i,j}}{v_j} \tag{3.7}$$

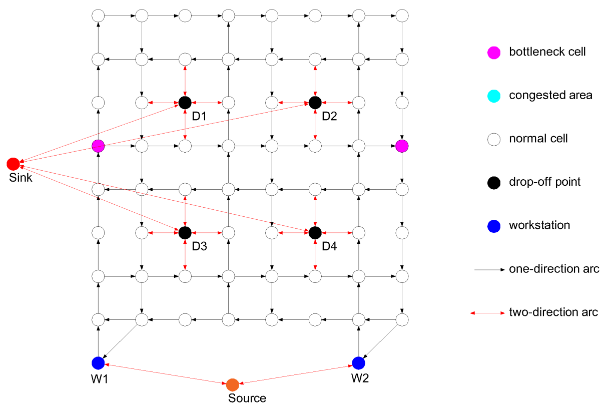
Transportation Network with All the Cells (M4)

To avoid the necessity of identifying congested areas, we can define a new network from (M3) by adding the merged buffer cell nodes back. We notice that due to our treatment of robot platooning and definition of the delay $c_{i,j}$, the transportation delay on links to buffer cells is trivial; therefore, adding them back will not cause much error from (M3), and we can estimate the transportation delay using (M4).

From (M3), we make the following modifications to build (M4):



(a) M3



(b) M4

Figure 3.4: Open queuing network models

- Decomposing each congested area node into one node representing bottleneck cell and several nodes representing buffer cells.
- The connection structure among the newly added nodes is the same as the original cell structure in (M1).
- The link delay function $c_{i,j}$ where C_j is a bottleneck cell is the same as Equation (3.4).
- The link delay function $c_{i,j}$ where C_j is a buffer cell is the same as Equation (3.6).

We define $c_{i,j}$ as the delay incurred by cell C_j when entering form C_i, so the delay in a congested area is considered as the delay in the bottleneck cell, not the buffer cell. For example, if a robot travels on the path C_k, C_i, C_j, where C_j is a bottleneck cell, and the queue spills to C_k, and the robot gets a delay of T_{kij} before leaving C_j, then $c_{k,i} = 0$, and $c_{i,j} = T_{kij}$ for this robot.

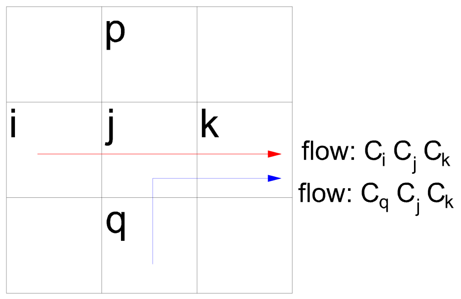


Figure 3.5: A buffer cell example

The total delay in (M4) is greater than in (M3) because we introduced new links with positive $c_{i,j}$ at buffer cells C_j compared with (M3). However, we will show that for the buffer cell C_j, the newly introduced $c_{i,j} \approx 0$ in (M4).

If the buffer cell C_j has a large flow input, almost all robots that pass C_j will take the same path near C_j and make no turns. For example, in Figure 3.5, C_j is a buffer, with C_i, C_k, C_p, C_q as its neighbor, and almost all robots follow the path C_i, C_j, C_k (red flow in Figure 3.5). In contrast, almost no robots take other paths like C_q, C_j, C_k (blue flow in Figure 3.5). Otherwise, the conflict of flow directions or turning operation with large flow will incur large delays in C_i, making C_j a bottleneck instead of a buffer cell. According to our treatment of the platoon effect (3.7), in (M4), $c_{i,j} = 0$ if all robots follow the path C_i, C_j, C_k since the service time $G_j = 2T_1$ because no robots are making turns or doing other tasks on C_j.

If the buffer cell C_j has a small flow input, $c_{i,j} \approx 0$ according to (3.6) since $v_{i,j} \approx 0$. Therefore, for newly added links (C_i, C_j) in (M4), $c_{i,j} \approx 0$, so (M4) approximates the total delay of (M3).

By Definition 2, although we cannot easily identify congested areas or buffer cells, we can identify bottleneck cells using Equation (3.5). If $\mathbb{E}[Q_j] > 1$, we can classify the cells as bottlenecks. Therefore, combining equations (3.4) and (3.6), we have for all cells $C_i, C_j \in \mathcal{C}$, C_j connected to C_i,

$$c_{i,j} = \begin{cases} \frac{v_j}{2}\mathbb{E}[G_{i,j}^2] & \text{if } \frac{v_j^2\mathbb{E}[G_j^2]}{2(1-v_j\mathbb{E}[G_j])} + v_j\mathbb{E}[G_j] \le 1 \\ \frac{v_j\mathbb{E}[G_j^2]}{2(1-v_j\mathbb{E}[G_j])} & \text{otherwise} \end{cases} \tag{3.8}$$

Consider the flow routes r from S to T for forward flow or T to S for backward flow. We can use the delay from (M4) to approximate the total delay in (M3) and thus in (M2) and (M1). The cost on the route r (machine time spent per hour) is

$$\zeta_r(\mathbf{f}) = \sum_{(i,j) \text{ on path } r} c_{i,j}(\mathbf{f}) + \sum_{(i,j),(j,k)\in r} T_2\delta_{ijk}^{\text{turn}} + \sum_{(i,j)\in r} T_1, \tag{3.9}$$

where $\mathbf{f} := [\mathbf{f}^F, \mathbf{f}^D] = [[f_r^F], [f_r^B]]$ is the vector of all path-flow variables, and $c_{i,j}(\mathbf{f})$ is the expected delay when entering C_i from C_j, and is a function of flow. $\delta_{ijk}^{\text{turn}} = 1$ if the robot that goes from cell C_i to C_j then to C_k needs to make a 90 degree turn in cell C_j; otherwise, $\delta_{ijk}^{\text{turn}} = 0$. The first term in Equation ((3.9)) is the travel delay incurred by waiting when C_j is blocked. The second term is the turning time because it takes T_2 to turn. The third term is the travel time from one cell to another. The sum of the second and third terms is the free-flow travel time. The total cost of the system transportation can be estimated as

$$TC(\mathbf{f}) = \sum_{r\in\mathcal{R}_F} f_r^F\zeta_r(\mathbf{f}) + \sum_{r\in\mathcal{R}_B} f_r^B\zeta_r(\mathbf{f}). \tag{3.10}$$

Therefore, using (M4), we can estimate the link and system transportation cost of the RSS.

3.4 Throughput Estimation

Iteratively Solving CQN and Link Delay

With the previous model (M4), we can estimate the traveling delay for each vehicle at a steady state if we know the traffic flow and system throughput. However, we cannot estimate link flow and transportation delay without throughput information, and we cannot estimate CQN throughput without knowing transportation delay. Therefore, we will develop an iterative algorithm to estimate throughput and traffic delay, given the number of robots and the system parameters. In this algorithm, (M4) is used to evaluate the traffic delay, and a new CQN (a re-organized (M1)) is used to evaluate the throughput with updated traffic delay information from (M4).

To obtain throughput and evaluate overall performance with a finite number of robots in the system, we return to the closed queuing network (M1) and re-organize the network to the CQN in Figure 3.6. In this CQN (Figure 3.6), there are R robots in total. The server node $W_1, ..., W_{n_W}$ represents the n_W workstations and are FCFS nodes. The first two moments of the processing time distribution T_{load} are known. D_j nodes represent the dropping process. Since the waiting time for drop-off points is already considered a traveling delay, these servers are infinite servers with service time T_{drop}.

After picking up a parcel from the workstations, the robots will follow their assigned route, traveling from the workstation W_i to the drop-off point D_j. The infinite server (IS) node W_iD_j represents the travel time from W_i to D_j. Let the set of paths from W_i to D_j be

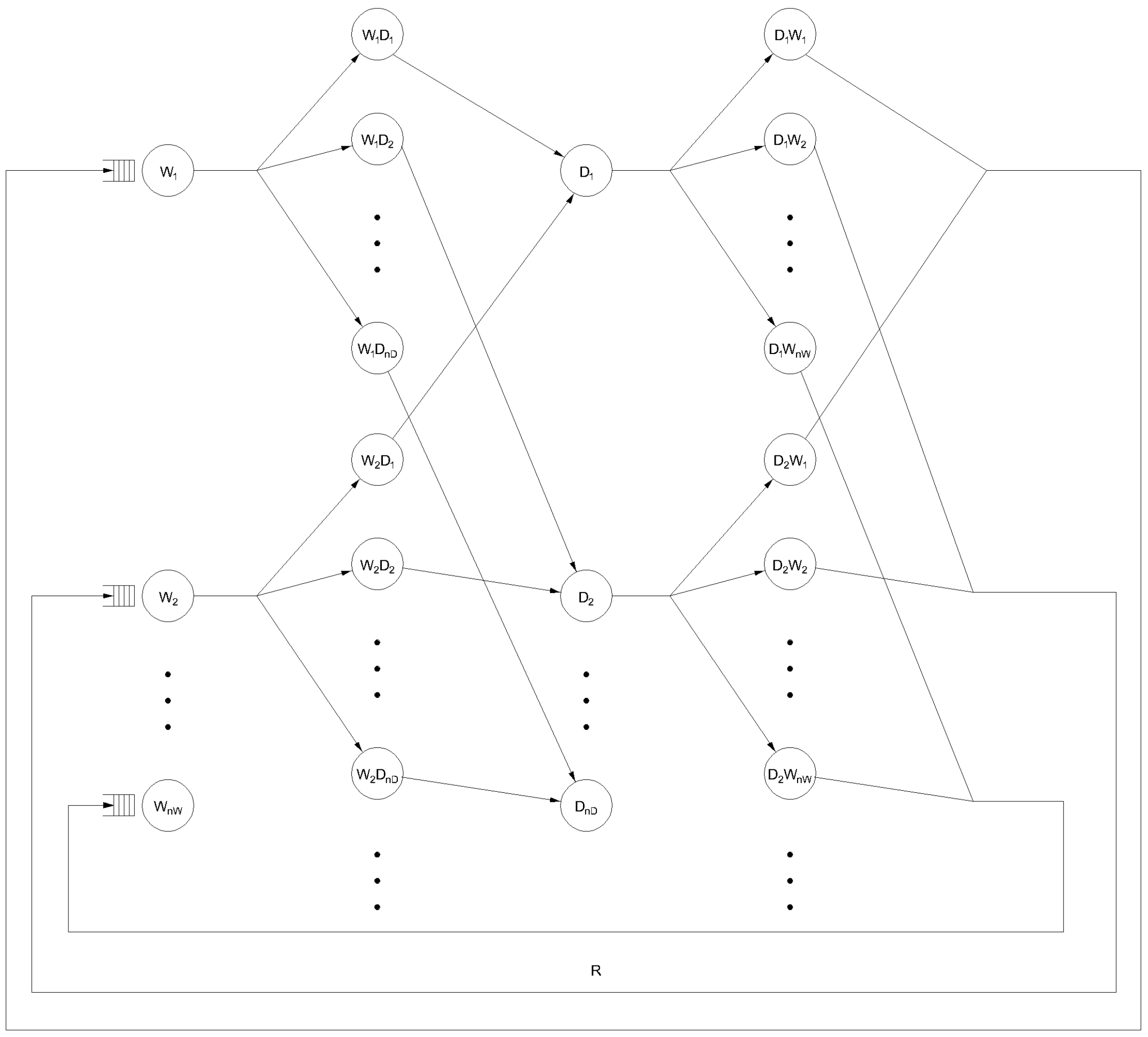


Figure 3.6: CQN embedding the traffic delay

$\mathcal{R}(W_i, D_j)$, the expected travel time from W_i to D_j is

$$\sum_{r \in \mathcal{R}(W_i, D_j)} \frac{f_r^F \zeta_r}{\sum_{r \in \mathcal{R}(W_i, D_j)} f_r^F}$$

. The split probabilities p_{W_i,D_j} depend on the assignment strategy used (for random assignment: $p_{W_i,D_j} = p_{W_i,D_k}, \forall D_j, D_k$, for zoning assignment, the probability of using a drop-off point outside the zone is zero). After dropping the parcel, the vehicle returns to the workstations through the D_iW_j nodes. These infinite server nodes work as W_iD_j, representing the travel time from drop-off points to workstations.

Since the travel delay is given by the traffic flow network (M4), depending on the throughput, and the throughput is estimated using the CQN, whose service time depends on the travel delay, we use an iterative algorithm to evaluate the system. We initialize the network with 0 travel delays. Then repeat the following steps until the throughput converges. The throughput and network flow assignment given in the last iteration step will be the final estimated throughput.

Throughput Estimation

- Step 0. Initialize: Set all travel delays to 0, and travel time to free-flow the shortest path travel time.
- Step 1. Solve the CQN with the current traffic delay to estimate the throughput using the AMVA Algorithm 1.
- Step 2. Using the throughput from the last step, assign traffic flow according to assignment and routing rules (for example, assign each origin-destination route using the shortest path), then update the link delay using (3.8) for each arc between cells, and calculate ζ_r for all paths.
- Step 3. Check if the updated flow distribution and delay estimation are converged. If converged, stop. Otherwise, go back to step 1.

AMVA Algorithm for CQN

To solve the CQN in Step 1, we used the approximate mean value analysis (AMVA) method in[14]. The basic approximation assumption for the algorithm is that we have a PASTA or similar property for this CQN. The first moment of the delay distribution can be estimated using M/G/1 or the renewal process, which approximates the delay function.

Note that the second moment is required for AMVA. Let $Y_{i,j}$ be the delay from C_i to C_j, $\mathbb{E}[Y_{i,j}] = c_{i,j}$. For M/G/1 approximated cells, we can use the Pollaczek-Khinchine method to obtain the probability generating functions (PGF). The derivatives of PGF give us $\mathbb{E}[Y_{i,j}^2]$. For renew-process approximated cells (non-congested cells), the second moment is $\frac{\mathbb{E}[G_{i,j}^3]v_j}{3}$. The second moment of the service time on one route is the second moment of the summation of independent link travel times:

$$\mathbb{E}[(\sum_{(i,j)\text{ on path } r} Y_{i,j} + \sum_{(i,j),(j,k)\in r} T_2\delta_{ijk}^{\text{turn}} + \sum_{(i,j)\in r} T_1)^2]$$

where Y_{ij} is the delay from C_i to C_j, $\mathbb{E}[Y_{ij}] = c_{i,j}$.

Algorithm 1 AMVA algorithm

```
Input: The queuing network in Figure 3.6 with $M$ nodes and $R$ robots, with the first two
    moments $\mathbb{E}[S_m], \mathbb{E}[S_m^2]$ of service time numerically calculated.
Output: Flow at each node $TH(n)\nu_m$
  Find visit ratio vector $\nu$ by solving $\nu P = \nu$ where $P$ is the route matrix, whose entry $P_{ij}$
  is the routing probability that a vehicle from node i that goes to node j, $\sum_j P_{ij} = 1$.
  Initialize: set expected queue length $EN_i(0) = 0$, $i = 1, ..., M$, and throughput $TH(0) = 0$
  for $n \leftarrow 1$ to $R$ do
      for $i \leftarrow 1$ to $M$ do
          if $i$ is FCFS then
              $ET_i(n) \leftarrow \frac{\mathbb{E}[S_i^2]}{2}\nu_i TH(n-1) + \mathbb{E}[S_i]\{EN_i(n-1) + 1 - \nu_i TH(n-1)\mathbb{E}[S_i]\}$
          else if $i$ is IS then
              $ET_i(n) \leftarrow \mathbb{E}[S_i]$
          end if
      end for
      $TH(n) \leftarrow \frac{n}{\sum_{i=1}^{M} \nu_i ET_i(n)}$
      for $i \leftarrow 1$ to $M$ do
          $EL_i(n) \leftarrow \nu_i TH(n) ET_i(n)$
      end for
  end for
```

3.5 Numerical Results

Validation of the Congestion Mechanism

In this section, we will validate our congestion mechanism and assumptions.

Simulation Settings

Our experiments were conducted on a small (16*20) RSS with two workstations and 24 drop-off points (see Figure 3.7). One of the project plans of [28] inspired the design of the system. We set $T_1 = 1$, $T_2 = 5$, $T_{\text{load}} = 4$, and $T_{\text{drop}} = 2$ in time steps. We set constant pickup and drop-off times to simplify the calculation. We assumed that a parcel's probability of going to each drop-off point is equal. We assumed that there were enough parcels to be sorted and run the system with given robots to complete as many tasks as possible, so idle robots were immediately assigned new tasks. (It is a common practice to collect parcels on the first day and sort them on the second day. On the second day, we run at the maximum capacity until all jobs are complete). Our traffic control and deadlock resolving algorithm was adapted from that in [32]. Note that this deadlock-resolving algorithm cannot guarantee a deadlock-free operation. If an unsolvable deadlock exists, we stop the simulation and record the successful time steps until the unsolvable deadlock.

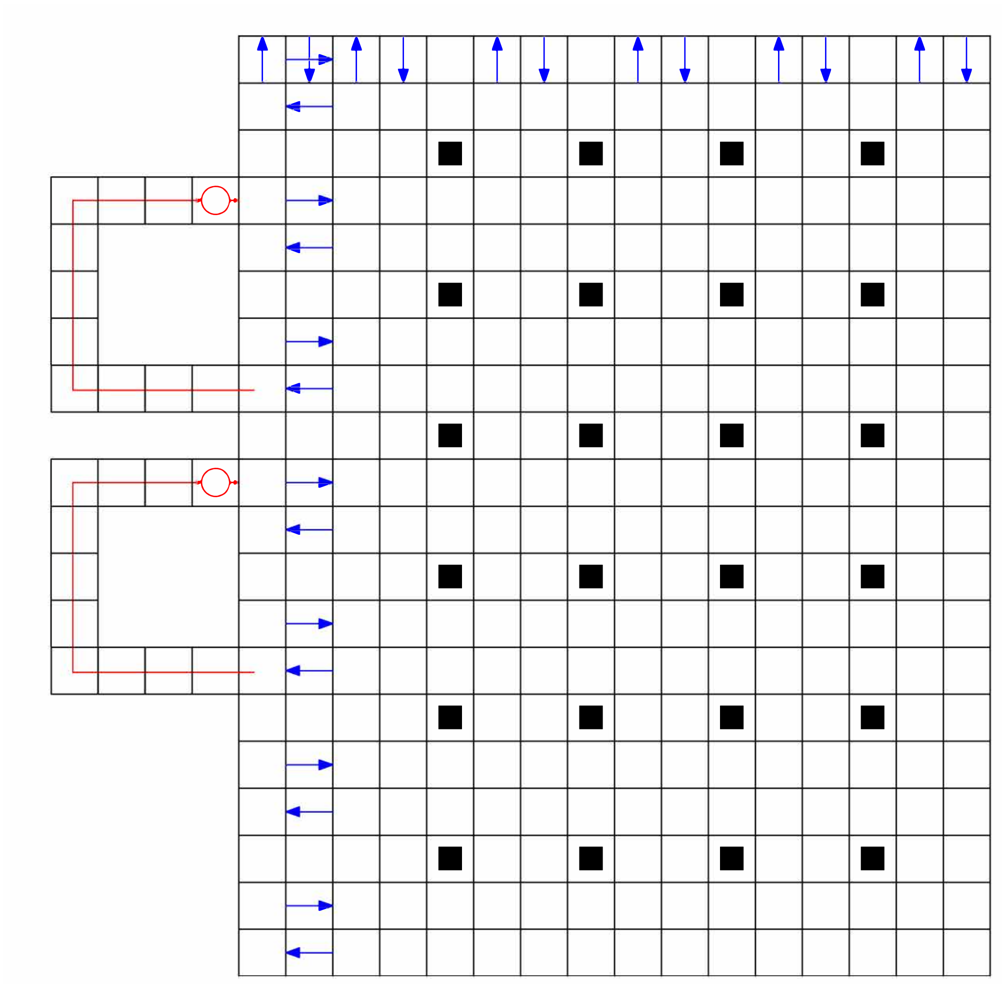


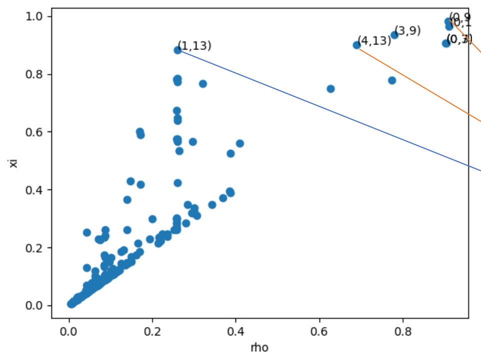
Figure 3.7: Simulation setting: congestion mechanism

Congestion Areas

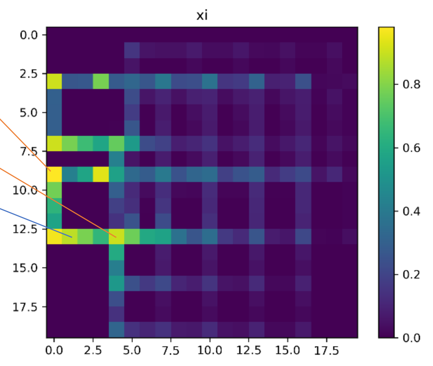
The first set of experiments was to verify our description of the congestion mechanism, assumptions, and theorem. Using the small system roadmap with the number of 15, 20, 25, 30, and 35 robots for 50,000 time steps each, we recorded cell occupations ξ_i, utilization ρ_i, and interarrival times in each cell.

The cell ξ_i, ρ_i distribution is shown in Figures 3.8a and 3.8c; each point corresponds to one cell in an experiment. Most cells have $\rho_i \approx \xi_i$, indicating that they are not affected by queue spillovers. The first group of cells lies near $\xi_i = \rho_i$ and has large ρ_i and ξ_i. They are bottleneck cells. The second group of cells lies above $\xi_i = \rho_i$, with small ρ_i but large ξ_i. These cells are buffer cells. The last group is the non-congested cells near $\xi_i = \rho_i$ with small ρ_i and ξ_i.

