Multi-Agent Pickup and Delivery Problems in Uncertain Environments

Automation of hospital workplaces using autonomous transportation robots

Master Thesis

Lucas Brouwer

Multi-Agent Pickup and Delivery Problems in Uncertain Environments

Automation of hospital workplaces using autonomous transportation robots

by

Lucas Brouwer

Thesis Committee:	Dr. Javier Alonso-Mora
	Dr. ir. Carlos Hernández Corbato
Daily Supervisors:	Nils Wilde
	Luzia Knödler
Institution:	Delft University of Technology
Date of Defense:	20-12-2022
Place:	Faculty of Mechanical, Maritime and Materials (3ME), Delft

Cover Image: The autonomous transportation robot of Project Harmony [1]

Abstract

The aging of society, population growth and chronic under-investments in education of healthcare workers have all led to a major imbalance between the demand for healthcare and the supply of healthcare. Autonomous robots can be used to relieve overqualified healthcare workers of their repetitive transportation tasks, allowing them to spend more time on actual patient care, hence reducing this imbalance. One of the greatest challenges that must be overcome is the dynamic changes in the environment, which affects the traversability of corridors and thus travel times. This challenge also arises in other human-centric and urban areas where constantly arriving delivery tasks have to be completed such as autonomous mobility-on-demand systems, autonomous warehouses and same-day delivery services. In the hospital environment these dynamic changes occur naturally and can only be observed on-site, which makes them difficult to model. This thesis attempts to fill this gap in literature and therefore studies the online Multi-Agent Pickup and Delivery (MAPD) problem with binary, recoverable and stochastic blockages. The task is twofold in that both the tasks have to be assigned to the agents and the shortest paths for each agent have to be found. The stochastic, binary blockages result in parts of the environment becoming untraversable for a random amount of time. First, an observation model is created that records the blockages and uses the input parameters of the blockage model to estimate the current expected travel times of the edges affected by a blockage. These estimates are then incorporated into the graph used to compute the agents' routes, and an online centralized MAPD algorithm is derived. A waiting state is introduced that allows agents to wait at blockages instead of replanning their paths, and it is demonstrated why replanning is necessary after each observation. This thesis presents numerous experiments on different maps with various blockage parameter configurations to show that the proposed algorithm is more effective than naive algorithms even with noisy blockage parameter estimates. Furthermore, if the initial blockage model parameters are unknown we show that the maximum likelihood estimates can be used as inputs of the blockage model and still outperform the naive methods.

Lucas Brouwer
Delft, December 2022

Acknowledgements

None of the courses I took at TU Delft during my bachelor's and master's degrees piqued my interest in a field of research as much as the Planning and Decision Making course taught by Dr. Javier Alonso-Mora. After finishing this course, I knew I wanted to focus my master's thesis on task assignment and route planning. Fortunately, I was able to secure a position in the autonomous multi-robots research group. I began this journey in November 2021, and since then time has flown by. I would first like to thank my daily supervisor Nils Wilde, for answering my myriad questions and proofreading my work. I really enjoyed the whiteboard sessions in which we picked our brains until we finally came to a conclusion that we both agreed upon. His extensive knowledge, expertise and patience were truly essential to elevate this thesis to a higher level. Furthermore, I would like to thank Nils for his assistance in drafting the paper by revising it countless times. Next, I would like to thank Luzia Knödler, who made numerous contributions to my literature review and was always willing to lend a helping hand. I would also like to thank Dr. Javier Alonso-Mora for his constructive feedback throughout the entire process. Finally, I would like to thank my family, friends and girlfriend for always pushing me to achieve the best results and for their unwavering support.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ADP	Approximate Dynamic Programming
AP	Arrival Process
ALNS	Adaptive Large Neighborhood Search
CCTP	Covering Canadian Traveller Problem
CTP	Canadian Traveller Problem
C-TSP	Colored Travelling Salesman Problem
DVRP	Dynamic Vehicle Routing Problem
FCFS	First-Come-First-Serve
NS	Number of Servers
IA	Instantaneous Assignment
ILP	Integer Linear Programming
k-CTP	k-edges Canadian Traveller Problem
MAPD	Multi Agent Pickup and Delivery
MAPF	Multi Agent Path Finding
MATA	Multi Agent Task Allocation
MRTA	Multi Robot Task Allocation
msTSP	Multiple Steiner Travelling Salesman problem
NP	Non-Polynomial
sTSP	Steiner TSP
TS	Tabu Search
TSP	Travelling Salesman Problem
ST	Service Time
USZ	Universitätsspital Zürich
VRP	Vehicle Routing Problem

List of Variables

Variable	**Explanation**
Environment	
$G = (V, E, d)$	A connected, undirected, weighted graph containing vertices and edges
$V = \{1, 2, 3, ..., n\}$	The set of vertices, where each vertex is described by a x coordinate and a y coordinate
$E = \{(i, j) : i, j \in V, i \neq j\}$	The set of edges, where each edge has a positive nonzero weight
$d(e_{i,j})$	The corresponding weight of edge i,j, where i denotes the starting vertex and j denotes the destination vertex
$P_{s,g}$	A path starting at vertex s and ending in vertex g, containing the sequence of edges and vertices connecting s and g
t_h	Time horizon, the end time of the experiment.
Blockages	
$S(t)$	The state of each blockage location at time t
$B = \{1, .., b\}$	The total set of blockage groups
$E_b \subset E$	Subset of edges which denote the set of edges which can be blocked at blockage location b.
$\Phi_b(t) = \{0, 1\}$	A group of edges which can be blocked 0, or free 1, at time t.
$\phi_e(t) = \{0, 1\}$	All edges within $E_{0 \leq b \leq B}$ can be blocked 0, or free 1, at time t.
Fleet of agents	
M	Total amount of agents
$R = \{r_1, ..., r_m\}$	Set of agents