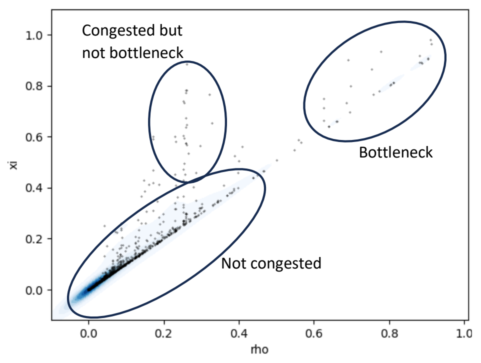
When $R = 30$, the system is almost saturated and we show the distribution of ξ_i, ρ_i of



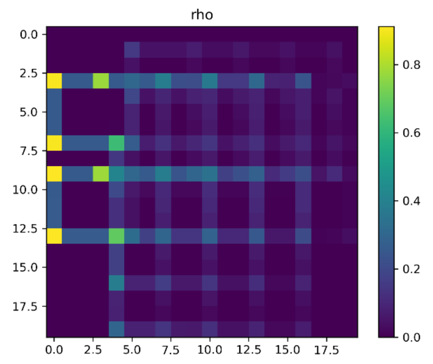
(a). Occupation ξ_i vs utilization ρ_i for one simulation



(b). Heatmap of ξ_i on each cell



(c). ξ_i vs ρ_i from different R combined



(d). Heatmap of ρ_i on each cell

Figure 3.8: Congestion in a grid-based AMR system

each cell in Figures 3.8a, 3.8b, and 3.8d. Some cells are bottlenecks, such as cells (0,9) and (4,13), with $\xi_{(0,9)} \approx \rho_{(0,9)} \approx 1$ in Figure 3.8a. Some cells are buffer cells, like cell (1,13): $\rho_{(1,13)} \approx 0.3$ but $\xi_{(1,13)} \approx 0.9$, it serves as a buffer for queues from cell (0,13).

Poisson Arrival

First, we verified whether the system can reach a steady state. After simulating for a very long time (50000 time steps, with $T_1 = 1$ time step), we counted the number of robots traveling through each cell every 1000 time steps. The results of some cells are shown in Figure 3.9. After the first 1000 steps, we can see that the flow becomes relatively stable and the fluctuation is about 10%. Therefore, we can conclude that this system can reach a steady state.

Then, we numerically verified Theorem 1 and Conjecture 1 in the systems with the simulation. We calculated the coefficient of variations (CV) of the inter-arrival time in

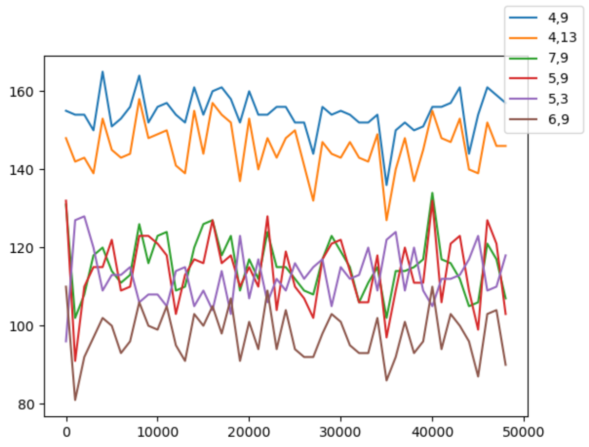
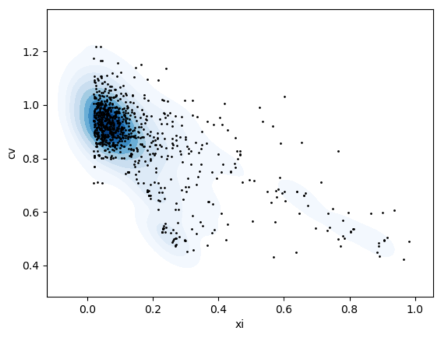
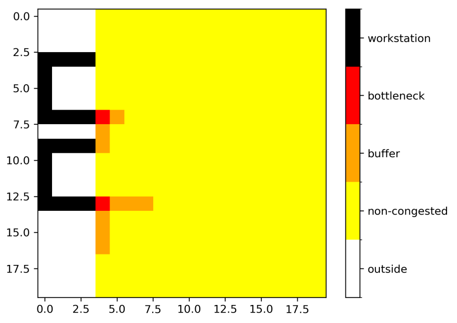


Figure 3.9: Flow on different cells every 1000 timesteps



(a) CV vs occupation



(b) Category of cells when $R = 30$

Figure 3.10: Accuracy of the objective function

different cells when $R = 30$. In Figure 3.10a, each point represents the occupation of one cell ξ_i and CV in one experiment, and the density map of the kernel of (ξ_i, CV) is the blue cloud. Most non-congested cells have a CV ≈ 1. CV becomes less than 1 if the cell is congested. From this observation, we can confirm that theorem 1 applies to non-congested cells.

Cells with large ξ seem to have $CV < 1$. However, this CV is not for the inter-arrival time at a congested area but is the inter-arrival between cells inside the same area. If C_i is a bottleneck cell and is always busy, the inter-arrival on C_i is just G_i. In our case, the free flow service time G_i is a categorical distribution (see Equation (3.1)) with $CV < 1$. Therefore, these cells' interarrival time CVs are less than 1. The actual CV for each congested area should be calculated for the whole area. In our experiment, the CV of the interarrival time is 0.971 and 0.940 for the two congested areas in Figure 3.10b. Therefore, our conjecture applies to this simulation, since both CVs are approximately 1.

An insight is that warehouses with a certain layout will result in large estimation errors due to the CV deviating from 1. For example, the corridor entering the workstation with two turning points (for example, cell (0,13), then (0,9) in our setting) resulted in tandem bottlenecks. Both cells have a high level of traffic; the second cell's input is the first cell's output. The inter-arrival in the second cell will be approximately the service time G of the first cell. If G is not exponential, the arrival in the second cell will not be Poisson. However, we assumed i.i.d. service time outside each congested area (Assumption 3). The assumption is valid if robot flows split after one server quickly and do not interfere with each other in a large system. But in this system, the robots all stay on the same path in a tandem bottleneck system because they must sequentially go through the same bottlenecks. We should merge these tandem bottlenecks into one congested area (see Figures 3.1a and 3.10b, where one workstation, including tandem turning points, is merged into one node). In our problem, since the service time for a given type of robot task is deterministic, robots leave the tandem queues with an interval of the largest service time among these queues, which can be used as the service time of the merged queues. For more complicated problems, we can use methods such as [19] to find the new waiting times distributions for the merged nodes. Therefore, our model is more suitable for a large system where robots have more freedom and are not limited to one or a few corridors, such that they split into different paths quickly after each congested area. Otherwise, these special layout structures must be modeled as special nodes in the transportation network.

Using the queue length condition in Equation ((3.8)), considering cells with $\xi > 0.4$ as congested, we showed the category of cells in Figure 3.10b. Note that we merge cells near each workstation as one congested area, called a "workstation", to deal with the tandem-bottleneck problem. There are bottlenecks near the entrance of each server, and several congested cells nearby serve as buffer areas. Our Assumption 1 is true for this experiment setting because there is one bottleneck in each congested area.

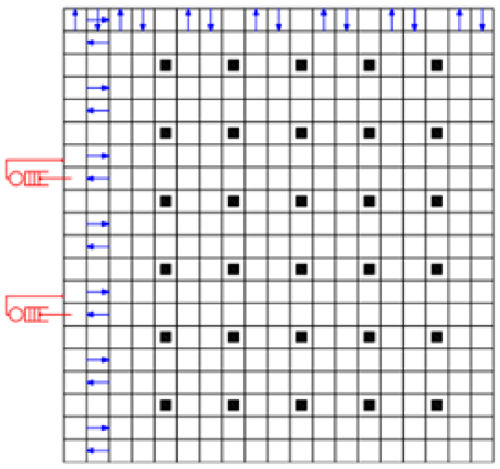
Conclusion

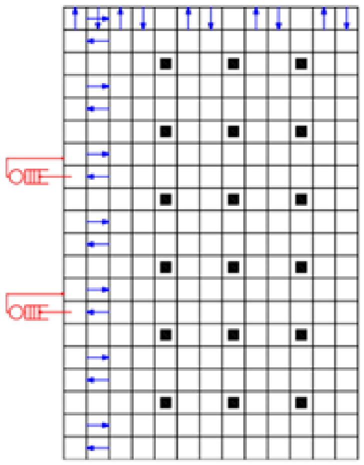
Based on the simulation, we conclude:

- The cells can be classified into non-congested cells, bottleneck cells, and buffer cells, based on the occupation ξ and utilization ρ (See Figure 3.8 and 3.10b).
- The system can reach steady state (see Figure 3.9).
- Our theorem and conjecture are valid for non-congested cells and most congested areas, since they have inter-arrival times $CV \approx 1$. (See Figure 3.10a)
- Layout structures like tandem queues violate our assumptions, so we must merge them into one congested area to reduce the estimation error. (See Figure 3.10b, where we merge congested areas near workstations into a "workstation" area.)

Validation of the Throughput Estimation

Simulation Settings


(a) A: standard system


(b) B: half system

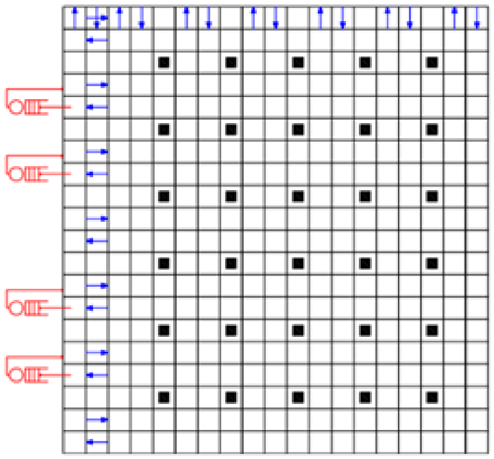

(c) C: more workstations

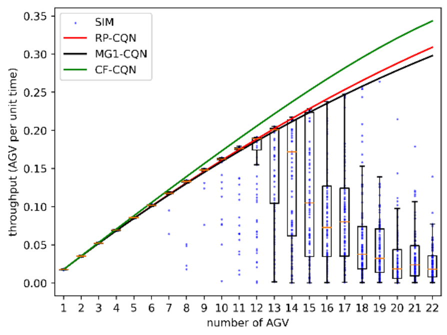
Figure 3.11: Different layout used in throughput validation experiments

We applied our method in Section 3.4 to three different layout designs shown in Table 3.1 and Figure 3.11. We assumed that all robots would follow the shortest path route, and tasks were assigned randomly among workstations and drop-off points. We assumed that the demand arrival rate was large enough to have no idle robot, since we were focused on the maximum throughput analysis. Our traffic control and deadlock resolving algorithm was adapted from that in [32]. Other parameters were the same as in the previous experiments. We run 10000 steps for each simulation experiment. We increased the number of robots from 1 to 22 and performed 100 trials for each setting of robot numbers.

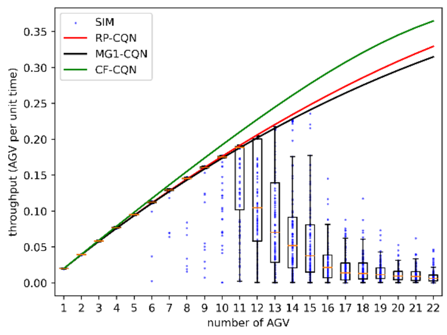
Layout	Height	Width	n_W	n_D	Roadmap
A	20	19	2	30	Figure 3.11a
B	11	19	1	15	Figure 3.11b
C	20	19	4	30	Figure 3.11c

Table 3.1: Layout of throughput estimation validation experiment

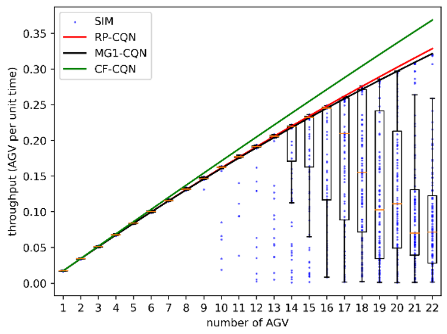
Throughput Results



(a) Throughput-density of the standard system (A)



(b) Throughput-density of the standard system (B)



(c) Throughput-density of the standard system (C)

Figure 3.12: Throughput validation results

We will compare the performance metrics calculated using the following four methods:

- RP-CQN (Renewal process approximated CQN): CQN with the renewal process approximated the cell delay function. This simplifies our method, assuming that all cells are non-congested and using Equation (3.6) to estimate the delay.

- MG1-CQN (M/G/1 queue approximated CQN): CQN with M/G/1 approximated cell delay function. This is our proposed method for evaluation.
- CF-CQN (Congestion-free CQN): Run AMVA without considering congestion. This is the benchmark method where we use the free-flow travel time and the shortest path travel time and ignore all congestion with AMVA CQN throughput estimation.
- SIM (Simulation): Simulation result.

The simulation shows that throughput increases as we add more robots in the system. At certain levels of robot density, the deadlock resolving algorithm can no longer handle deadlocks, and the system is saturated, resulting in a large decrease in throughput (see boxplot and scatters in Figure 3.12).

Comparing different layouts, we can see that additional workstations (system C vs. A, B) can improve the throughput. The processing time at the workstations will not be a bottleneck, and each workstation will provide an extra buffer area for the system. Decreasing the drop-off points does not improve the throughput, as the operation area also decreases, and the system is more likely to saturate with the same number of robots. Our evaluation method performs equally well on the layout we tested.

The simulation results are usually scattered below the analytical predictions because deadlocks occur stochastically. If a deadlock is formed, the throughput observed in the simulation will be greatly reduced. Improving the deadlock-resolving algorithm can improve the mean of simulation throughput (Deadlock-resolving is beyond the scope of this study). Our prediction algorithm did not assume deadlocks (Assumption 1). Otherwise, there will be multiple bottlenecks in one congested area. Therefore, our prediction gives an upper bound for the simulation throughput.

Our method (considering non-congested cells and bottlenecks) can give accurate predictions for stable systems. Before the density is so high that the system becomes unstable, the M/G/1 throughput estimation is very close to the simulation results. (See black MG1-CQN curves in Figure 3.12). If the system is saturated, our M/G/1 approximated estimation is no longer accurate. We can conclude that the renewal-process approximation (RP-CQN) can give good predictions in light-traffic conditions. Congestion plays a large role in evaluation. As shown in Figure 3.12 (MG1-CQN vs. CF-CQN), our method greatly reduced the estimation error after introducing congestion.

In conclusion, if the robot number and the layout are given, assuming the system is not saturated and the deadlock deadlock-resolving algorithm is good enough, our evaluation method can accurately predict throughput. The estimation error is greatly reduced compared to the prediction given by CQN without congestion.

3.6 Summary

We present an evaluation model that can predict the throughput and traffic delay of a robotic warehouse (RSS in our setting). We approximated the total delay in a closed queueing network by blocking it with a transportation network. The key idea is to show that in a large system with many robots, the arrival at one cell is approximately Poisson. The Poisson property is an intrinsic characteristic of the system. This emerges from the dynamics of robots in large systems constantly splitting and merging. The arrival pattern approximates a Poisson distribution for one cell because the time a robot spends outside the cell is a mixture of travel times on different paths, and all the robots are identical.

We validated our assumptions and the method using simulation experiments. The simulation validated our congestion mechanism: the system can be divided into non-congested cells and congested areas inside congested areas. Usually, one cell is the bottleneck in each congested area while the other cells are buffers. The coefficient of variation observed in the simulation also validated the Poisson approximation. Compared to the benchmark (without considering congestion), our method greatly improved the prediction accuracy if the system is stable and the deadlock-resolving algorithm is powerful enough.

Our method can predict performance if we know the system is stable. However, it cannot predict whether the system is stable and can only be applied to systems with relatively light traffic. In future research, we need to analyze the mechanism behind the stability of the system. The link delay model developed also laid the foundation for optimizing operation strategies in the next chapter.

Chapter 4

Operation of Robotic Warehouses

4.1 Introduction

As mentioned in the Introduction's motivation section, the robotic warehouse operation is real-time, and the controller must be decentralized. To have an online method with a near-optimal solution, we developed an offline-online method to simultaneously do path-planning and workstation assignments for robots and parcels in a robotic warehouse. We use RSS (in Section 1.2) as our setting, but this method can be easily extended to RMFS. The main idea of our method was to find an approximate optimal steady-state operation pattern offline. Then, we assign parcels/robots to workstations and plan robot paths online in real time to create a traffic flow as close to the calculated optimal result as possible. Our method has three major benefits:

1. Decentralized online control with linear complexity: the method can be applied to large systems.
2. Close to a system-optimal solution: the robots cooperate trying to re-create the optimal traffic flow assignment.
3. Integrated problem: We solve the path-planning and task assignment simultaneously, so different parts of the systems work together.

Our numerical analyses validate our model assumptions through simulation data and identify scenarios where our algorithm performs poorly, notably those involving special layout structures and high robot densities. Furthermore, we introduce an evaluative indicator capable of assessing the efficacy of our integrated offline-online method without real-world or simulation-based testing. Our simulation results demonstrate an appreciable improvement in throughput, quantified at approximately 10% compared to zone-based assignment with cooperative shortest path planning.

In Section 4.2, we introduce the offline part of our method: How to find an optimal traffic assignment for the steady state. In Section 4.3, we show how to use the offline solution to operate the system online. In Section 4.4, we discuss the theoretical limits and benefits and develop an indicator to evaluate our method using historical data for existing systems. We summarized the findings and provided ideas for future research in Section 4.6

4.2 The Offline Algorithm: Finding an Approximated Optimal Network Flow

We will develop our method on the basis of RSS settings. Consider the system in Figure 4.1a. We will use the same notation for the system as in Section 3.3. The system description is already given in Section 3.3. Given the demand d_{D_k} for each workstation, we want to find the optimal traffic assignment to minimize the total cost of transportation.

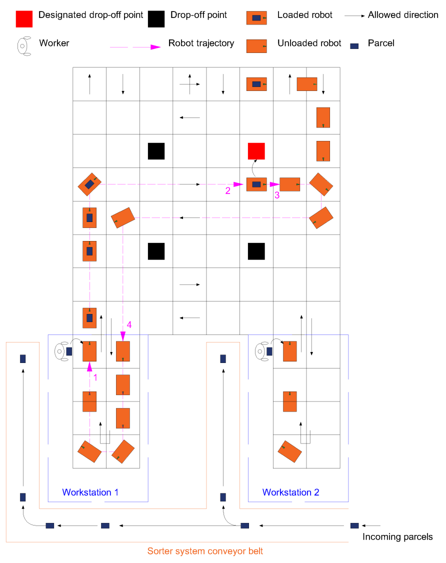
Recap: (M4) and Approximated Delay Estimator

Based on our analysis in Section 3.3, we can approximate the steady-state delay of the RSS using a transportation network (M4) (Figure 4.1b.)

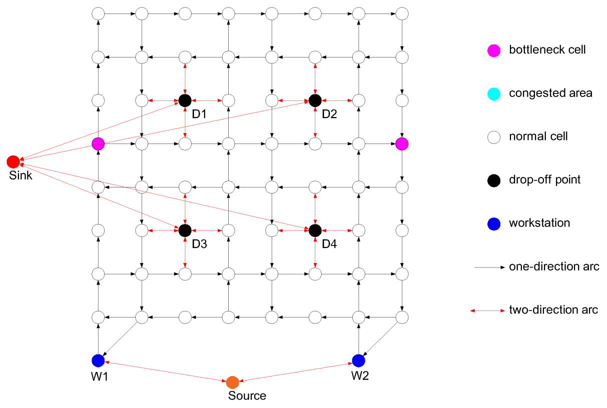
The operations (path, robot-to-workstation, and parcel-to-workstation assignment) under the steady states are represented by the path-flow traffic assignment in (M4). Consider the flow routes r from S to T for forward flow or T to S for backward flow. In a steady state, if we know all the forward and backward flows f_r^F and f_r^B, we know how the system assigns parcels and robots to the workstations and all the robot paths. The meaning of f_r^F is that for a path $r = (S, W_i, C_m, \ldots, C_n, D_j, T)$, there are f_r^F parcels loaded onto robots at workstation W_i and dropped at D_j per hour, while the robot carrying these parcels will follow the path $(C_m, \ldots, C_n) \subset r$. For f_r^B, $r = (T, D_j, C_n, \ldots, C_m, W_i, S)$, there are f_r^B robots returning from D_j to W_i following the path $(C_n, \ldots, C_m)$ per hour.

Therefore, given all f_r^F, in a steady state, the probability of assigning one parcel with destination D_j will be assigned to the workstation W_i with probability $\frac{\sum_{r:W_i \in r, D_j \in r} f_r^F}{\sum_{r:D_j \in r} f_r^F}$. The probability that the robot will carry this package using path $r_1 := (C_m, \ldots, C_n)$ from W_i to D_j is $\frac{\sum_{r:W_i \in r, D_j \in r, r_1 \subset r} f_r^F}{\sum_{r:W_i \in r, D_j \in r} f_r^F}$. Given f_r^B, the probability of assigning an empty robot to the workstation W_i, who just released the parcel at C_n near D_j, is $\frac{\sum_{r:W_i \in r, D_j \in r, C_n \in r} f_r^B}{\sum_{r:D_j \in r, C_n \in r} f_r^B}$. The probability of using the path $r_2 := (C_n, \ldots, C_m)$ is $\frac{\sum_{r:W_i \in r, D_j \in r, C_n \in r, r_2 \subset r} f_r^B}{\sum_{r:W_i \in r, D_j \in r, C_n \in r} f_r^B}$.

Given f_r^F and f_r^B, using the link delay from Equation (3.4) and (3.6) for cell-to-cell links, and imposing $c_{i,j}^F$, $c_{i,j}^B$ for forward and backward flow on the link connecting the source and drop-off points (see the description of (M3) in Section 3.3) to avoid unwanted flow directions. The total transportation cost (in machine time per hour) can be calculated using Equations (3.8), (3.9), and (3.10). Here, we recap the link delay, path cost, and total cost functions.



(a) RSS layout



(b) Transportation network (M4)

Figure 4.1: RSS model

link delay:

$$c_{i,j} = \begin{cases} \frac{v_j}{2}\mathbb{E}[G_{ij}^2] & \text{if } \frac{v_j^2\mathbb{E}[G_j^2]}{2(1-v_j\mathbb{E}[G_j])} + v_j\mathbb{E}[G_j] \le 1 \\ \frac{v_j\mathbb{E}[G_j^2]}{2(1-v_j\mathbb{E}[G_j])} & \text{otherwise} \end{cases},$$

transportation cost on path r:

$$\zeta_r(\mathbf{f}) = \sum_{(i,j) \text{ on path } r} c_{i,j}(\mathbf{f}) + \sum_{(i,j),(j,k)\in r} T_2\delta_{ijk}^{\text{turn}} + \sum_{(i,j)\in r} T_1,$$

system total transportation cost:

$$TC(\mathbf{f}) = \sum_{r\in\mathcal{R}_F} f_r^F \zeta_r(\mathbf{f}) + \sum_{r\in\mathcal{R}_B} f_r^B \zeta_r(\mathbf{f}).$$

Note that the total transportation cost is a non-linear function of path flows.

Optimal Traffic Assignment Problem

In steady state, the approximated optimal path-flow-based integrated assignment-routing problem (PB-IARP) can be formulated as

$$\begin{aligned}
\text{(PB-IARP)}\quad & \min\ TC(\mathbf{f}) \\
\text{s.t.}\quad & \text{Equations (3.1), (3.2), (3.3), (3.7), (3.8), (3.9), and (3.10) hold and} \\
& d_{D_k} = \sum_{r\in\mathcal{R}_F: D_k\in r} f_r^F, \forall \text{ drop-off point } D_k \qquad (4.1)\\
& d_{D_k} = \sum_{r\in\mathcal{R}_B: D_k\in r} f_r^B, \forall \text{ drop-off point } D_k \qquad (4.2)\\
& \sum_{r\in\mathcal{R}_B: W_k\in r} f_r^F = \sum_{r\in\mathcal{R}_B: W_k\in r} f_r^B, \forall \text{ workstation } W_k \qquad (4.3)
\end{aligned}$$

The constraints (4.1) and (4.2) state that the total forward and backward flow should satisfy the parcel demand for any drop-off points, where d_{D_k} is the demand rate for drop-off point D_k. In real-world problems, since we do not know the real throughput capacity of the system, we need a rough estimate of the demand and search for a demand level that matches the number of robots, as is discussed in the next sections (Algorithm 2). Constraints (4.3) state that each workstation's forward and backward flow must be balanced.

Link-Flow Based Formulation

Since there are exponentially many paths, the problem (PB-IARP) has many variables, making it difficult to solve. We will re-organize the transportation network in (M4) and use link-flow as the variables.

Note that the delay function Equation (3.8) has only link-flow variables; the path-flow variables are introduced in Equation (3.9) to include the turning time T_2. The turning flow information can be retained if we use the re-organized network shown in Figure 4.2.

We can decompose and re-index nodes in (M4). Each cell C_j in (M4) can be decomposed into four nodes: $4j, 4j-1, 4j-2, 4j-3$. Each of the new nodes represents one of the cell's four headings.

We add in-cell arcs between $4j-i$ and $4j-k$ to represent possible turning from direction i to k, and set the transportation cost as T_2. Note that, at most, two nodes in each cell are

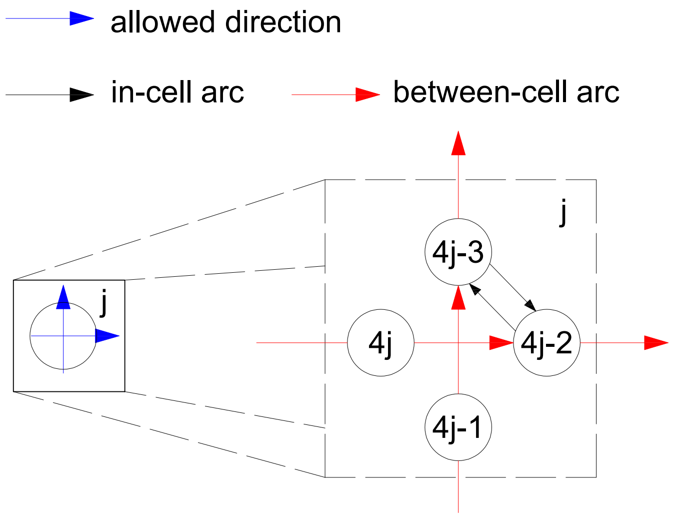


Figure 4.2: Cell node decomposition

connected since one cell only allows two moving directions (see Figure 4.1a). The movement from cell C_i to C_j is represented by between-cell arcs $4i - k$ to $4j - k$, where k is the corresponding heading direction for this movement, and the transportation cost on the between-cell arc is T_1.

Using $u_{p,q}$ to represent the link flow in the decomposed (M4), the task-specific flow v_j^k in Equation (3.1), (3.2), the link flow $v_{i,j}$ are:

$$v_j^{turn} = u_{4j-k_1,4j-k_2} + u_{4j-k_2,4j-k_1} \tag{4.4}$$

$$v_j^{through} = u_{4j-k_1,4i-k_1} + u_{4j-k_2,4m-k_2} \tag{4.5}$$

$$v_j^{drop} = u_{4j-k_1,4n} + u_{4j-k_2,4p} \tag{4.6}$$

$$v_j^{pick} = v_j^{turn} + v_j^{through} = v_j \text{ if } C_j \text{ is a picking location} \tag{4.7}$$

$$v_{j,i} = u_{4j-k_1,4i-k_1} \tag{4.8}$$

where k_1, k_2 be the two allowed direction on C_j, C_i and C_m are the two downstream neighbors of C_j, C_n and C_p are drop-off points connected to C_j (if exists, otherwise set flow to 0). For each type of flow (forward and backward), Equations (4.4) to (4.8) hold, so we omitted the upper index F and B in these equations. $u_{p,q} = u^F_{p,q} + u^B_{p,q}$ where $u^F_{p,q}$ and $u^F_{p,q}$ are forward and backward flow on the decomposed network. Therefore, the delay $c_{i,j}$ in Equation (3.8) is a nonlinear function of $\mathbf{u}$, and the (PB-IARP) can be reformulated as the link-based integrated assignment-routing problem (LB-IARP):

$$\text{(LB-IARP)}\ \min_{\mathbf{u}} TC(\mathbf{u}) := \sum_{i,j:C_i,C_j \text{are neighbors}} (c_{i,j}(\mathbf{u}) + T_1)v_{i,j} + \sum_{i:C_i \text{is a cell}} T_2 v_i^{turn} \tag{4.9}$$

$$\text{s.t. Equations (3.1), (3.2), (3.7), (3.8), and (4.4)–(4.8) hold and}$$

$$d_{D_k} = \sum_{j:C_j \text{are neighbor of } D_k} u^F_{4j-dir(j,D_k),D_k}, \forall \text{ Drop-off point } D_k \tag{4.10}$$

$$d_{D_k} = \sum_{j:C_j \text{are neighbor of } D_k} u^B_{D_k,4j-dir(j,D_k)}, \forall \text{ Drop-off point } D_k \tag{4.11}$$

$$0 = \sum_{i:\exists(i,j)\in \text{Decomposed (M4)}} u^F_{i,j}, \forall j \text{ not in source or sink} \tag{4.12}$$

$$0 = \sum_{i:\exists(i,j)\in \text{Decomposed (M4)}} u^B_{i,j}, \forall j \text{ not in source or sink} \tag{4.13}$$

$$u^F_{4j-dir(j,W_k),W_k} = u^B_{W_k,4j-dir(j,W_k)} \forall \text{ Workstation } W_k \tag{4.14}$$

where $dir(j, D_k)$ the the direction on C_j which robots are allowed to drop parcel to D_k, and $dir(j, W_k)$ is the direction allowing robots to be picked by W_k on C_j. Equations (4.10) and (4.11) state the forward and backward demand-flow balance for each drop-off point. Equations (4.12) and (4.13) are the flow conservation for each node. Equation (4.14) is the forward-backward flow balance at each workstation.

(PB-IARP) and (LB-IARP) solves the same problem, but (LB-IARP) has much fewer variables. (LB-IARP) is a nonlinear programming with linear constraints, which can be solved by the Frank-Wolfe method.

Frank-Wolfe Method

If we have a feasible solution $\mathbf{u}^{(iter)}$, the Frank-Wolfe algorithm can solve the following linearized subproblem (L-SUB) for (LB-IARP):

$$\text{(L-SUB)}\ \min_{\mathbf{u}}\ TC(\mathbf{u}^{(iter)}) + \frac{\partial TC}{\partial \mathbf{u}^{(iter)}}(\mathbf{u} - \mathbf{u}^{(iter)}) \tag{4.15}$$

$$\text{s.t. Equations (3.1), (3.2), (3.7), (3.8), (4.4)–(4.8), and (4.10)–(4.14) hold}$$

(L-SUB) is a linear programming which can be easily solved. The gradient for the piecewise function Equation (3.8) is the gradient at the current value. The gradient at a discontinuous point is defined as the left derivative. Given an error tolerance ϵ, the Frank-Wolfe method includes the following steps:

Frank-Wolfe Method (FWM):

1. Find the free-flow shortest path for all source-workstation pairs. Assign flow $\mathbf{u}^{(0)}$ using free-flow shortest path assignment. Set $iter = 0$.

2. Solve the linear subproblem (L-SUB) to obtain the search direction $\mathbf{u}_{\text{linear}}^{(iter)}$.

3. Do a line search from $\mathbf{u}^{(iter)}$ to $\mathbf{u}_{\text{linear}}^{(iter)}$ to find the optimal objective value $TC^{(iter)}$ and step size $\alpha^{(iter)}$. Update the flow assignment $\mathbf{u}^{(iter+1)} = \mathbf{u}^{(iter)} + \alpha^{(iter)}(\mathbf{u}_{\text{linear}}^{(iter)} - \mathbf{u}^{(iter)})$. $iter = iter + 1$.

4. Convergence check: if $|TC_{\text{linear}}^{(iter)} - TC_{\text{linear}}^{(iter-1)}| < \epsilon$ stop, otherwise, go back to step 2.

We can solve the (LB-IARP) and find a link-flow traffic assignment result, but we still need a path-flow assignment for the online algorithm. The path flow can be reconstructed using the algorithm proposed in [29] or the greedy algorithm mentioned in this study. We do not need an optimal flow decomposition. One acceptable algorithm is the path-length greedy algorithm. Each time, the greedy algorithm picks the shortest path from the source to the sink (or from the sink to the source) and pushes flow until one arc reaches its link flow. It removes the critical link and recurses. The greedy algorithm has linear time complexity $O(N)$ in a grid-based system, where N is the number of cells. The greedy algorithm's number of paths is also $O(N)$. This means the search space of the online algorithm is $O(N)$ if we use the path-length greedy algorithm.

Binary Search to Match the Number of Robots

To solve (PB-IAR), demand parameters d_{D_k} in Equation (4.1) and (4.2) must be estimated. However, d_{D_k} is the actual fulfilled demand in the system, and we don't know it without evaluating the system throughput.

Although the actual d_{D_k} is not known, usually, the ratio $d_{D_{k1}}/d_{D_{k2}}$ is known. We can start with a normalized demand vector $\tilde{d}_{D_k}$ with $\sum_{D_k} \tilde{d}_{D_k} = 1$, and try to find γ such that $\gamma \tilde{d}_{D_k} = d_{K_k}$

If the demand d_{D_k} is given, after finding the optimal flow, R, the number of robots in the system can be estimated using Little's law:

$$R = \sum_{r \in \mathcal{R}} \zeta_r(\mathbf{f}) f_r. \tag{4.16}$$

Therefore, we can search for a proper demand scale γ such that the estimated number of robots in the queuing system matches the actual robot number R. The number of robots required is a non-decreasing function in γ: $R(\gamma) = \sum_{r \in \mathcal{R}} \zeta_r(\mathbf{f}^*(\gamma)) f_r^*(\gamma)$, where $\mathbf{f}^*(\gamma)$ is the optimal flow assignment when the demand is $\gamma \tilde{d}_{D_k}$, because a greater demand leads to heavier traffic flow thus higher robot transportation cost. We can pick a small γ_0 and a large γ_1 such that $R(\gamma_0) < R < R(\gamma_1)$, and find an approximate demand level γ^* such that $R(\gamma^*) = R$ using binary search between γ_0, γ_1. The binary search algorithm is given in Algorithm 2.

Algorithm 2 Binary Search to Solve $R(\gamma) = R$

```
1: procedure BinarySearch(Function $R(\gamma)$ as Equation (4.2), $\gamma_{\text{low}}, \gamma_{\text{high}}, \epsilon$)
2:     while $\gamma_{\text{high}} - \gamma_{\text{low}} > \epsilon$ do
3:         $\gamma_{\text{mid}} \leftarrow (\gamma_{\text{low}} + \gamma_{\text{high}})/2$
4:         Estimate $R(\gamma_{\text{mid})})$ by solving the (LB-IARP) using the Frank-Wolfe Method.
5:         if $R(\gamma_{\text{mid}}) = R$ then
6:             return $\gamma_{\text{mid}}$
7:         else if $(R(\gamma_{\text{mid}}) - R) \times (R(\gamma_{\text{low}}) - R) < 0$ then
8:             $\gamma_{\text{high}} \leftarrow \gamma_{\text{mid}}$
9:         else
10:            $\gamma_{\text{low}} \leftarrow \gamma_{\text{mid}}$
11:        end if
12:    end while
13:    return $(\gamma_{\text{low}} + \gamma_{\text{high}})/2$
14: end procedure
```

Summary of the Offline Algorithm

The input of the offline algorithm are:

- The system roadmap and robot parameters
- Number of robots R
- Destination normalized demand $\tilde{d}_{D_k}$
- Searching region γ_0, γ_1

Based on the input, we can run the binary search Algorithm 2 to find the expected fulfilled demand d_{D_k}. With the demand d_{D_k}, we can find an approximated optimal steady-state link-flow assignment on the decomposed (M4), i.e. the (LB-IARP) using the Frank-Wolfe algorithm. The link flows can be decomposed into path flows using algorithms in [29]. The output of the offline algorithm is a vector of an optimal path flow $\mathbf{f}$ that can minimize total transportation cost under steady state.

4.3 The Offline-Online Algorithm

Using the offline method in the previous section, we obtain a vector of path flow $\mathbf{f} = [f_r], r \in \mathcal{R}$. The main idea of the online phase is to select paths according to the optimal flow so that the system has a similar steady-state flow distribution as the offline solution. The flow chart of the offline-online algorithm is given in Figure 4.3.

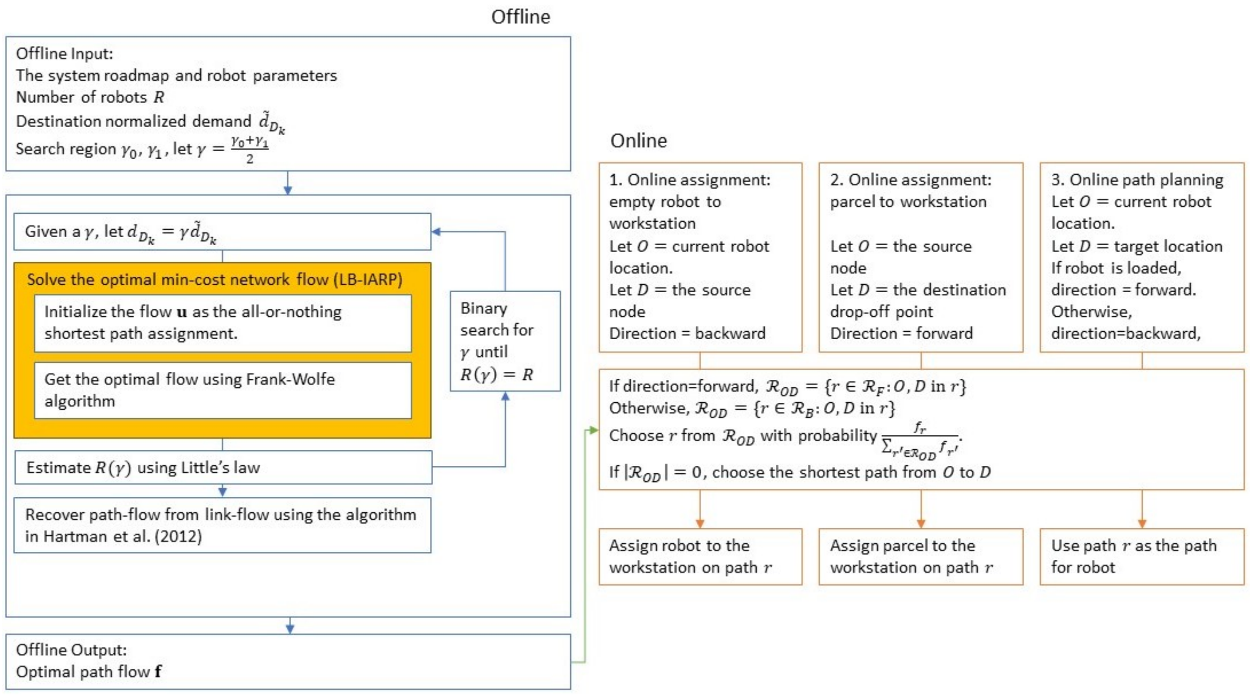

Offline
Offline Input:
The system roadmap and robot parameters
Number of robots R
Destination normalized demand $\tilde{d}_{D_k}$
Search region γ_0, γ_1, let $\gamma = \frac{\gamma_0+\gamma_1}{2}$
Given a γ, let $d_{D_k} = \gamma \tilde{d}_{D_k}$
Solve the optimal min-cost network flow (LB-IARP)
Initialize the flow $\mathbf{u}$ as the all-or-nothing shortest path assignment.
Get the optimal flow using Frank-Wolfe algorithm
Binary search for γ until $R(\gamma) = R$
Estimate $R(\gamma)$ using Little's law
Recover path-flow from link-flow using the algorithm in Hartman et al. (2012)
Offline Output:
Optimal path flow $\mathbf{f}$
Online
1. Online assignment: empty robot to workstation
Let O = current robot location.
Let D = the source node
Direction = backward
2. Online assignment: parcel to workstation
Let O = the source node
Let D = the destination drop-off point
Direction = forward
3. Online path planning
Let O = current robot location.
Let D = target location
If robot is loaded, direction = forward.
Otherwise, direction=backward,
If direction=forward, $\mathcal{R}_{OD} = \{r \in \mathcal{R}_F : O, D \text{ in } r\}$
Otherwise, $\mathcal{R}_{OD} = \{r \in \mathcal{R}_B : O, D \text{ in } r\}$
Choose r from $\mathcal{R}_{OD}$ with probability $\frac{f_r}{\sum_{r' \in \mathcal{R}_{OD}} f_{r'}}$.
If $|\mathcal{R}_{OD}| = 0$, choose the shortest path from O to D
Assign robot to the workstation on path r
Assign parcel to the workstation on path r
Use path r as the path for robot

Figure 4.3: Flow chart of the offline-online algorithm

When operating an online system, we must deal with three problems in real-time: assign an empty robot to a workstation, assign a parcel to a workstation, and find a path for the robot. We can define an origin-destination (OD) pair for each task:

- For robot-to-workstation assignment, the origin is the robot's current location, and the destination is the "source node". The path direction is backward.
- For a parcel-to-workstation assignment, the origin is the source node, and the destination is the node representing its destination drop-off point. The path direction is forward.
- For robot path-finding, the origin and destination are the OD locations of the current robot. The path direction depends on the status of the robot. If the robot is loaded with a parcel, the direction is forward. Otherwise, the direction is backward.

Based on the OD pair and the direction, we choose route r for each task. If the task is forward, then the probability of choosing r is $\frac{f_r^F}{\sum_{r' \in \mathcal{R}_F:\text{O, D on } r'} f_{r'}^F}$. For the backward task, the probability of choosing a path r is $\frac{f_r^B}{\sum_{r' \in \mathcal{R}_B:\text{O, D on } r'} f_{r'}^B}$. (See more examples in Section 4.2, Recap of (M4), paragraph "Therefore, given...") If there is no known path between OD, this method is not suitable, and we use the shortest (in free-flow travel time) path instead. Fortunately, this is rarely the case in our algorithm since robots only need to decide their path near workstations and drop-off points, and most of the workstation-dropoff point paths have been included in the offline algorithm.

For an assignment task, after randomly assigning a path to each task according to the flow intensity, the assignment result can be recovered by identifying the workstation node along the path. Due to our definition of $c_{i,j}$ for special arcs (see the definition below Equation (3.8)), each path with finite cost can go through workstations at most once. Therefore, the workstation node on the path is unique, and we can use this workstation as the assignment result. For a path-finding task, we can use the chosen path as the path plan for the robot.

4.4 Discussions

Limits of the Algorithm: When the Assumptions Fail and a Diagnostic Indicator

As is shown in the numerical experiments in Section 3.5 when analyzing the congested areas, there are two causes for large estimation error:

- Tandem bottlenecks: if the departure robots from one bottleneck almost all arrived at the second bottleneck, then these robots' arrivals are correlated, and our Assumption 3 failed.
- Congested systems: if the system is saturated, there are multiple bottlenecks in one congested area, and our Assumption 1 failed.

The solution to the first problem is to manually identify these tandem bottlenecks and merge them as one congested area, using the bottleneck with the highest utilization ρ_i as the actual bottleneck. The second problem is not common because companies tend to avoid putting too many robots in one system.

Therefore, our offline-online method is limited to certain layouts and robot numbers. We will introduce a diagnostic indicator to help decide if our algorithm is good for specific systems without running the offline-online algorithm.

Let the total cost of Equation (3.10) be TC, and the true total cost be TTC. TC and TTC are functions of the traffic flow distribution $\mathbf{f}$. TC is the approximation of TTC using our (M4) model.

Without running our offline-online method, a robotic warehouse company can accumulate historical operational data, including the observed robot flow $\mathbf{f}$ and the actual transportation cost $TTC(\mathbf{f})$. These results are based on their current operational strategies, not our method.

Our offline algorithm can find an optimal $\mathbf{f}^*$ that can minimize TC. Define relative approximation error as

$$ERR := \frac{(TC(\mathbf{f}) - TTC(\mathbf{f}))}{TTC(\mathbf{f})} \tag{4.17}$$

and relative optimal gap

$$GAP := \frac{(TC(\mathbf{f}) - TC(\mathbf{f}^*))}{TC(\mathbf{f})} \tag{4.18}$$

ERR comes from the deviation from our assumptions and asymptotic results. GAP is the power of our offline optimization, which shows how much our offline algorithm can improve from the benchmark solution. Note:

$$TTC(\mathbf{f}^*) = \frac{TTC(\mathbf{f})(1 - GAP(\mathbf{f}))(1 + ERR(\mathbf{f}))}{(1 + ERR(\mathbf{f}^*))}. \tag{4.19}$$

Therefore, if $(1 - GAP(\mathbf{f}))(1 + ERR(\mathbf{f}))/(1 + ERR(\mathbf{f}^*)) > 1$, our offline algorithm's improvement outperforms the estimation error. Note that our $ERR < 0$ because we ignore some robot blockings in our Assumptions 1 and 5, so we underestimate the total travel time. We have

$$\frac{(1 - GAP(\mathbf{f}))(1 + ERR(\mathbf{f}))}{(1 + ERR(\mathbf{f}^*))} > (1 - GAP)(1 + ERR(\mathbf{f}))$$

. We can define the estimated *improvement bound* (IB) as

$$IB := (1 + ERR(\mathbf{f}))(1 - GAP(\mathbf{f})) \tag{4.20}$$

If $IB > 1$, our algorithm can outperform the benchmark even with the estimation error. Therefore, we can use IB as a diagnostic indicator to evaluate our offline-online method based on historical data.

Note that, to calculate $ERR(\mathbf{f})$ and $GAP(\mathbf{f})$, we only need $TTC(\mathbf{f})$, which is given in historical data, $TC(\mathbf{f})$, which can be estimated using historical flow observation and Equation 3.10, and $TC((\mathbf{f})^*)$, which is solved using the offline algorithm (Algorithm 2). All these calculations don't involve doing simulations or deploying our offline-online method in the real world. Therefore, the diagnostic indicator IB can be estimated based on historical observations.

Better Performance Compared with Priority Planning

We will explain why our method is better than our benchmark in throughput performance. One popular method for fast MAPF is prioritized planning (cooperative A* or CA*)[51]. When a new task arrives in the system, it searches for the fastest path without interrupting existing plans. More specifically, the prioritized planning method maintains a time-expansion graph. It makes T copies of the network, and every vertex represents a pair (i, t), where i is the node index and t is the time step. (i, t_1) and (j, t_2) are connected only if a robot in i at time t_1 can move to its neighbor j in $t_2 - t_1$. If one path is assigned to one robot, then all the nodes that the robot passes on the time-expansion graph will be blocked. When a new robot with lower priority arrives, it will search for the shortest path on the time-expansion graph without using blocked nodes. Usually, the newest task has the lowest priority, so the previously assigned paths will not change for computational simplicity. We can show that such an algorithm leads to a stochastic user equilibrium at a steady state.

Proposition 1. *CA* leads to a stochastic user equilibrium under a steady state.*

Proof. Let $\mathcal{R}$ be the set of all paths connecting one origin–destination pair. For path $r \in \mathcal{R}$, the total travel time is

$$c_r := \alpha_r T_1 + \beta_r T_2 + \sum_{(i,j):(C_i,C_j)\in r} S_{ij},$$

where α_r is the number of movements from one cell to another on r, β_r is the number of 90 degree turns on path r, and S_{ij} is the random waiting time when entering from C_i to C_j.

When planning a path using CA* for one robot, the other $R-1$ robots are already executing their paths, so the new path always has the lowest priority. This robot will choose the path with the lowest travel time. In a steady state, S_{ij} is a time-independent random variable determined by the flow distribution of the other $R-1$ robots, and the probability of choosing path r is

$$p_r = \mathbb{P}(c_r \le c_p : \forall p \notin r; p, r \in \mathcal{R}).$$

Therefore, the flow distribution will be a stochastic user equilibrium. □

Since our method seeks an approximate system optimum, whereas priority planning can only achieve stochastic user equilibrium, our method will be better if the approximation is relatively accurate. As mentioned in the previous section, we only need $IB > 1$.

4.5 Numerical Results

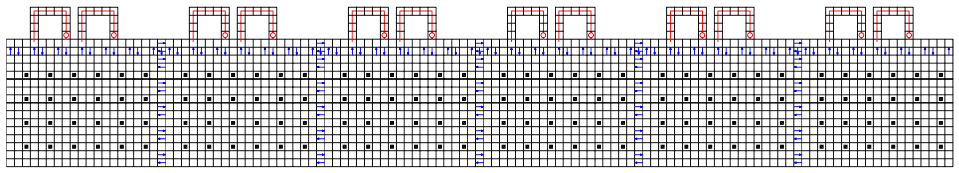
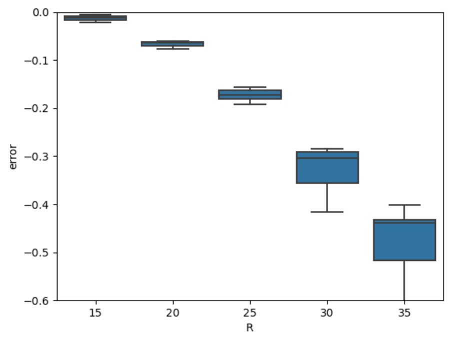


Figure 4.4: Large system map

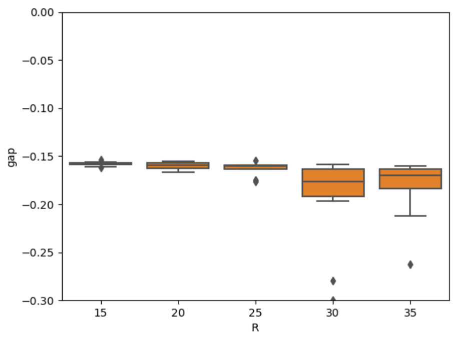
The simulation setting is the same as the experiments in Section 3.5. We also introduced a large system to validate our method in an industrial-level problem. The small system is the same as in Figure 3.7. The large system is shown in Figuire 4.4.

Validation of the Diagnostic Indicator

We will show that our estimation is accurate enough that the improvement from offline optimization can outperform the estimation error in our settings. We will also validate the indicator IB from Equation (4.20) to help decide if our algorithm is good for specific systems without running the offline-online algorithm.



(a) Relative error in predicting travel time



(b) Relative improvement of the optimal assignment

Figure 4.5: Accuracy of the objective function

R	ERR mean	ERR min	ERR max	GAP	IB
15	-0.013	-0.022	-0.006	-0.158	1.151
20	-0.066	-0.077	-0.060	-0.160	1.080
25	-0.174	-0.191	-0.157	-0.163	0.980
30	-0.325	-0.416	-0.285	-0.196	0.85
35	-0.482	-0.660	-0.402	-0.183	0.707

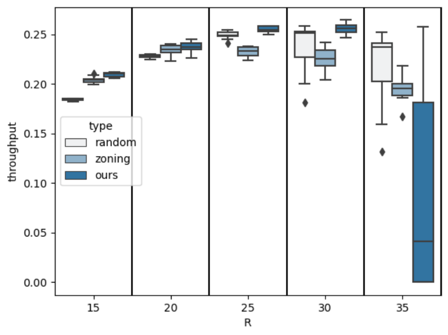
Table 4.1: Error, gap and improvement bound

For the small warehouse, we conducted simulations using the random assignment and the shortest path for each robot. We simulated 50,000 timesteps for each trail and 50 trails for each setting. We recorded flow vector $\mathbf{f}$ based on robot movement in simulations and calculated the estimated travel cost $TC(\mathbf{f})$ using Equation (3.10).

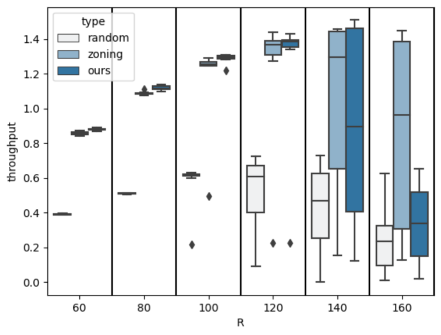
Let the true total cost TTC be the travel time calculated from the simulation. We calculated the output from our offline algorithm $\mathbf{f}^*$, using the throughput from each simulation experiment to minimize TC, and estimated ERR, GAP, and IB from Equations (4.17), (4.18), and (4.20). The distribution of ERR, GAP, and the estimated IB is shown in Figure 4.5 and Table 4.1.

Notice that $|ERR|$ increases as we add more robots. A larger R leads to higher density, more deadlocks, and complex queuing structures, so our assumptions deviate more from the real world. In addition, as mentioned in the previous section, some layout structures may lead to large estimation errors, thus poor performance of our offline-online algorithm. IB can be used as a diagnostic indicator to help evaluate our offline-online method before using the algorithm. We can also confirm that $ERR < 0$ because we ignore some robot blockings in our Assumptions 1 and 5.

The GAP is approximately the same when the density of the robot increases (around



(a) Throughput for different methods: a small system



(b) Throughput for different methods: a large system

Figure 4.6: Throughput for different settings

15%). According to estimated IB, when $R < 25$, the improvement provided by our offline algorithm outperforms the estimation error and, therefore, can guarantee a better solution. When $R = 25$, $IB \approx 1$. Since it is just a lower bound, it's still highly likely that our algorithm performs better. But for $R \geq 30$, we know without running our offline-online algorithm that it is likely that our algorithm cannot outperform the benchmark due to the large estimation error. As shown in the next section, Figure 4.6a, this result agrees with the actual performance of our algorithm.

Therefore, IB indicates how good our algorithm can be and how it can be calculated without running our offline-online method. Using IB calculated from historical data, the manager can choose our method if $IB > 1$.

Validation of the Offline-Online Method

We will show that our algorithm can improve system throughput. For the small and large systems (the concatenation of 6 small systems), we set the number of robots to different values and conducted 50 trials for each setting, with 3,000 steps for the large system and 10,000 steps for the small system. We compared our algorithm with the benchmark (random assignment + shortest path priority routing)[51] and zoning (divide the systems into zones and do random assignment + shortest path priority routing in each zone; each zone has one workstation and several drop-off points).

The throughput (TH) is given in Figure 4.6. We also recorded the average time steps (TS) that each algorithm can run without unsolvable deadlocks to show the stability of each method. For each algorithm, the average TH will increase and then decrease. As the density of the robot increases, the system becomes unstable and prone to deadlocks. Most deadlocks are solvable, but will increase the delay. In this experiment, we notice that R must be less than 30 for the small system and 120 for the large system to avoid too many deadlocks.

	random		zoning		offline-online		offline-online vs random		offline-online vs zoning	
R	TH	TS	TH	TS	TH	TS	ΔTH	ΔTS	ΔTH	ΔTS
60	0.3913	3,000	0.8587	3,000	0.8786	3,000	124.52%	0.00%	2.32%	0.00%
80	0.5107	3,000	1.0869	3,000	1.1176	3,000	118.84%	0.00%	2.82%	0.00%
100	0.5772	2560	1.1845	2718	1.2885	3,000	123.24%	17.21%	8.78%	10.37%
120	0.5204	1165	1.2508	2709	1.2664	2709	143.34%	132.60%	1.24%	0.00%
140	0.4243	454	1.0224	1856	0.8986	1311	111.79%	188.51%	-12.11%	-29.38%
160	0.2648	270	0.8646	1561	0.3411	130	28.83%	-51.85%	-60.55%	-91.68%
15	0.1841	10,000	0.2039	10,000	0.2092	10,000	13.66%	0.00%	2.59%	0.00%
20	0.2280	8159	0.2342	10,000	0.2372	10,000	4.05%	22.57%	1.29%	0.00%
25	0.2492	8270	0.2322	10,000	0.2548	10,000	2.24%	20.91%	9.72%	0.00%
30	0.2365	3128	0.2250	10,000	0.2556	10,000	8.07%	219.65%	13.59%	0.00%
35	0.216669	3025.2	0.19536	10,000	0.090655	298	-58.16%	-90.15%	-53.60%	-97.02%

Table 4.2: Performance improvement

R	simulation time per step		
	random	zoning	offline-online
15	2.05E-03	2.59E-03	1.35E-03
20	5.64E-03	3.50E-03	2.04E-03
25	8.90E-03	4.37E-03	2.82E-03
30	4.07E-03	3.29E-03	2.36E-03
60	9.85E-03	8.87E-03	5.57E-03
80	1.45E-02	1.30E-02	8.87E-03
100	6.19E-02	1.90E-02	1.31E-02
120	5.55E-02	2.64E-02	1.99E-02

Table 4.3: Computation time per simulation step in second

Therefore, a managerial insight is that no matter what assignment/path-finding method we use, we should check if complex congestion is common when the throughput is less than expected. If there are multiple bottlenecks in congested areas, we should decrease the number of robots. The marginal utility for each additional robot diminishes in an RSS and becomes negative if too many robots exist.

Another insight is that even with zoning, a large system cannot outperform many smaller systems because of the spread of congestion and the risk of deadlocks. The peak for the large system is about 1.27, and 0.25 for the smaller system $1.27 < 6 \times 0.25 = 1.5$. This is because the arrival rate of deadlocks in large systems is greater. A deadlock in one zone may spread congestion to zones, leading to system collapse. Therefore, with a large system, we should physically divide the system into smaller ones to deal with deadlocks and system collapse separately; zoning strategy alone is insufficient.

The relative improvement in performance Δ TH and the time steps before unsolvable deadlocks ΔTS are shown in Table 4.2. Before reaching the critical density, ΔTH will increase. After the critical density, we cannot guarantee that our algorithm is better, since the deadlock resolving contributes more to the delay, and the system becomes unstable when running.

Our discussion with [28] shows that the system usually runs with robot numbers near the peak throughput to achieve maximum efficiency. At peak throughput, our algorithm can increase throughput by approximately 10% in the small system (R=30) and the large system (R=100) compared to the zoning method, and much more than random assignment. Our algorithm also increases the stability since the time steps before hitting an unsolvable deadlock increase when using our method.

Finally, the computation time is greatly reduced since the online algorithm picks paths from $O(N)$ known paths, while the shortest path takes $O(N \log(N))$. We show the calculation time of one timestep for different algorithms in Table 4.3, using a personal laptop with an Intel(R) i9-12900HK processor and 16GB RAM with a simulator written in Python 3.8. Our algorithm is about twice as fast as the benchmark methods, since they need to find the shortest paths for each robot.

Conclusion

In conclusion, our experiments showed the following.

- The diagnostic indicator IB can predict if our method outperforms the current system based on historical data without running simulations with our offline-online method (See Figure 4.5 and Table 4.1).
- Our offline-online method can improve throughput by about 10% (See Figure 4.6 and Table 4.2)
- Our method has better computation time performance compared to existing methods (See Table 4.3), making it suitable for large systems with hundreds of robots.
- The marginal utility for each additional robot is diminishing in an RSS and will be negative if we have too many robots.
- If using a zoning strategy, instead of only using zoning for assignment, it would be better to disallow robots from moving from one zone to another to prevent queue spillover from one zone to another.

4.6 Summary

We present an integrated offline-online assignment and decentralized routing framework for grid-based robotic warehouse systems. Given the system layout, robot fleet, and estimated demand, our approach can dynamically and efficiently allocate paths to robots. Our offline algorithm employs a directed graph representation to approximate the steady-state, system-optimal flow distribution. Subsequently, this information guides the online component, achieving a computational complexity of $O(N)$. Thus, our framework is well-suited for large systems involving hundreds of robots and thousands of cells. Simulations indicate superior performance compared to zoning or random assignment schemes combined with prioritized shortest-path routing, yielding a throughput improvement of approximately 5-10%.

Nevertheless, our approach has its limitations. It is best suited for expansive systems featuring freely navigable robots. Systems characterized by elongated corridors, tandem bottlenecks, or exceedingly high robot density leading to complex blockages render our assumptions useless. Before deploying our algorithm, operators can estimate the IB indicator (discussed in Section 4.4)—calculated from historical data—to assess the algorithm's applicability to their specific settings.

There are several directions for future studies. Firstly, while our model's applicability to RMFS systems, comprehensive empirical validation at the industrial scale remains crucial to realizing its full potential. Secondly, we have touched upon the influence of battery consumption on fleet size, yet a more expansive model incorporating battery charging and swapping mechanisms warrants exploration. Lastly, strict proof of Conjecture 1 remains an open question; achieving this will necessitate a deeper comprehension of heavy-traffic approximations for departure and arrival flows in closed queuing systems.

Chapter 5

New Design for Robotic Warehouses

5.1 Introduction

To make fully automated RMFS with internal workstations possible, as mentioned in Section 1.3, we need to handle assembled orders at internal workstations. The main idea of our design is to pick up items and assemble order totes using robotic arms in internal workstations and temporarily store the totes in some special pods near these interior workstations. Once in a while, we dispatch a mobile robot to transport these totes from one internal workstation to one external workstation carrying the filled special pod. Using this design, items are batched at internal workstations, thus reducing transportation costs and improving the system's performance.

To determine the location of these internal and external workstations, we developed a location-allocation-queuing model to minimize the total transportation cost. This model can optimize workstation locations for both our and traditional (KIVA) designs. The system is modeled with assumptions as an open queueing network (OQN) that does not have a product-form solution. Assuming that the internal workstations have sufficient temporary storage capacities (i.e., the batch size of assembled totes is large), we approximated the waiting times of robots at workstations (internal and external) using closed-form functions of flow intensity, and the total transportation cost can be approximately estimated. We validated our approximation using numerical simulation and compared our new design with KIVA-like designs. Our experiments showed that our design can significantly decrease robot transportation and waiting costs for large and deep systems. Our experiment also suggested deploying fewer internal workstations if the processing time for special pods is long.

5.2 System Design

In this section, we will describe our design and recap the KIVA system.

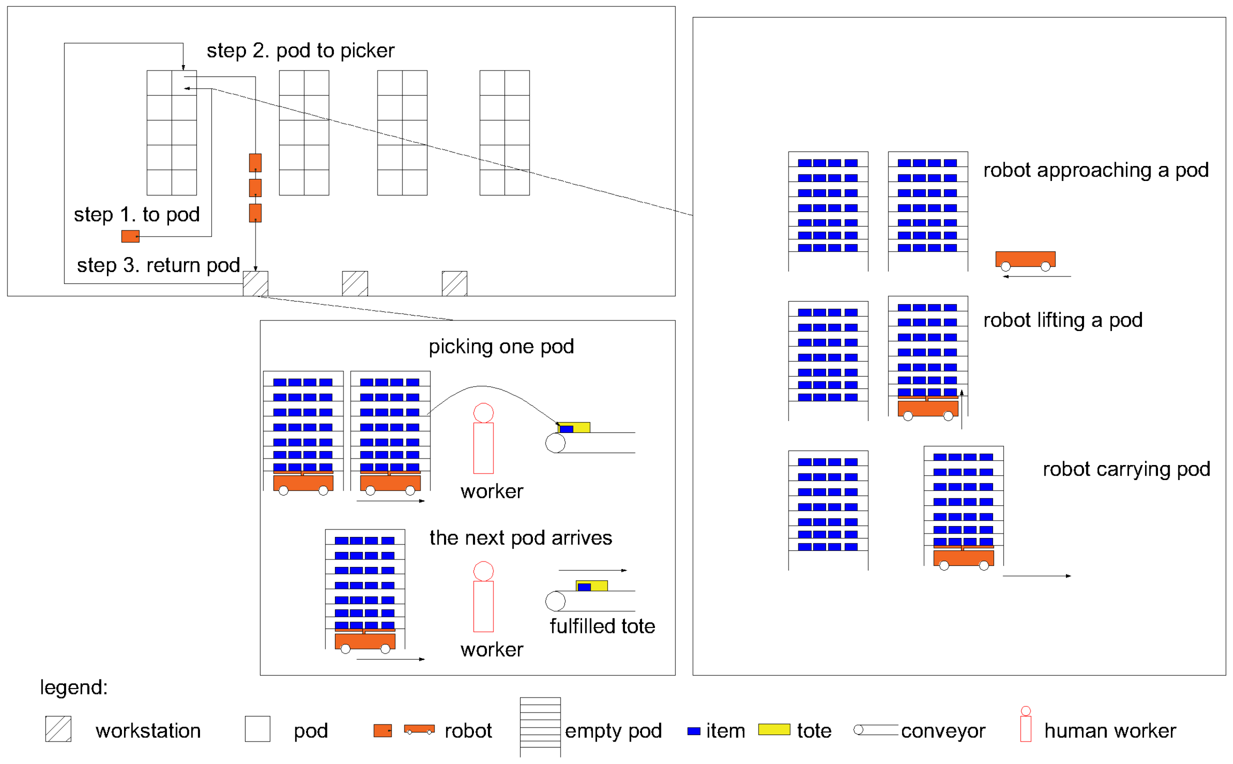


Figure 5.1: KIVA-like RMFS systems

Recap: KIVA RMFS

KIVA RMFS comprises mobile robots, moving pods, and picking workstations. Robots move pods from storage areas to workstations on the periphery. Human workers pick items from the pods at the workstations and place them in totes. Each tote is an assembled order and will exit the system via a conveyor for further packing and consolidating. The robots will also return the picked pods to the storage area. See Figure 5.1 and for more details, see Section 1.2.

New RMFS Design

In our fully automated RMFS with a grid-like roadmap, the items are stored in movable pods called *normal pods*. When a pick-up order arrives, a robot will move to the pod with the target item (Step 1 in Figure 5.2). The robot then lifts the pod and carries it to one of the workstations (step 2), where a picker can pick up the item from the pod (step 3). The robots will form a line near the workstation to be picked.

After picking, the pod will be returned to the storage area. If the workstation is located near a conveyor or the exit (*external workstations*), the totes with the picked items will be

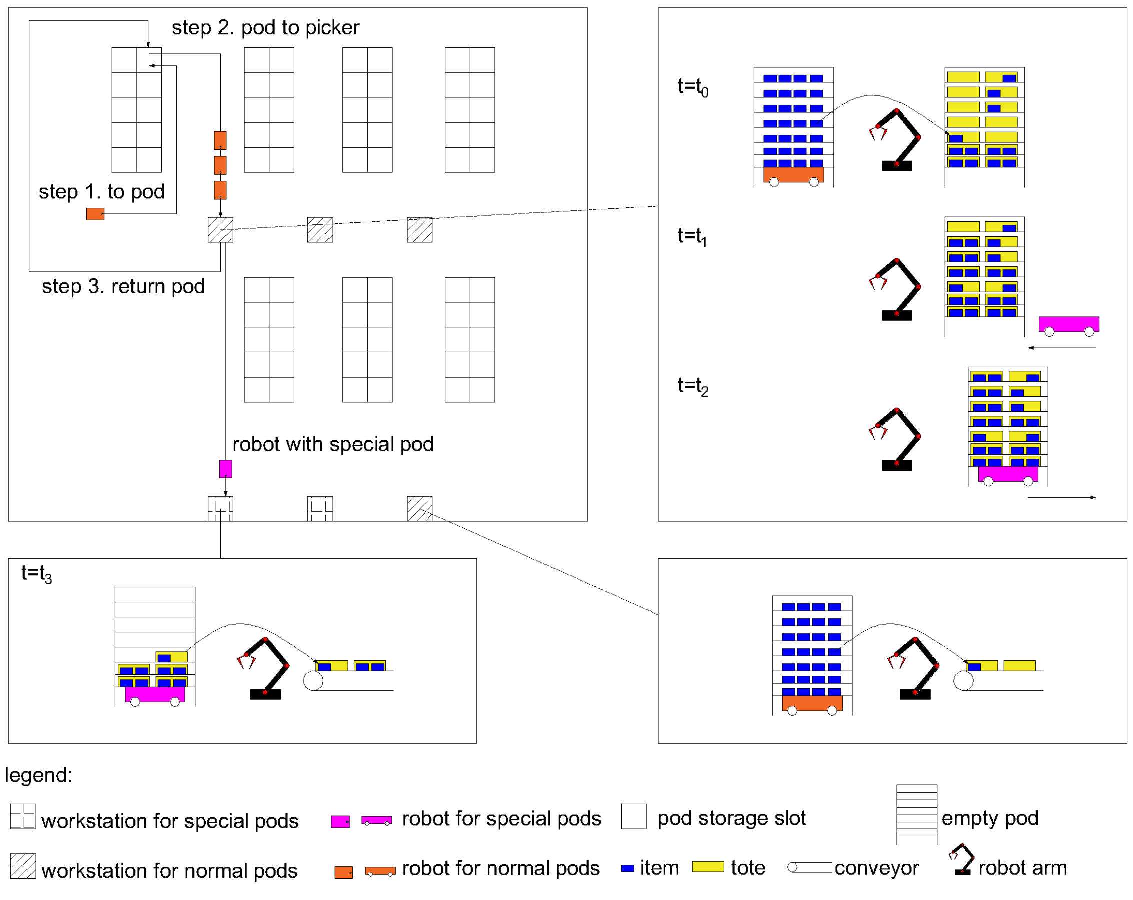


Figure 5.2: New RMFS design

shipped immediately after pick-up (lower right part of Figure 5.2). If the workstation is isolated in the storage area (*internal workstations*), picked totes will be temporarily stored in a *special pod* inside the workstation ($t = t_0$ in Figure 5.2). The special pod serves as a pool for completed totes: the empty totes on the pod will be filled over time ($t = t_0$ to $t = t_1$ in Figure 5.2). Every once in a while, an empty robot will come to the workstation and carry this special pod filled with picked totes to an external workstation near the conveyor or exit (see the robot with special pod (the purple robot) in Figure 5.2).

Our design can reduce transportation costs compared to the conventional KIVA system because the items from the storage area are batched at internal workstations, reducing the total travel distance. Using the special pod with totes, we also find a way to connect isolated internal workstations with external workstations.

5.3 Model

In this section, we analyze our design and model it as an open queueing network (OQN), so that we build a location-allocation-queueing model to find the optimal location of the workstation.

Assumptions

We make the following assumptions:

Assumption 6 (Steady State)**.** *The system operates in a steady state. All processes are stationary.*

Our analysis concerns the steady state of the system. In a real-world system, there could be occasional interruptions and fluctuations. However, in the strategic decision-making phase, like layout design, we can ignore these interruptions and focus on a stable and smooth-running state.

Assumption 7 (Poisson Arrival)**.** *The picking request for each pod is independently Poisson. The arrival rate depends only on the storage location of the pod.*

Assumption 8 (Enough Robots)**.** *There are enough robots in the system. A newly received order will be processed immediately.*

Assumption 7 and 8 are approximations for the warehouse management system (WMS). A real-world system has two levels of operation: WMS and robot management system (RMS). WMS will pre-process orders and split and batch orders to reduce the number of picking operations. RMS does not know order details; it only manages robots picking pods.

In an e-commerce system, most orders are single-line. Even if an order is not single-line, the WMS usually decomposes it into multiple single-line tasks and assembles them in the order consolidation process after picking them. The actual order arrival process is not Poisson. They usually arrive in batches, and the WMS would combine different orders so that they can be fulfilled by one pod-picking operation. In this process, WMS usually does not know the pod location (e.g., [33]), so location-wisely, pods containing the requested item have an equal probability of being chosen. In addition, the WMS's goal is to stabilize the system's operation, so it will not release orders if all robots or workstations are busy or congested, and it will release more orders if the system is idle.

From the RMS point of view, WMS introduces smoothness in the order arrival process and randomness in the requested pod location. We will treat this smooth and stable arrival as Poisson and assume that the WMS stores items randomly. If WMS is smart enough, it will never assign orders to busy robots, equivalent to, say, enough robots. Therefore, we approximate robot operation using assumptions 7 and 8.

Assumption 9 (Collision-free Path)**.** *The robot travel time is deterministic.*

We ignore robot blocking in most parts of the system. Congestion is modeled as queues near workstations, and the robot travel time is assumed to be deterministic. In a fully automated real-world RMFS, a workstation is much more expensive than mobile robots or moving pods, so we let robots queue near workstations. Consequently, most traffic delays occur in queues near workstations instead of intersections or aisles.

Open Queuing Network

RMFS as an OQN

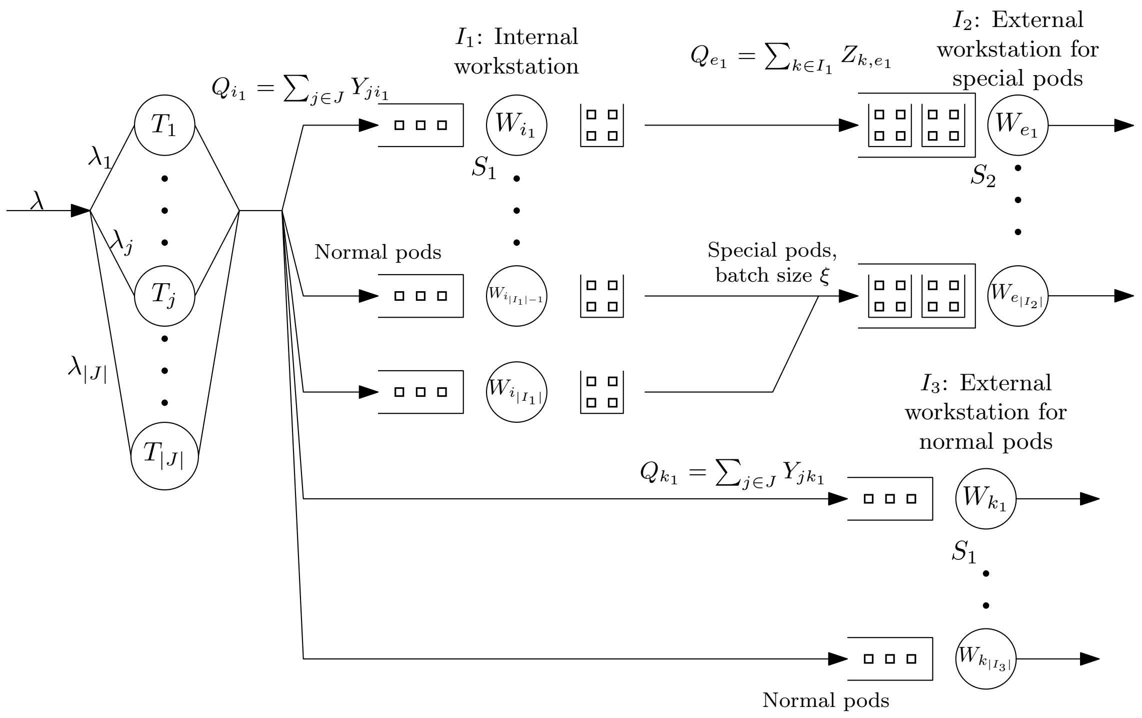

I_1: Internal workstation
$Q_{e_1} = \sum_{k\in I_1} Z_{k,e_1}$
I_2: External workstation for special pods
$Q_{i_1} = \sum_{j\in J} Y_{ji_1}$
T_1
λ_1
λ
λ_j
T_j
$\lambda_{|J|}$
$T_{|J|}$
W_{i_1}
S_1
W_{e_1}
S_2
Normal pods
Special pods, batch size ξ
$W_{i_{|I_1|-1}}$
$W_{e_{|I_2|}}$
$W_{i_{|I_1|}}$
I_3: External workstation for normal pods
$Q_{k_1} = \sum_{j\in J} Y_{jk_1}$
W_{k_1}
S_1
$W_{k_{|I_3|}}$
Normal pods

Figure 5.3: OQN for fully robotic RMFS

In an RMFS with internal workstations, let $\mathcal{T} := \{T_j : j \in J\}$ be the set of storage slots for pods, each T_j is represented by a small pod square in Figure 5.2). J is the index set of all pods. Let the set of workstations be $\mathcal{W} := \{W_i : i \in I\}$ with the index set $I := \{1, 2, 3, ...\}$. $I = I_1 \cup I_2 \cup I_3$ where $\{W_i : i \in I_1\}$ is the set of internal workstations, $\{W_i : i \in I_2\}$ is the set of external workstations that process special pods, and $\{W_i : i \in I_3\}$ is the set of external workstations that process normal pods.

Let $Y_{j,i}$ be the flow of normal pods from the storage slot $T_j : j \in J$ to the workstation $W_i : i \in I_1 \cup I_3$. Let $Z_{k,i}$ be the flow of special pods from $W_k : k \in I_1$ to $W_i : i \in I_2$, and let Q_i be the incoming flow to $W_i : i \in I$.

Let ξ be the tote capacity of a special pod. Let the random service time of a normal pod and a special pod at the workstation be S_1 and S_2, respectively. Based on our assumptions, each internal workstation W_i should only send out one special pod when it is filled.

Proposition 2. *Under Assumption 8, sending special pods with ξ filled totes is optimal in transportation cost.*

Proof. If internal workstation W_i sends N special pods with $X_1, X_2, ..., X_n, ..., X_N$ totes, and $\exists X_n < \xi$. We can set $X_i = \xi, X_N = X_N - (\xi - X_i)$ for $i = 1, 2, ...$ until $X_N = 0$, which means the task can be completed using $N-1$ special pods travel with less. Repeating this process, to reduce transportation cost, $X_1, X_2, ...$ should be ξ. □

Based on our Assumptions 6 and 8, the RMFS with internal workstations can be modeled as the open queueing network (OQN) shown in Figure 5.3. Each T_j node is a pod storage location in the network, and each W_i node is a workstation. The arrival of order at T_j is a Poisson flow λ_j. T_j will dispatch normal pods to $W_i : i \in I_1 \cup I_3$ with Poisson flow $Y_{j,i}$.

If $i \in I_3$ (for example, W_{k_1} in Figure 5.3), W_i is an external workstation that can process robots that carry normal pods and send completed totes out of the system. Otherwise, if $i \in I_1$ (for example, W_{i_1} in Figure 5.3) , W_i is an internal workstation. $W_i : i \in I_1$ processes the robots in the queue and stores the totes in a special pod as a batching pool. The service time for $W_i : i \in I_1 \cup I_3$ is S_1 per pod and the service rate $\mu_i = \frac{1}{\mathbb{E}[S_1]}$. The arrival rate of the pods at $W_i : i \in I_1 \cup I_3$ is

$$Q_i = \sum_{j \in J} Y_{j,i} : i \in I_1 \cup I_3 \tag{5.1}$$

The internal workstations process the robots in the queue and store totes in a special pod that serves as a batching pool. $W_k : k \in I_1$ is assigned to $W_i : i \in I_2$, an external workstation designed for a special pod. The flow of special pods from $W_k : k \in I_1$ to $W_i : i \in I_2$ is $Z_{k,i}$.

If $i \in I_2$, W_i processes special pods from $W_k : k \in I_1$ with random service time S_2 and service rate $\mu_i = \frac{1}{\mathbb{E}[S_1]}$ (for example, W_{k_1} in Figure 5.3). The arrival rate at $W_i : i \in I_2$ is $Q_i = \sum_{k \in I_1} Z_{k,i}$. Based on Proposition 2, $Z_{k,i} = \frac{Q_k}{\xi}$. The external workstation $W_i : i \in I_2$ connected to W_k will process the special pod and send the completed totes out of the system. The arrival rate at $W_i : i \in I_2$ is

$$Q_i = \sum_{k \in I_1} Z_{k,i} : i \in I_2 \tag{5.2}$$

Each $W_i : i \in I$ can be modeled as a FCFS G/G/1 queue, with random service time S_1 for $W_i : i \in I_1 \cup I_3$ and S_2 for $W_i : i \in I_2$. We can use Kingman's formula to estimate queueing delays. Therefore, we must first estimate the expectation and CV of the inter-arrival times on each $W_i : i \in I$.

Estimating CV of Inter-Arrival Times at Workstations

The order arrival of an internal workstation $W_i : i \in I_1 \cup I_3$ comes from pods $T_j : j \in J : Y_{j,i} > 0$, that is a combination of many Poisson flows. Therefore, the arrival process at W_i is also Poisson, and the CV of inter-arrival times is 1.

The order arrival of an external workstation $W_i : i \in I_2$ comes from the flows $Z_{k,i}$ from internal workstations $W_k, k \in I_1$. Using Equation (31) from [61], the arrival process at each $W_i : i \in I_2$ has CV of inter-arrival times:

$$CV^2_{i,\text{Arrival}} = \frac{\sum_{k \in I_1} Z_{k,i} CV^2_{k,i}}{Q_i} \tag{5.3}$$

For $k \in I_1$, using Proposition 3, if the robot arrival rate at W_k is Q_k, the flow from $W_k : k \in I_1$ to $W_i : i \in I_2$ has a intensity of $Z_{k,i} = \frac{Q_k}{\xi}$, and the CV of inter-departure times is

$$CV^2_{k,i} = \frac{1}{\xi}((Q_k \mathbb{E}[S_1] CV_{S_1})^2 + 1 - (Q_k \mathbb{E}[S_1])^2) \tag{5.4}$$

Proposition 3. *If an internal workstation has random service time S and Poisson arrival λ, the inter-departure time from it is approximately Normal$(\frac{\xi}{\lambda},\ \xi(Var[S] + \frac{1}{\lambda^2} - \frac{1}{\mu^2}))$ for large ξ and high work load ($\rho = \frac{\lambda}{\mu} \to 1$).*

The proof is given in Appendix Section B.1.

Estimation of Transportation Cost with Queue Delay

Using Kingman's formula for G/G/1 queues, the robot sojourn time D_i (sum of service time and waiting time) in $W_i : i \in I_1 \cup I_3$ with Poisson arrival Q_i is:

$$\mathbb{E}[D_i] = Q_i \left(\frac{1}{\mu_i} + \frac{1 + CV^2_{S_1}}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \tag{5.5}$$

The robot queuing delay at $W_i : i \in I_2$ with arrival Q_i, $CV_{i,\text{Arrival}}$ from Equations 5.3 and 5.4 is:

$$\mathbb{E}[D_i] = Q_i \left(\frac{1}{\mu_i} + \frac{CV^2_{k,\text{Arrival}} + CV^2_{S_2}}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \tag{5.6}$$

If the travel time is $d_{j,i}$ from T_j to W_i, and $d_{k,i}$ from $W_k : k \in I_1$ to $i \in I_3$, the total transportation cost is:

$$\sum_{i \in I_1 \cup I_3} \left[\sum_{j \in J} d_{j,i} Y_{j,i} + Q_i \left(\frac{1}{\mu_i} + \frac{1 + CV^2_{S_1}}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right]$$
$$+ \sum_{i \in I_2} \left[\sum_{k \in I_1} d_{k,i} Z_{k,i} + Q_i \left(\frac{1}{\mu_i} + \frac{CV^2_{k,\text{Arrival}} + CV^2_{S_2}}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right]$$

Location-Allocation-Queue Model (LAQM)

Formulation as Mixed Integer Programming(MIP)

Let $X_i \in \{0,1\}$ indicate whether we open a workstation at location W_i. $Y_{i,j}$ and $Z_{k,i}$ are pod-to-workstation and internal-to-external-workstation traffic flows (in robots per unit time). We introduce the binary variable $V_{k,i}$ to ensure that one internal workstation is assigned to one external workstation. The Location-Allocation-Queue Model (LAQM) is:

$$
\begin{aligned}
\text{(LAQM)}\quad \min \sum_{i\in I_1\cup I_3} & \left[\sum_{j\in J} d_{j,i}Y_{j,i} + Q_i\left(\frac{1}{\mu_i} + \frac{1+CV_{S_1}^2}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right)\right] \\
& + \sum_{i\in I_2}\left[\sum_{k\in I_1} d_{k,i}Z_{k,i} + Q_i\left(\frac{1}{\mu_i} + \frac{CV_{i,\text{Arrival}}^2 + CV_{S_2}^2}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right)\right] \qquad (5.7)\\
\text{s.t.}\quad & \text{Equations (5.1), (5.2), (5.3), and (5.4)hold and} \\
& \sum_{i\in I}\alpha_i X_i \le \theta \qquad (5.8)\\
& \sum_{i\in I_1\cup I_3} Y_{j,i} = \lambda_j, \forall j\in J \qquad (5.9)\\
& Q_i \le X_i\mu_i, \forall i \in I \qquad (5.10)\\
& \sum_{i\in I_2} Z_{k,i} = \frac{Q_k}{\xi}, \forall k\in I_1 \qquad (5.11)\\
& Z_{k,i} - MV_{k,i} \le 0, \forall k \in I_1, i\in I_2 \qquad (5.12)\\
& \sum_{i\in I_2} V_{k,i} = 1, \forall k\in I_1 \qquad (5.13)\\
& X_i \in \{0,1\}, \forall i\in I \qquad (5.14)\\
& V_{k,i}\in\{0,1\}, \forall k\in I_1, i\in I_2 \qquad (5.15)\\
& Y_{i,j}, Z_{ik}, Q_i \ge 0 \qquad (5.16)
\end{aligned}
$$

Constraints Equations (5.8) limit the budget of opened workstations where θ is the total budget. Constraints Equations (5.1)-(5.4) give definitions of variables Q and CV. Equation (5.9) is the demand satisfaction condition for pods to workstations, and Equation (5.2) is the demand satisfaction condition for special pods from $W_k : k \in I_1$ to $W_i : i \in I_2$. Equation (5.10) enforces that no flow can use closed workstations. Equations (5.12) and (5.13) ensure that an internal workstation is connected to at most one external workstation using a binary variable $V_{k,i}$ and a large number M.

Approximated LAQM (A-LAQM)

The CV of the inter-arrival times at external workstations makes (LAQM) too complicated to solve. Here we propose an approximation:

Approximation 1. *For special pods with large capacity ξ:*

$$CV_{i,k}^2 = \frac{1}{\xi}((Q_i\mathbb{E}[S_1]CV_{S_1})^2 + 1 - (Q_i\mathbb{E}[S_1])^2) \approx 0 \tag{5.17}$$

so that the sojourn time at $W_i : i \in I_2$ is approximately:

$$\mathbb{E}[\tilde{D}_i] = Q_i\left(\frac{1}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right) \tag{5.18}$$

The following theorem validates this approximation:

Theorem 2. *Assume $\exists\epsilon : 1 > \epsilon > 0$, such that the utilization $\rho_i \leq 1 - \epsilon$, $\forall i \in I_2$. If $\exists\varepsilon : 1 > \varepsilon > 0$, such that*

$$\xi \geq \frac{1-\epsilon}{2\varepsilon\epsilon}(1 + |CV_{S_1}^2 - 1|) \tag{5.19}$$

then the absolute relative error $\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]}$ in the sojourn time compared to Kingman's Formula (5.6) *is bounded: $\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]} \leq \varepsilon$*

The proof is given in Appendix Section B.2.

Remark 3. *We show in Theorem 2 that the relative error decreases proportionally to $\frac{1}{\xi}$. With $\xi = 100$, an exponentially distributed service time S_1, and load $\rho_i = 0.8$ we can guarantee that the relative error in the sojourn time using approximation 1 is less than 2%.*

Remark 4. *Theorem 2 gives a an upper bound for relative error: $\frac{\rho}{2\xi(1-\rho)}(1 + |CV_{S_1}^2 - 1|)$*

Using Approximation 1, the LAQM can be approximated as the Approximated LAQM (A-LAQM)

$$\begin{aligned}\text{(A-LAQM)} \quad \min \sum_{i \in I_1 \cup I_3}&\left[\sum_{j \in J} d_{j,i}Y_{j,i} + Q_i\left(\frac{1}{\mu_i} + \frac{1 + CV_{S_1}^2}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right)\right] \\ + \sum_{i \in I_2}&\left[\sum_{k \in I_1} d_{k,i}Z_{k,i} + Q_i\left(\frac{1}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right)\right] \\ \text{s.t. Equations (5.1), (5.2), and (5.8)–(5.16) hold}&\end{aligned} \tag{5.20}$$

5.4 Solution Method

Small-to-Medium Scale Problem: Reformulate as SOCP

The (A-LAQM) in Equation 5.20 is a nonlinear mixed integer programming with linear constraints. Inspired by [5] for the queuing cost, we can reformulate (A-LAQM) as mixed integer second order conic programming (MISOCP). Introduce auxiliary variables $G_i, H_i : i \in I$ and new constraints so that $G_i \geq \frac{Q_i^2}{\mu_i - Q_i}$, and $H_i = \mu_i X_i - Q_i$, the (A-LAQM) can be transformed into:

$$\begin{aligned}
\text{(SOCP-A-LAQM)} \quad \min \; & \sum_{i \in I_1 \cup I_3} \left[\sum_{j \in J} d_{j,i} Y_{j,i} + \frac{Q_i}{\mu_i} + \frac{1 + CV_{S_1}^2}{2\mu_i} G_i \right] \\
& + \sum_{i \in I_2} \left[\sum_{k \in I_1} d_{k,i} Z_{k,i} + \frac{Q_i}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i} G_i \right] && (5.21) \\
\text{s.t. } & \text{Equations (5.1), (5.2), and (5.8)–(5.16) hold, and} \\
& Q_i^2 \leq G_i H_i, \forall i \in I && (5.22) \\
& H_i = \mu_i X_i - Q_i, \forall i \in I && (5.23) \\
& G_i \geq 0, H_i \geq 0, \forall i \in I && (5.24)
\end{aligned}$$

The problem (SOCP-A-LAQM) has a linear objective function. The constraint (5.22) is hyperbolic, and all the other constraints are linear. Constraints (5.22) and (5.23) are introduced to include nonnegative continuous variables G_i and H_i to transform the problem into a MISCOP. (SOCP-A-LAQM) has the same solution as (A-LAQM), and can be solved using off-the-shelf conic programming solvers.

Large Scale Problem: Lagrangian Relaxation with SOCP Sub-Problems

For large-scale problems with many candidates in I, the (SOCP-A-LAQM) becomes difficult to solve directly. We can use Lagrangian Relaxation techniques to solve the (A-LAQM) by relaxing Equation 5.10. Consider the dual function

$$\begin{aligned}
L(\pi) = & \min_{X,Y,Z,V,Q} \sum_{i \in I_1 \cup I_3} \left[\sum_{j \in J} d_{j,i} Y_{j,i} + Q_i \left(\frac{1}{\mu_i} + \frac{1 + CV_{S_1}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] \\
& + \sum_{i \in I_2} \left[\sum_{k \in I_1} d_{k,i} Z_{k,i} + Q_i \left(\frac{1}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] + \sum_{i \in I} \pi_i (Q_i - X_i \mu_i) \\
= & \sum_{i \in I} (-\mu_i \pi_i) X_i + \sum_{i \in I_1 \cup I_3} \left[\sum_{j \in J} d_{j,i} Y_{j,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} + \frac{1 + CV_{S_1}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] \\
& + \sum_{i \in I_2} \left[\sum_{k \in I_1} d_{k,i} Z_{k,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] && (5.25) \\
\text{s.t. } & \text{Equations (5.1), (5.2), (5.8), (5.9), and (5.11) –(5.16) hold}
\end{aligned}$$

Note $L(\pi) = L_1(\pi) + L_2(\pi)$, where we define two sub-problems:

(L1 Problem):

$$L_1(\pi) = \min_X \sum_{i\in I} (-\mu_i \pi_i) X_i \tag{5.26}$$

s.t. Equations (5.8) and(5.14) hold

(L2 Problem):

$$\begin{aligned} L_2(\pi) = \min_{Y,Z,V,Q} & \sum_{i\in I_1\cup I_3} \left[\sum_{j\in J} d_{j,i}Y_{j,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} + \frac{1+CV_{S_1}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] \\ & + \sum_{i\in I_2} \left[\sum_{k\in I_1} d_{k,i}Z_{k,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} + \frac{CV_{S_2}^2}{2\mu_i} \frac{Q_i}{\mu_i - Q_i} \right) \right] \end{aligned} \tag{5.27}$$

s.t. Equations (5.1), (5.2), (5.9), and (5.11)–(5.13), (5.15)–(5.16) hold

Note we can transform (L2 Problem) into a MISOCP by introducing $G_i' \geq \frac{Q_i^2}{\mu_i - Q_i}$

(SOCP L2 Problem)

$$\begin{aligned} L_2^{SOCP}(\pi) = \min_{Y,Z,V,Q,G'} & \sum_{i\in I_1\cup I_3} \left[\sum_{j\in J} d_{j,i}Y_{j,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} \right) + \frac{1+CV_{S_1}^2}{2\mu_i} G_i' \right] \\ & + \sum_{i\in I_2} \left[\sum_{k\in I_1} d_{k,i}Z_{k,i} + Q_i \left(\pi_i + \frac{1}{\mu_i} \right) + \frac{CV_{S_2}^2}{2\mu_i} G_i' \right] \end{aligned} \tag{5.28}$$

s.t. Equations (5.1), (5.2), (5.9), and (5.11)–(5.13), (5.15)–(5.16) hold, and

$$Q_i^2 \leq G_i(\mu_i - Q_i) \tag{5.29}$$

Similar to Equation 5.22, Equation 5.29 introduces G_i' and transforms (L2 Problem) into a MISOCP.

(L1 Problem) is a 0-1 knapsack problem. In our case, assuming the cost of workstations are the same as long as they have the same type (Assumption 10), if we know $n_{X,1} = \sum_{i\in I_1} X_i$, $n_{X,2} = \sum_{i\in I_1} X_i$, and $n_{X,3} = \sum_{i\in I_1} X_i$, at optimal, the $n_{X,1}$ opened workstation in I_1 will be those with the smallest $-\mu_i\pi_i$. Therefore, with this assumption, we only need to find $n_{X,1}, n_{X,2}$, and $n_{X,3}$ to find the optimal solution. The problem can be solved using Algorithm 3, with the time complexity of $O(|I_1||I_2||I_3|)$.

Assumption 10. *Assume $\alpha_i = \alpha_j$ if $i, j \in I_1$ or $i, j \in I_2$, or $i, j \in I_3$.*

(SOCP L2 Problem) is a MISOCP which can be solved using off-the-shelf solvers. Therefore, $L(\pi)$ in Equation 5.25 can be found numerically.

The subgradient $\frac{\partial L}{\partial \pi_i} = Q_i - X_i\mu_i$, and we have $\pi_i \geq 0$ since we relaxed inequality constraints. We can update the dual variable π using the subgradient-projection method (Algorithm 4).

Algorithm 3 L_1 Knapsack Algorithm

```
1: $Solution = \infty$
2: for $n_1 = 1, 2, ..., |I_1|$ do
3:     $X_i = 0, \forall i \in I$
4:     $X_i = 1$ if $-\pi_i\mu_i$ is one of the $n_1$ smallest in $\{-\pi_i\mu_i : i \in I_1\}$
5:     for $n_2 = 1, 2, ..., |I_2|$ do
6:         $X_i = 1$ if $-\pi_i\mu_i$ is one of the $n_2$ smallest in $\{-\pi_i\mu_i : i \in I_2\}$
7:         for $n_3 = 1, 2, ..., |I_3|$ do
8:             $X_i = 1$ if $-\pi_i\mu_i$ is one of the $n_3$ smallest in $\{-\pi_i\mu_i : i \in I_3\}$
9:             if Equation 5.8 is satisfied then
10:                $Solution = \min\{Solution, \sum_{i\in I}(-\mu_i\pi_i X_i)\}$
11:            end if
12:        end for
13:    end for
14: end for
```

Algorithm 4 Lagrangian Relaxation

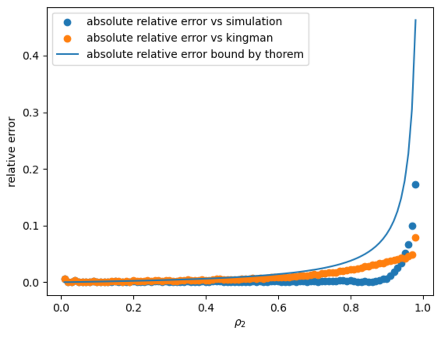
```
1: Initialize: Lagrange multipliers $\pi_i \geq 0, i \in I$, tolerance $\epsilon$, step size $\alpha$
2: Set: $UB = +\infty$, $LB = -\infty$
3: while $UB - LB > \epsilon$ do
4:     Solve the relaxed problem:
5:         $L(\pi) = L_1(\pi) + L_2^{SOCP}(\pi)$
6:     Update $LB = \max(LB, L(\pi))$
7:     Update the Lagrange multipliers:
8:         $\pi_i = \max(0, \pi_i + \alpha\frac{UB-L(\pi)}{\sum_{i\in I}(Q_i - X_i\mu_i)^2}(Q_i - X_i\mu_i)), i \in I$
9:     Update $UB = \min(UB, \text{Solution to (A-LAQM) with } X \text{ given by } L_1)$
10: end while
11: Output: Optimal solution and optimal objective value $UB$
```

5.5 Numerical Results

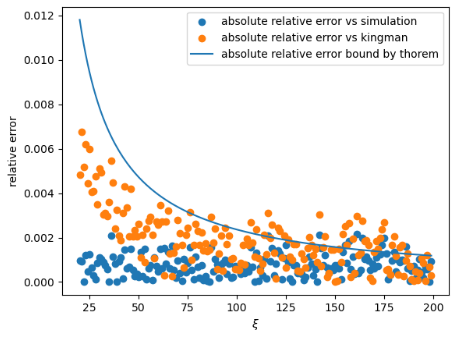
Validation of Sojourn Time Approximation and Theorem 2

In this section, we validate our approximation method and the bound (Theorem 2) using numerical simulation. Consider the simple system with three workstations in Figure 5.4c. We set $I_1 = \{0, 1\}$ and $I_2 = \{2\}$. Workstations 0 and 1 process incoming normal pods, batch the totes in special pods, and send special pods to workstation 2. Assume that the service times $\mathbb{E}[S_2] = 100$, $\text{Var}[S_2] = 16$, $CV_{S_1} = 1/3$, S_1, S_2 are normally distributed.

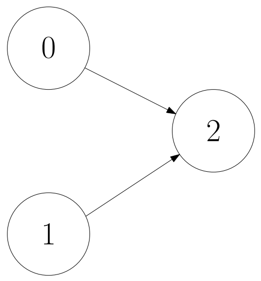
From Theorem 2, the relative error of our approximation (5.18) compared to Kingman's formula (5.6) is bounded by $\frac{1-\epsilon}{2\xi\epsilon}(1 + |CV_{S_1}^2 - 1|)$, where we can set $\epsilon = 1 - \rho_2$. We conducted two experiments to see how the relative error changed with the utilization of the workstation



(a) ρ_2 vs errors



(b) ξ vs errors



(c) System for estimation validation

Figure 5.4: Validate the error bound

ρ_2 and the special capacity of the pod ξ.

In the first experiment, we fixed the special pod capacity at $\xi = 100$ and changed ρ_2 from 0.01 to 0.99. For each ρ_2, we set $\mathbb{E}[S_1] = \frac{1}{\rho_2 \xi \mathbb{E}[S_2]}$ so that $\rho_1 \approx 0.5$ (note that we have identical I_1 workstations). We run each simulation for 1000000 time units and estimate the average sojourn time and the coefficient of variance of inter-arrival time at workstation 2. We calculated the average sojourn time, the sojourn time from Kingman's Formula (5.6) using simulated CV, and our approximation (5.18). We also calculated the error bound $(\frac{1-\epsilon}{2\xi\epsilon}(1 + |CV_{S_1}^2 - 1|))$ with $\epsilon = \rho_2$. The result is given in Figure 5.4a.

From Figure 5.4a, we notice that the absolute estimation error increases with ρ_2. The errors are bounded by the error bound from Theorem 2. Note that the error compared with simulation can be less than the error compared with Kingman's formula because Kingman's formula tends to overestimate the sojourn time while our approximation underestimates Kingman's formula.

In the second experiment, we set the service time $\mathbb{E}[S_1]$ to a proper value so that $\rho_2 \approx 0.2, \rho_1 \approx 0.5$, and change the special pod size ξ from 20 to 200. The results are shown in Figure 5.4b. Since the absolute error is estimated from simulation, this estimation has its variance, so some scatter dots lie over the error bound, since the error bound only bounds the expectation. We notice that the error compared to Kingman's formula (orange dots in Figure 5.4b) has a similar trend with our error bound, and the error bound from Theorem 2 is an upper bound for the absolute error compared to Kingman's formula.

In conclusion, from these experiments, our approximation is accurate for the sojourn time of the workstation in I_2. If $\rho < 0.8$, the error is less than the approximate error of Kingman's formula, resulting in an absolute relative error $< 3\%$. As we increase the size of special pods, the approximation error further decreases. In addition, the bound given in Theorem 2 is an upper bound of the relative approximation error. Managers can use this bound to evaluate our method.

Evaluation of the Design Optimality

In this section, we investigated the optimal workstation location found by (SOCP-A-LAQM) and the Lagrangian Relaxation Algorithm 4 in both our new design and conventional KIVA systems and compared their performance in terms of the sum of robot travel time and waiting time at workstations. We compared different system layouts and the budgets for workstations. We considered three levels of system storage capacity. We considered a square and rectangle layout for each level of storage capacity. The budget for workstations increased with the storage capacity of the system. The scenarios of numerical experiments are presented in Table 5.1. Using the parameters $\mathbb{E}[S_1] = 10, \mathbb{E}[S_2] = 200, CV_{S_1} = 4, CV_{S_2} = 16$, and setting the fixed cost $\alpha_i = 1$ for all the workstations, we assumed that the external workstations were located on two vertical edges. We defined *relative depth* as $\frac{Width}{Height}$ in our experiments due to the position of our external workstations

scenario	1	2	3	4	5	6
Height	15	10	20	15	30	20
Width	15	22	20	27	30	45
Relative depth	1	2.2	1	1.8	1	2.25
budget (# of workstations)	5	5	8	8	12	12

Table 5.1: Scenarios of numerical experiments

Table 5.2 includes the cost comparison result. Transportation cost was estimated using discrete-even simulations for the open queueing network in Figure 5.3, representing the total robot machine time per unit time (or the average active robot number in the system). Compared with the KIVA system, the experiment showed that our new design can lower the operational cost concerning robot travel time and waiting time at workstations.

The relative improvement is more significant for larger systems because implementing internal workstations reduces robot travel time. In a larger system, robot travel time plays a larger role in total machine time, so our new design with internal workstations provides more benefits.

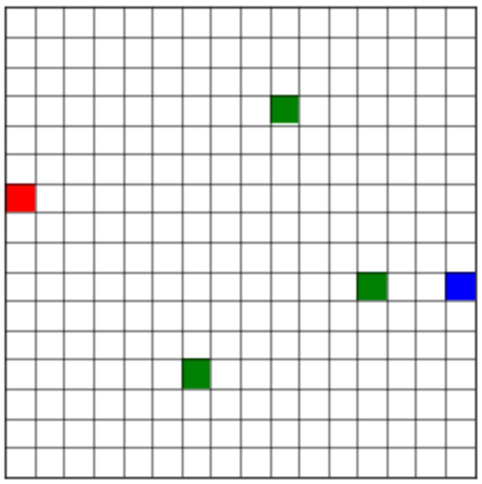
A larger relative improvement can be expected in a deeper system, that is, with a larger relative depth. More reduction in robot travel time can be obtained in a deeper system with internal workstations due to the batching of robots traveling from internal workstations to external workstations.

Therefore, our design outperforms KIVA for larger and deeper systems.

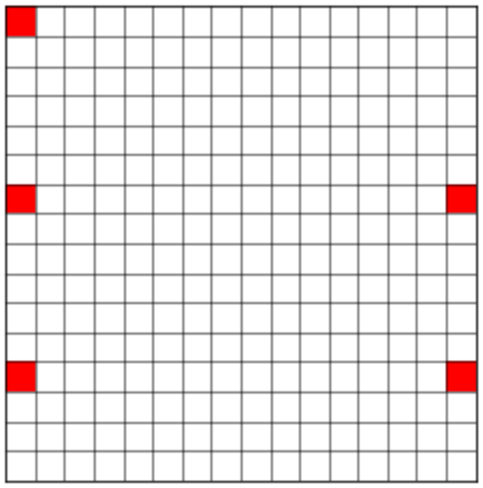
scenario	1	2	3	4	5	6
cost of new design	0.77	1.85	3.4	3.45	8.96	8.98
cost of KIVA	0.8	2	3.63	3.8	9.97	11.22
Imp(%)	3.75	7.50	6.34	9.21	10.13	19.96

Note: Imp $= 100\% * \frac{\text{cost of KIVA -cost of new design}}{\text{cost of KIVA}}$.

Table 5.2: Cost comparison between our new design and KIVA system



(a) new design



(b) KIVA

Figure 5.5: Layout of scenario 1

Legend:
Green: internal workstations $i \in I_1$.
Blue: external workstation for special pods $i \in I_2$.
Red: external workstation for normal pods $i \in I_3$

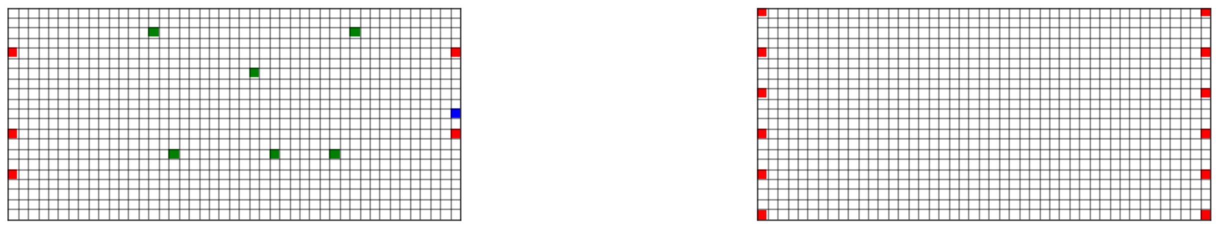

(a) new design (b) KIVA

Figure 5.6: Layout of scenario 6

Legend:
Green: internal workstations $i \in I_1$.
Blue: external workstation for special pods $i \in I_2$.
Red: external workstation for normal pods $i \in I_3$

Effect of Processing Time of Workstation on The Optimal Design

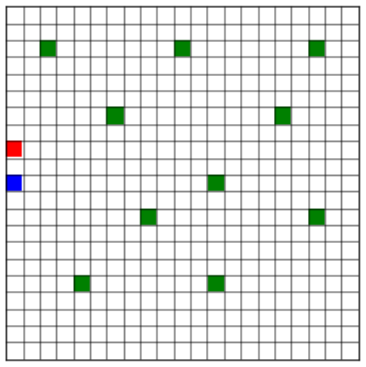
In this section, we explored the effect of the processing time of internal and external workstations on the optimal workstation location and the comparison between our new design and KIVA systems. To this aim, we fixed $\mathbb{E}[S_1] = 10, CV_{S_1} = 4, CV_{S_2} = 16$ and vary $\mathbb{E}[S_2]$ from 50 to 450 with a step size of 50. The system size was $W = H = 20$, and the fixed cost of each workstation was set as $\alpha_i = 1$.

The opened workstation location of our new design varied with $\mathbb{E}[S_2]$, which is depicted in Fig.5.7. With the increase of $\mathbb{E}[S_2]$, more budgets will be allocated to external workstations, and fewer internal workstations will be used. Under the optimal solution given by our location-allocation-queuing model, the budget assignment will balance the workloads between internal and external workstations to prevent long queues at certain workstations. A larger $\mathbb{E}[S_2]$ means a smaller processing capacity of each external workstation. More budget workstations will be used for external workstations to balance the workload between internal and external workstations.

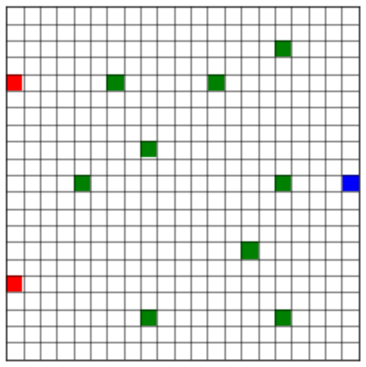
Therefore, managers should deploy more internal workstations if $\mathbb{E}[S_2]$ is small.

Effect of Demand Density on the Optimal Design

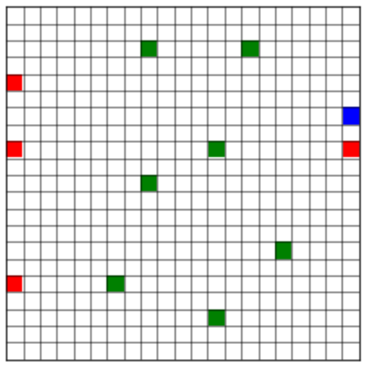
In this section, we investigate the effect of the demand density of the storage pods by comparing the machine time of our new design with that of the KIVA system. We evenly took 10 values from 0.01 to 0.05 for the demand density and fixed the system size as $W = H = 20$, the budget of the workstations as 12, and the fix ed cost of each workstation as $\alpha = 1$. The comparison result is included in Table 5.3. It shows that using our new design leads to



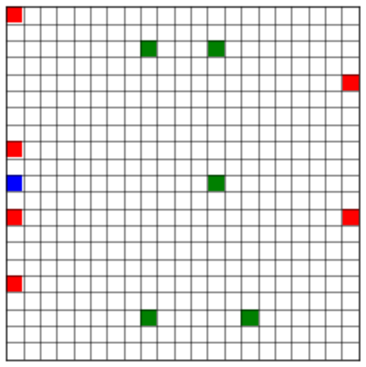
(a) $\mathbb{E}[S_2] = 50$



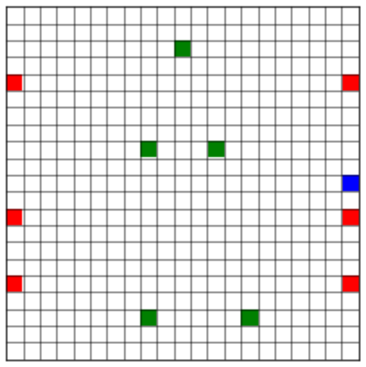
(b) $\mathbb{E}[S_2] = 100$



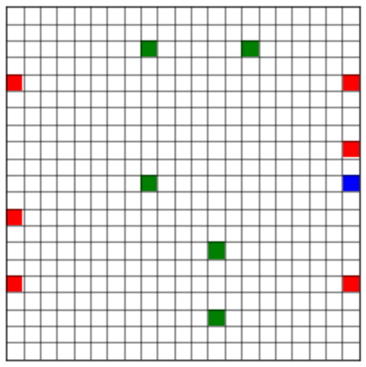
(c) $\mathbb{E}[S_2] = 150$



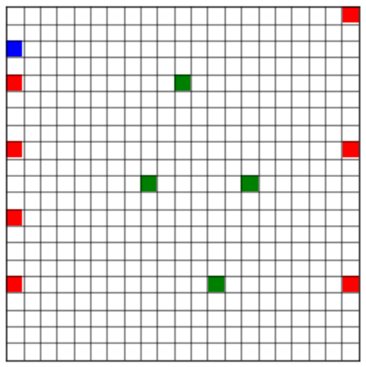
(d) $\mathbb{E}[S_2] = 200$



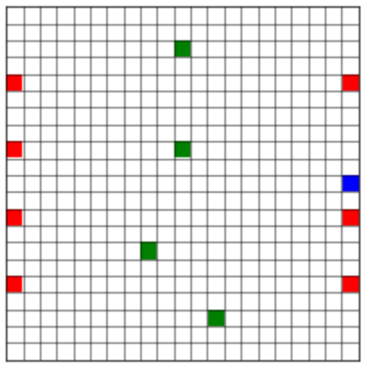
(e) $\mathbb{E}[S_2] = 250$



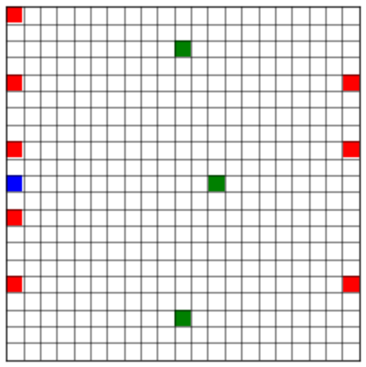
(f) $\mathbb{E}[S_2] = 300$



(g) $\mathbb{E}[S_2] = 350$



(h) $\mathbb{E}[S_2] = 400$



(i) $\mathbb{E}[S_2] = 450$

Figure 5.7: Workstations location with respect to varying $\mathbb{E}[S_2]$

Legend:
Green: internal workstations $i \in I_1$.
Blue: external workstation for special pods $i \in I_2$.
Red: external workstation for normal pods $i \in I_3$

a smaller total machine time in a low demand density system, whereas the KIVA system performs better in the system with a high demand density system.

The trade-off between the robot travel time reduction and the robot waiting time results in this phenomenon. Our design sacrificed queue waiting time for transportation time. By introducing internal workstations, the travel distance is greatly reduced. However, orders fulfilled in internal workstations must go through two queue servers (in I_1 and I_2 workstations), while orders in the KIVA-like system only go through one (at I_3) in the OQN (Figure 5.3).

In a large system with low-demand density, the robot travel time reduction dominates the robot waiting time at workstations, so our new design outperforms the KIVA system in large systems. The internal workstation is less effective for smaller systems with high demand density, so a conventional KIVA-like design works fine.

Therefore, combined with the result in Table 5.1, one insight is that our design performs better for larger systems with lower demand density.

Demand density	0.010	0.014	0.019	0.023	0.028	0.032	0.037	0.041	0.046	0.050
NewDesign cost	3.21	4.80	6.59	8.70	11.14	14.29	18.43	24.90	36.37	66.84
KIVA cost	3.42	5.13	7.00	9.09	11.49	14.34	17.91	22.67	29.76	42.27
Imp(%)	5.98	6.36	5.76	4.30	3.07	0.40	-2.95	-9.83	-22.2	-58.1

Table 5.3: Operational cost comparison with respect to different demand densities

Conclusion

Based on our simulation experiments, we conclude that:

- Our approximation is accurate for external workstations that handle special pods, with small utilization ρ and large special pod capacity ξ. (See Figure 5.4)
- The approximation error bound in Theorem 2 provides an upper bound of the relative estimation error on sojourn time at each I_2 workstation. (See Figure 5.4)
- Compared to KIVA, our design can decrease machine time by 10-20% on large and deep systems. (See Table 5.2)
- The reduction in machine time of our design compared to KIVA is more significant for a larger, deeper system with smaller demand density. (See Table 5.2 and 5.3)
- According to our algorithm, fewer internal workstations should be deployed if the processing time for special pod $\mathbb{E}[S_2]$ is large. (See Figure 5.7.)

5.6 Summary

We introduced a novel design for a fully automated Robotic Mobile Fulfillment System (RMFS), which features certain workstations with robotic arms located within storage areas. These designated workstations retrieve items from standard pods and consolidate completed orders into a special pod. Once filled, this special pod is dispatched to external workstations. This configuration facilitates the batching of items within the special pod, thereby minimizing the travel distance for robots.

For both this innovative design and the traditional KIVA system, strategically positioning workstations to optimize transportation time and minimize queueing delays is crucial. To address this, we developed a location-allocation-queuing model to determine optimal workstation placements. This problem can be formulated as mixed integer second-order conic programming (MISOCP) and tackled using commercial solvers. We employ Lagrangian Relaxation methods with MISOCP subproblems for larger-scale problems, analyzing the approximation errors and establishing an error boundary.

Our numerical simulations confirm the accuracy of our model, demonstrating that if the external workstation serving special pod has $\rho < 0.8$, the relative error in sojourn time remains below 3%. We also validate our error-bound theorem. Furthermore, our approach can reduce the robot machine time by $10 - 20\%$ compared to the KIVA system with optimally located workstations. Our study showed that our design benefits larger and deeper systems with a lower demand density. In addition, as suggested by our algorithm, managers should deploy fewer internal workstations when special pods take more time to process.

Our findings pave the way for several future research avenues. The computational efficiency of our Lagrangian Relaxation algorithm can be improved through improved trimming, branching, and cutting techniques. Moreover, real-world validation of our design is necessary to fully realize its potential and applicability.

Chapter 6

Conclusions

This study investigated the evaluation, operation, and new designs for robotic warehouses for e-commerce. The evaluation model can accurately predict throughput and traffic congestion in a complicated robotic warehouse. The operation strategy is an offline-online method based on the evaluation model, where the robot assignment and path-finding problems are solved simultaneously. The new design is a fully robotic warehouse, motivated by the development of robotic arms and the idea of batch-pooling to reduce transportation costs.

The robotic warehouse can be modeled as a CQN with blocking. We approximated the system as a transportation network with a closed-form link cost function that represents transportation and waiting costs. Our simulation experiments provided support for our proposed congestion mechanism and approximation using the Poisson flow. We proved asymptotic Poisson properties for light-traffic cells and showed a similar property for heavy-traffic cells using numerical experiments. We also combined the CQN with the approximated transportation cost and developed an iterative algorithm to estimate the system throughput. The simulation showed that our throughput prediction is accurate if the system is stable.

The operation in a robotic warehouse includes robot-to-workstation assignments, parcel-to-workstation assignments, and multi-robot path-finding. We developed an integrated assignment and path-finding method that has two parts: online and offline. Based on our evaluation model, we can find a near-optimal traffic assignment offline and use the offline traffic assignment to guide the robot's operation in real time. Since most of the complicated optimization problems were solved offline, the online algorithm was fast enough for large-scale problems. Due to the approximation error of the evaluation, our method would not perform well in certain layouts, and we developed a diagnostic indicator to evaluate our method using historical operation data before actually running the offline-online method. The simulation showed that our method can improve throughput by about 10% compared to zoning assignment, and the diagnostic indicator accurately predicts whether our method is suitable for certain systems.

With the development of robotic arms, it becomes possible to have a fully robotic warehouse. We proposed a design for a fully robotic fulfillment facility where some workstations are located inside the storage area to reduce transportation costs (internal workstations), and some workstations are on the periphery with access to exits or conveyors. We designed a batch-pooling process using special pods to transport completed orders from internal workstations to external workstations. To find the optimal locations and number of workstations, we developed a location-allocation-queue model with approximation so that the model can be transformed into a mixed-integer second-order-conic programming (MISOCP). We also developed Lagrangian relaxation procedures with MISOCP as subproblems to solve large-scale problems. Our simulation results show that our design can significantly reduce transportation costs compared to Kiva's design for large and deep systems.

Bibliography

[1] Robert Aboolian, Oded Berman, and Zvi Drezner. “Location and allocation of service units on a congested network”. In: *IIE Transactions* 40.4 (2008), pp. 422–433.

[2] Robert Aboolian, Oded Berman, and Dmitry Krass. “Profit maximizing distributed service system design with congestion and elastic demand”. In: *Transportation Science* 46.2 (2012), pp. 247–261.

[3] Robert Aboolian, Oded Berman, and Vedat Verter. “Maximal accessibility network design in the public sector”. In: *Transportation Science* 50.1 (2016), pp. 336–347.

[4] Saligrama R Agnihothri, Sridhar Narasimhan, and Hasan Pirkul. “An assignment problem with queueing time cost”. In: *Naval Research Logistics (NRL)* 37.2 (1990), pp. 231–244.

[5] Amir Ahmadi-Javid and Pooya Hoseinpour. “Convexification of queueing formulas by mixed-integer second-order cone programming: An application to a discrete location problem with congestion”. In: *INFORMS Journal on Computing* 34.5 (2022), pp. 2621–2633.

[6] Amazon. *10 years of Amazon robotics: how robots help sort packages, move product, and improve safety*. Web Page. 2022. URL: `https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety`.

[7] Ben Ames. *Amazon unveils Xanthus and Pegasus fulfillment robots*. Web Page. 2019. URL: `https://www.dcvelocity.com/articles/30765-amazon-unveils-xanthus-and-pegasus-fulfillment-robots#.XP8tsesSJVs.twitter`.

[8] Tuan Le-Anh and M. B. M. De Koster. “A review of design and control of automated guided vehicle systems”. In: *European Journal of Operational Research* 171.1 (2006), pp. 1–23. ISSN: 0377-2217. DOI: `10.1016/j.ejor.2005.01.036`. URL: `https://dx.doi.org/10.1016/j.ejor.2005.01.036`.

[9] Kaveh Azadeh, René De Koster, and Debjit Roy. “Robotized and automated warehouse systems: Review and recent developments”. In: *Transportation Science* 53.4 (2019), pp. 917–945.

[10] Maria Torcoroma Benavides-Robles et al. “Robotic Mobile Fulfillment System: A Systematic Review”. In: *Available at SSRN 4445297* (2023).

[11] Nils Boysen, Dirk Briskorn, and Simon Emde. “Parts-to-picker based order processing in a rack-moving mobile robots environment”. In: *European Journal of Operational Research* 262.2 (2017), pp. 550–562.

[12] Nils Boysen, René de Koster, and Felix Weidinger. “Warehousing in the e-commerce era: A survey”. In: *European Journal of Operational Research* 277.2 (2019), pp. 396–411. ISSN: 03772217. DOI: `10.1016/j.ejor.2018.08.023`.

[13] Nils Boysen, Konrad Stephan, and Felix Weidinger. “Manual order consolidation with put walls: the batched order bin sequencing problem”. In: *EURO Journal on Transportation and Logistics* 8 (2019), pp. 169–193.

[14] Ronald Buitenhek, Geert-Jan Van Houtum, and Henk Zijm. “AMVA-based solution procedures for open queueing networks with population constraints”. In: *Annals of Operations Research* 93.1/4 (2000), pp. 15–40. ISSN: 0254-5330. DOI: `10.1023/a:1018967622069`. URL: `https://dx.doi.org/10.1023/a:1018967622069`.

[15] Xiao Cai, Sunderesh S. Heragu, and Yang Liu. “Modeling and evaluating the AVS/RS with tier-to-tier vehicles using a semi-open queueing network”. In: *IIE Transactions* 46.9 (2014), pp. 905–927. ISSN: 0740-817X 1545-8830. DOI: `10.1080/0740817x.2013.849832`.

[16] Yongcan Cao et al. “An overview of recent progress in the study of distributed multi-agent coordination”. In: *IEEE Transactions on Industrial Informatics* 9.1 (2012), pp. 427–438.

[17] Ignacio Castillo, Armann Ingolfsson, and Thaddeus Sim. “Social optimal location of facilities with fixed servers, stochastic demand, and congestion”. In: *Production and Operations Management* 18.6 (2009), pp. 721–736.

[18] Meg Coyle. *New robots, new jobs*. Web Page. 2019. URL: `https://www.aboutamazon.com/news/operations/new-robots-new-jobs`.

[19] Yves Dallery and Yannick Frein. “On decomposition methods for tandem queueing networks with blocking”. In: *Operations Research* 41.2 (1993), pp. 386–399. ISSN: 0030364X, 15265463. URL: `http://www.jstor.org/stable/171785` (visited on 10/11/2023).

[20] M. De Ryck, M. Versteyhe, and F. Debrouwere. “Automated guided vehicle systems, state-of-the-art control algorithms and techniques”. In: *Journal of Manufacturing Systems* 54 (2020), pp. 152–173. ISSN: 0278-6125. DOI: `10.1016/j.jmsy.2019.12.002`. URL: `https://dx.doi.org/10.1016/j.jmsy.2019.12.002`.

[21] Valerio Digani, Lorenzo Sabattini, and Cristian Secchi. “A probabilistic Eulerian traffic model for the coordination of multiple AGVs in automatic warehouses”. In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 26–32. DOI: `10.1109/LRA.2015.2505646`.

[22] Valerio Digani et al. “Ensemble coordination approach in multi-AGV systems applied to industrial warehouses”. In: *IEEE Transactions on Automation Science and Engineering* 12.3 (2015), pp. 922–934.

[23] Valerio Digani et al. “Hierarchical traffic control for partially decentralized coordination of multi AGV systems in industrial environments”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6144–6149.

[24] Banu Yetkin Ekren et al. “Matrix-geometric solution for semi-open queuing network model of autonomous vehicle storage and retrieval system”. In: *Computers & Industrial Engineering* 68 (2014), pp. 78–86. ISSN: 03608352. DOI: 10.1016/j.cie.2013.12.002.

[25] John J Enright and Peter R Wurman. “Optimization and coordinated autonomy in mobile fulfillment systems”. In: *Workshops at the twenty-fifth AAAI conference on artificial intelligence*. 2011.

[26] FedEx. *FedEx: Manage your shipments and returns*. Web Page. 2024. URL: https://www.fedex.com/en-us/home.html.

[27] K.J.C. Fransen et al. “A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems”. In: *Computers & Operations Research* 123 (2020), p. 105046. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2020.105046. URL: https://www.sciencedirect.com/science/article/pii/S0305054820301635.

[28] Geekplus Robotics. *GeekPlus Robotics*. Web Page. 2023. URL: https://www.geekplus.com/en/.

[29] Tzvika Hartman et al. “How to split a flow?” In: *2012 Proceedings IEEE INFOCOM*. IEEE. 2012, pp. 828–836.

[30] Sunderesh S. Heragu et al. “Analytical models for analysis of automated warehouse material handling systems”. In: *International Journal of Production Research* 49.22 (2011), pp. 6833–6861. ISSN: 0020-7543. DOI: 10.1080/00207543.2010.518994. URL: https://doi.org/10.1080/00207543.2010.518994.

[31] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. “On the complexity of motion planning for multiple independent objects; PSPACE- hardness of the ”warehouseman’s problem””. In: *The International Journal of Robotics Research* 3.4 (1984), pp. 76–88. DOI: 10.1177/027836498400300405. URL: https://doi.org/10.1177/027836498400300405.

[32] Markus Jager and Bernhard Nebel. “Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots”. In: *Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium*. Vol. 3. IEEE. 2001, pp. 1213–1219.

[33] Hyun-Jung Kim, Cristobal Pais, and Zuo-Jun Max Shen. “Item Assignment Problem in a Robotic Mobile Fulfillment System”. In: *IEEE Transactions on Automation Science and Engineering* 17.4 (2020), pp. 1854–1867. ISSN: 1545-5955. DOI: 10.1109/tase.2020.2979897.

[34] Suryakant Kumar, Jiuh-Biing Sheu, and Tanmoy Kundu. "Planning a parts-to-picker order picking system with consideration of the impact of perceived workload". In: *Transportation Research Part E: Logistics and Transportation Review* 173 (2023), p. 103088.

[35] Tim Lamballais, Debjit Roy, and MBM De Koster. "Estimating performance in a robotic mobile fulfillment system". In: *European Journal of Operational Research* 256.3 (2017), pp. 976–990.

[36] Tim Lamballais Tessensohn, Debjit Roy, and René B. M. De Koster. "Inventory allocation in robotic mobile fulfillment systems". In: *IISE Transactions* 52.1 (2019), pp. 1–17. ISSN: 2472-5854 2472-5862. DOI: `10.1080/24725854.2018.1560517`.

[37] Xiaowen Li et al. "A simulation study on the robotic mobile fulfillment system in high-density storage warehouses". In: *Simulation Modelling Practice and Theory* 112 (2021), p. 102366.

[38] Vladimir Marianov and Daniel Serra. "Location–allocation of multiple-server service centers with constrained queues or waiting times". In: *Annals of Operations Research* 111 (2002), pp. 35–50.

[39] K. T. Marshall and Richard V. Evans. "Some inequalities in queuing". In: *Operations Research* 16.3 (1968), pp. 651–668. ISSN: 0030364X, 15265463. URL: `http://www.jstor.org/stable/168590` (visited on 03/31/2024).

[40] M. Merschformann et al. "Decision rules for robotic mobile fulfillment systems". In: *Operations Research Perspectives* 6 (2019). ISSN: 22147160. DOI: `10.1016/j.orp.2019.100128`.

[41] M. Mountz. "Kiva the disrupter". In: *Harvard Business Review* 90 (2012), pp. 74–80.

[42] US Bureau of Public Roads. Office of Planning. Urban Planning Division. *Traffic assignment manual for application with a large, high speed computer*. US Department of Commerce, 1964.

[43] Ling Qiu et al. "Scheduling and routing algorithms for AGVs: A survey". In: *International Journal of Production Research* 40.3 (2002), pp. 745–760. ISSN: 0020-7543 1366-588X. DOI: `10.1080/00207540110091712`.

[44] Seyed Habib A Rahmati et al. "A multi-objective model for facility location–allocation problem with immobile servers within queuing framework". In: *Computers & Industrial Engineering* 74 (2014), pp. 1–10.

[45] Debjit Roy, Akash Gupta, and René B. M. De Koster. "A non-linear traffic flow-based queuing model to estimate container terminal throughput with AGVs". In: *International Journal of Production Research* 54.2 (2015), pp. 472–493. ISSN: 0020-7543 1366-588X. DOI: `10.1080/00207543.2015.1056321`.

[46] Debjit Roy et al. "Queuing models to analyze dwell-point and cross-aisle location in autonomous vehicle-based warehouse systems". In: *European Journal of Operational Research* 242.1 (2015), pp. 72–87. ISSN: 03772217. DOI: 10.1016/j.ejor.2014.09.040.

[47] Debjit Roy et al. "Robot-storage zone assignment strategies in mobile fulfillment systems". In: *Transportation Research Part E: Logistics and Transportation Review* 122 (2019), pp. 119–142.

[48] Hamid R Sayarshad and Joseph YJ Chow. "Non-myopic relocation of idle mobility-on-demand vehicles as a dynamic location-allocation-queueing problem". In: *Transportation Research Part E: Logistics and Transportation Review* 106 (2017), pp. 60–77.

[49] Melanie Schranz et al. "Swarm robotic behaviors and current applications". In: *Frontiers in Robotics and AI* (2020), p. 36.

[50] Jiuh-Biing Sheu and Tsan-Ming Choi. "Can we work more safely and healthily with robot partners? A human-friendly robot–human-coordinated order fulfillment scheme". In: *Production and Operations Management* 32.3 (2023), pp. 794–812.

[51] David Silver. "Cooperative pathfinding". In: *Proc. AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 1. 2005, pp. 117–122.

[52] Statista. *Amazon sellers: statistics & facts*. Web Page. 2024. URL: https://www.statista.com/topics/8024/third-party-3p-selling-on-amazon.

[53] Statista. *Annual retail e-commerce sales growth worldwide from 2014 to 2020*. Web Page. 2017. URL: https://www.statista.com/statistics/288487/forecast-of-global-b2c-e-commerce-growt/.

[54] Gelu-Ovidiu Tirian. "Automation of a warehouse by means of a robotic arm". In: *Annals of Faculty Engineering Hundeoara–International Journal of Engineering* 11 (2013).

[55] Tompkins Robotics. *Postal and parcel sortation*. Web Page. 2020. URL: https://tompkinsrobotics.com/postal-and-parcel-sortation/..

[56] U.S. Department of Commerce. *FOR IMMEDIATE RELEASE: TUESDAY, FEBRUARY 20, 2024: Quarterly Retail E-Commerce Sales*. Web Page. 2024. URL: https://www.census.gov/retail/ecommerce.html#:~:text=Total%20e%2Dcommerce%20sales%20for,14.7%20percent%20of%20total%20sales..

[57] R T Underwood. "Quality and theory of traffic flow". In: *Speed, Volume, and Density Relationships*. Yale University Bureau of Highway Traffic, 1961, pp. 141–187.

[58] Navneet Vidyarthi and Sachin Jayaswal. "Efficient solution of a class of location–allocation problems with stochastic demand and congestion". In: *Computers & Operations Research* 48 (2014), pp. 20–30.

[59] Zheng Wang et al. "Robot scheduling for mobile-rack warehouses: Human–robot coordinated order picking systems". In: *Production and Operations Management* 31.1 (2022), pp. 98–116.

[60] Felix Weidinger, Nils Boysen, and Dirk Briskorn. “Storage assignment with rack-moving mobile robots in KIVA warehouses”. In: *Transportation Science* 52.6 (2018), pp. 1479–1495. ISSN: 0041-1655 1526-5447. DOI: 10.1287/trsc.2018.0826.

[61] W. Whitt. “The queueing network analyzer”. In: *Bell System Technical Journal* 62.9 (1983), pp. 2779–2815. DOI: https://doi.org/10.1002/j.1538-7305.1983.tb03204.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1983.tb03204.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1983.tb03204.x.

[62] Shasha Wu et al. “Research of the layout optimization in robotic mobile fulfillment systems”. In: *International Journal of Advanced Robotic Systems* 17.6 (2020), p. 1729881420978543.

[63] Lin Xie et al. “Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems”. In: *European Journal of Operational Research* 288.1 (2021), pp. 80–97.

[64] Xianhao Xu et al. “Assignment of parcels to loading stations in robotic sorting systems”. In: *Transportation Research, Part E Logistics and Transportation Rev.* 164 (2022). ISSN: 13665545. DOI: 10.1016/j.tre.2022.102808.

[65] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. “A Survey and Analysis of Multi-Robot Coordination”. In: *International Journal of Advanced Robotic Systems* 10.12 (2013). ISSN: 1729-8814 1729-8814. DOI: 10.5772/57313.

[66] Peng Yang, Randy A Freeman, and Kevin M Lynch. “Multi-agent coordination by decentralized estimation and control”. In: *IEEE Transactions on Automatic Control* 53.11 (2008), pp. 2480–2496.

[67] Xiuqing Yang et al. “Non-traditional layout design for robotic mobile fulfillment system with multiple workstations”. In: *Algorithms* 14.7 (2021), p. 203.

[68] Xiying Yang et al. “Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system”. In: *Computers & Operations Research* 135 (2021), p. 105467.

[69] Jingjin Yu and Steven M. LaValle. “Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs”. In: *Proc. AAAI Conference on Artificial Intelligence.* 2013, pp. 1443–1449.

[70] Zhe Yuan and Yeming Yale Gong. “Bot-in-time delivery for robotic mobile fulfillment systems”. In: *IEEE Transactions on Engineering Management* 64.1 (2017), pp. 83–93.

[71] Youmin Zhang and Hasan Mehrjerdi. “A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations”. In: *2013 Int. Conf. on Unmanned Aircraft Systems (ICUAS).* 2013, pp. 1087–1096. DOI: 10.1109/ICUAS.2013.6564798.

[72] Lu Zhen et al. “How to deploy robotic mobile fulfillment systems”. In: *Transportation Science* (2023).

[73] Yanling Zhuang et al. “Order picking optimization with rack-moving mobile robots and multiple workstations”. In: *European Journal of Operational Research* 300.2 (2022), pp. 527–544.

[74] Bipan Zou, Xianhao Xu, René De Koster, et al. “Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system”. In: *European Journal of Operational Research* 267.2 (2018), pp. 733–753.

[75] Bipan Zou et al. “Assignment rules in robotic mobile fulfilment systems for online retailers”. In: *International Journal of Production Research* 55.20 (2017), pp. 6175–6192.

[76] Bipan Zou et al. “Robotic sorting systems: performance estimation and operating policies analysis”. In: *Transportation Science* 55.6 (2021), pp. 1430–1455. ISSN: 0041-1655 1526-5447. DOI: `10.1287/trsc.2021.1053`.

Appendix A

The Asymptotic Poisson Property

A.1 Proof of Theorem 1

Proof. **Proof of Theorem 1** If we have $R = v(\mathbb{E}[X] + \mathbb{E}[G])$ robots, let A_{ik} be the time the i-th agent visits the first server for the kth time, we have

$$A_{ik} = A_{i(k-1)} + G + X.$$

Therefore, A_{ik} is the k-th renewal time for a renewal process with an interarrival time of $G + X$. Consider a time period $t_1 < t_2$. Let n_i be the number of arrivals of customer i from t_1 to t_2. As $X \to \infty$ in probability, $t_2 - t_1 < X + G$ with high probability. Conditioned on $t_2 - t_1 < X + G$, we must have $n_i \in \{0, 1\}$. From the renewal theorem

$$\mathbb{P}(n_i = 1 | t_2 - t_1 < X + G) = \frac{t_2 - t_1}{\mathbb{E}[X] + \mathbb{E}[G]}.$$

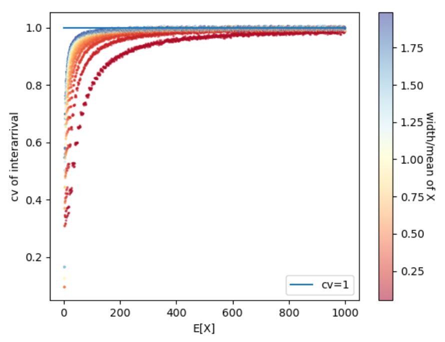
Therefore, $n_i \to$ Bernoulli$(\frac{t_2-t_1}{\mathbb{E}[X]+\mathbb{E}[G]})$ as $X \to \infty$ in probability. Let $N(t_1, t_2) = \sum_{i=1}^{R} n_i$ be the number of arrivals, $N(t_1, t_2) \to$ Binomial$(R, \frac{t_2-t_1}{\mathbb{E}[X]+\mathbb{E}[G]}) \to$ Poisson$(\frac{R(t_2-t_1)}{\mathbb{E}[X]+\mathbb{E}[G]})$ =Poisson(v). □

A.2 Simulation Study for Conjecture 1

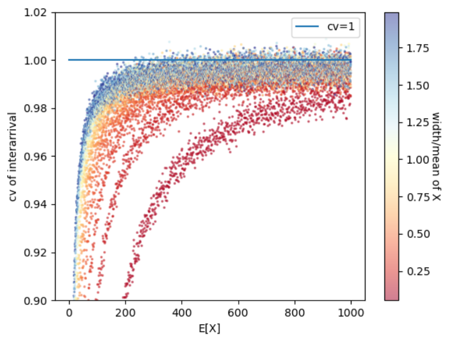
Proof. **Non-strict Proof of Conjecture 1** If we have R robots, and the arrival rate is v. Let A_{ik} be the time that ith agent visits the first server for the kth time; we have

$$A_{ik} = A_{i(k-1)} + G + X + W_{ik}.$$

where W_{ik} is the waiting time for ith agernt visits G for the kth time. Non-strictly, we argue that W_{ik} and W_{im} are i.i.d. for $k \neq m$ if G is large enough. Because the agent i spends a long time outside the queue G, when it returns, the queue's status has been updated many times, and the distribution of queue length at arrival should be i.i.d.

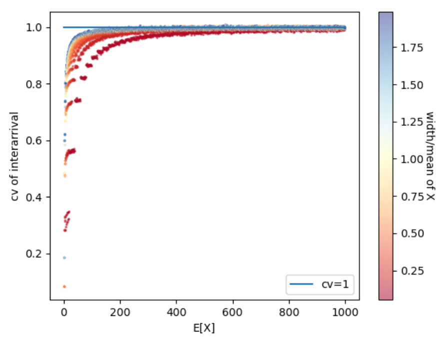


(a) All experiment results

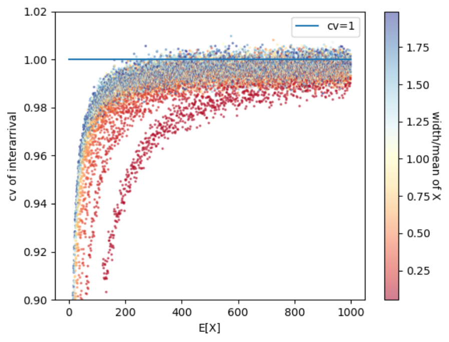


(b) Zoom in for $CV > 0.9$

Figure A.1: CV of inter-arrival times vs E[G] where G is normally distributed with different CV_X



(a) All experiment results

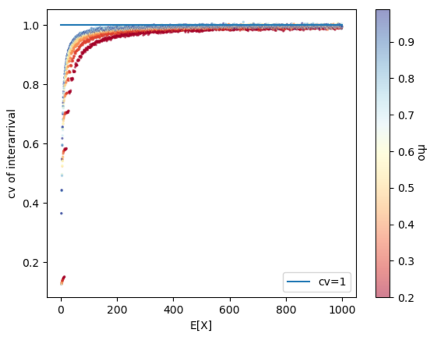


(b) Zoom in for $CV > 0.9$

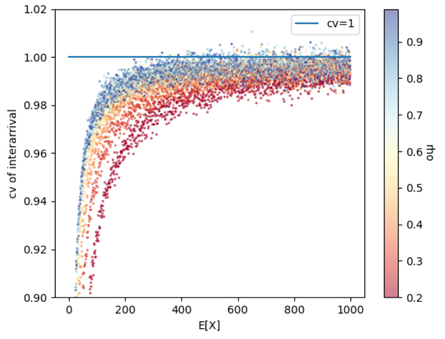
Figure A.2: CV of inter-arrival times vs E[G] where G is uniformly distributed with different CV_X

Therefore, A_{ik} is the k'th renewal time for a renewal process with an interarrival time of $G + X + W_{ik}$. Consider a time period $t_1 < t_2$. Let n_i be the number of arrivals of customer i from t_1 to t_2. As $X \to \infty$ in probability, $t_2 - t_1 < X + G + W_{ik}$ with high probability. Conditioned on $t_2 - t_1 < X + G$, we must have $n_i \in \{0, 1\}$. From the renewal theorem

$$\begin{aligned}\mathbb{P}(n_i = 1 | t_2 - t_1 < X + G) &= \lim_{K\to\infty} \mathbb{P}(n_i = 1 | t_2 - t_1 < X + G, k < K) \\ &= \lim_{K\to\infty} \frac{K(t_2 - t_1)}{\sum_{k=1}^{K}(X_k + G_k + W_{ik})} \\ &= \frac{t_2 - t_1}{W_0 + \mathbb{E}[X] + \mathbb{E}[G]}.\end{aligned}$$

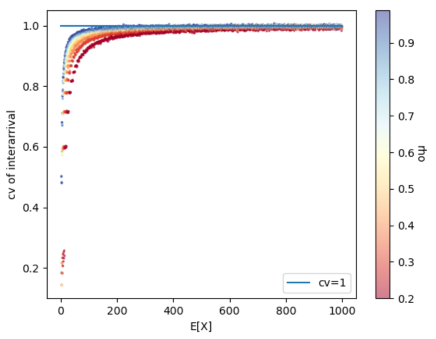


(a) All experiment results

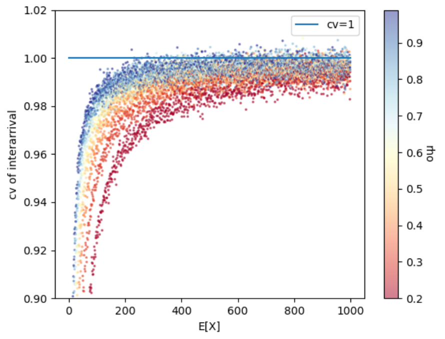


(b) Zoom in for $CV > 0.9$

Figure A.3: CV of inter-arrival times vs E[G] where G is normally distributed with different ρ



(a) All experiment results



(b) Zoom in for $CV > 0.9$

Figure A.4: CV of inter-arrival times vs E[G] where G is uniformly distributed with different ρ

The last inequality comes from the dominated convergence theorem since it's upper bounded by $\frac{t_2-t_1}{1/K(\sum_{k=1}^{K} X_k+G_k)}$.

Therefore, $n_i \to \text{Bernoulli}(\frac{t_2-t_1}{\mathbb{E}[X]+\mathbb{E}[G]})$ as $X \to \infty$ in probability. Let $N(t_1,t_2) = \sum_{i=1}^{R} n_i$ be the number of arrivals, $N(t_1,t_2) \to \text{Binomial}(R, \frac{t_2-t_1}{\mathbb{E}[X]+\mathbb{E}[G]+W_0}) \to \text{Poisson}(\frac{R(t_2-t_1)}{\mathbb{E}[X]+\mathbb{E}[G]+W_0})$. Therefore, the CV of interarrival $\to 1$ $\square$

Since the proof introduced more assumptions, we will show the Conjecture 1 numerically using simulation experiments.

The first set of experiments set X to be normally distributed. Let G be Uniform$(1,2)$, X be Normal(μ,σ), and define width $w = 6\sigma$ such that $X \in [\mu - w/2, \mu + w/2]$ with high probability. Run simulations for 1,000,000 arrivals at the first node, and set R according to

G	E[G]	std[G]	X	E[X]	std[X]	CV of inter-arrival time
deterministic	1.000	0.000	uniform	100.000	28.868	0.980
deterministic	1.000	0.000	normal	100.000	16.667	0.984
uniform	1.500	0.289	uniform	100.000	28.868	0.982
uniform	1.500	0.289	normal	100.000	16.667	0.978
normal	1.000	0.167	uniform	100.000	28.868	0.987
normal	1.000	0.167	normal	100.000	16.667	0.984

Table A.1: Simultion estimated CV for different G distributions with $\rho = 0.9$

the observed flow so that $\rho = v\mathbb{E}[G] \approx 0.8$. By changing μ from 2 to 1000, and w from 0 to 2μ, the result is shown in Figure A.1a (and Figure A.1b for the zoomed-in version that focuses on $CV > 0.9$). Each data point is the estimated result from one trail of 1,000,000 arrivals. The color of the data point represents the relative width w/μ. As we can see, no matter how we choose w/μ, CV increases to 1 as $\mathbb{E}[X] \to \infty$. A larger CV of X results in a faster convergence.

Similar results can be found by setting X to be Uniform($\mu - w/2$, $\mu + w/2$), where w is the width of the distribution. We did the same experiments and found the same result, that as $X \to \infty$ in probability, CV of the arrival interval will $\to 1$. (See Figures A.2a, and A.2b)

Then, we let X to be Normal$(\mu, \mu/6)$, and we change R so that utilization $\rho = v\mathbb{E}[G]$ increase from 0.2 to 0.99, μ from 2 to 1000. The result is given in Figures A.3a and A.3b. As we can see from the results, $CV \to 1$ no matter how we choose ρ, and larger ρ leads to faster convergence. Similar results can also be found when we set X to be Uniform ($\mu/2$,$3\mu/2$). (See Figures A.4a and A.4b)

We also do simulations for different G distributions. Let G come from uniform, normal, or deterministic. Do simulations with 1,000,000 arrivals and set R so that $\rho \approx 0.9$. In Table A.1, we can see CV all come close to 1 if $\mathbb{E}[X]$ is about 100 times of $\mathbb{E}[G]$.

Although there are some fluctuations in the previous experiments due to the estimation error in CV and the fact that R must be an integer so that we can only approximately set the value of ρ and v, the trend is clear: our conjectures hold when G is uniformly distributed/ deterministic/ normally distributed, and X is uniform or normally distributed. In our problem, G is a categorical distribution, which has a similar property as a uniform distribution or deterministic, X is a mixture of all path lengths, so it is similar to a uniform (if all paths have similar length) or normal distribution for a large system (if paths have different length, according to the central limit theorem). Therefore, our conjecture holds for our grid-based multi-robot problem.

Appendix B

Proof of Theorems for Warehouse Design

B.1 Proof of Proposition 3

Proof. **Proof of Proposition 3** Consider one internal workstation. Let ω be the time interval between two complete totes at the workstation. Note ω is the inter-departure time from an M/GI/1 queue with service time S. Let W be the waiting time of robots at the internal workstation.

Based on [39], equation (9),

$$\begin{aligned}\mathbb{E}[\omega] &= \frac{1}{\lambda}\\ \mathrm{Var}[\omega] &= \frac{1}{\lambda^2} + 2\mathrm{Var}[S] - \frac{2(1-\rho)\mathbb{E}[W]}{\lambda}\end{aligned}$$

With $\rho := \frac{\lambda}{\mu}$ close to 1. We can apply Kingman's formula to approximate waiting time:

$$\mathbb{E}[W] = \frac{\rho}{2(1-\rho)\mu}(CV_a^2 + CV_S^2)$$

Therefore, plug in $\mathbb{E}[W]$ into $\mathrm{Var}[\omega]$, we have

$$\begin{aligned}\mathrm{Var}[\omega] &= \frac{1}{\lambda^2} + 2\frac{CV_S^2}{\mu^2} - \frac{2(1-\rho)}{\lambda}\frac{\rho}{2(1-\rho)\mu}((CV_a^2 + CV_S^2))\\ &= \frac{1}{\lambda^2} + 2\frac{CV_S^2}{\mu^2} - (1 + CV_S^2\frac{1}{\mu^2}\\ &= \frac{CV_S^2}{\mu^2} + \frac{1}{\lambda^2} - \frac{1}{\mu^2}\end{aligned}$$

Based on Proposition 2, the departure interval is $\sum_{l=1}^{\xi}\omega_l$, where ω_l are i.i.d. with the same distribution of ω. We have

$$\mathbb{E}[\sum_{l=1}^{\xi}\omega_l] = \frac{\xi}{\lambda} \tag{B.1}$$

$$\mathrm{Var}[\sum_{l=1}^{\xi}\omega_l] = \xi(\mathrm{Var}[S] + \frac{1}{\lambda^2} - \frac{1}{\mu^2})) \tag{B.2}$$

Using the central limit theorem, for large ξ, the inter-departure time distribution is approximately Normal$(\frac{\xi}{\lambda},\ \xi(\mathrm{Var}[S] + \frac{1}{\lambda^2} - \frac{1}{\mu^2}))$ □

B.2 Proof of Theorem 2

Proof. **Proof of Theorem 2** For convenience, denote $\mathbb{E}[S_2] = ES_2, \mathbb{E}[S_1] = ES_1$. For $i \in I_2$, combine Equations 5.3 and 5.4, we have

$$CV^2_{i,\text{Arrival}} = \frac{1}{Q_i}\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{\xi}(1 + (Q_k ES_1)^2(CV^2_{S_1} - 1)]$$

where $I_{1,i} = \{k \in I_1, Z_{ki} > 0\}$ are the set of internal workstations assigned to the external workstation $W_i, i \in I_2$. Note $\rho_i = Q_i ES_2 = \frac{Q_i}{\mu_i}$, $\rho_k = Q_k ES_1 = \frac{Q_k}{\mu_k}$, the sojourn time

$$\begin{aligned}\mathbb{E}[D_i] &= Q_i\left(\frac{1}{\mu_i} + \frac{CV^2_{S_2}}{2\mu_i}\frac{Q_i}{\mu_i - Q_i} + \frac{\frac{1}{Q_i}\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{\xi}(1 + (Q_k ES_1)^2(CV^2_{S_1} - 1)]}{2\mu_i}\frac{Q_i}{\mu_i - Q_i}\right)\\ &= \rho_i + \frac{CV^2_{S_2}\rho_i^2}{2(1-\rho_i)} + \frac{\rho_i^2}{2(1-\rho_i)}\frac{1}{\xi}\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{Q_i}(1 + \rho_k^2(CV^2_{S_1} - 1)]\end{aligned}$$

The relative error

$$\begin{aligned}\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]} &= \frac{\frac{\rho_i^2}{2(1-\rho_i)}\frac{1}{\xi}|\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{Q_i}(1 + \rho_k^2(CV^2_{S_1} - 1)]|}{\rho_i + \frac{CV^2_{S_2}\rho_i^2}{2(1-\rho_i)}}\\ &= \frac{1}{\xi}\frac{\rho_i^2|\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{Q_i}(1 + \rho_k^2(CV^2_{S_1} - 1)]|}{2\rho_i(1-\rho_i) + \rho_i^2 CV^2_{S_2}}\\ &= \frac{1}{\xi}\frac{\rho_i|\sum_{k\in I_{1,i}}[\frac{Z_{ki}}{Q_i}(1 + \rho_k^2(CV^2_{S_1} - 1)]|}{2(1-\rho_i) + \rho_i CV^2_{S_2}}\\ &\le \frac{1}{\xi}\frac{\rho_i\sum_{k\in I_{1,i}}\frac{Z_{ki}}{Q_i}|(1 + \rho_k^2(CV^2_{S_1} - 1)|}{2(1-\rho_i) + \rho_i CV^2_{S_2}}.\end{aligned}$$

Note $\rho_k \leq 1$ and $\sum_{k\in I_{1,i}} \frac{Z_{ki}}{Q_i} = 1$, we have

$$\begin{aligned}
\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]} &\leq \frac{1}{\xi} \frac{\rho_i \sum_{k\in I_{1,i}} \frac{Z_{ki}}{Q_i} |(1 + \rho_k^2(CV_{S_1}^2 - 1)|}{2(1-\rho_i) + \rho_i CV_{S_2}^2} \\
&\leq \frac{1}{\xi} \frac{\rho_i \sum_{k\in I_{1,i}} \frac{Z_{ki}}{Q_i} (1 + |CV_{S_1}^2 - 1|)}{2(1-\rho_i) + \rho_i CV_{S_2}^2} \\
&= \frac{1}{\xi} \frac{\rho_i (1 + |CV_{S_1}^2 - 1|)}{2(1-\rho_i) + \rho_i CV_{S_2}^2}
\end{aligned}$$

Since $CV_{S_2}^2 \geq 0$, $\rho_i < 1 - \epsilon$, and $1 - \rho_i > \epsilon$, we have

$$\begin{aligned}
\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]} &\leq \frac{1}{\xi} \frac{\rho_i (1 + |CV_{S_1}^2 - 1|)}{2(1-\rho_i) + \rho_i CV_{S_2}^2} \\
&\leq \frac{1-\epsilon}{2\xi\epsilon} (1 + |CV_{S_1}^2 - 1|)
\end{aligned}$$

Therefore, if $\xi \geq \frac{1-\epsilon}{2\varepsilon\epsilon}(1 + |CV_{S_1}^2 - 1|)$, we have the relative error $\frac{|\mathbb{E}[D_i] - \mathbb{E}[\tilde{D}_i]|}{\mathbb{E}[\tilde{D}_i]} \leq \varepsilon$. $\square$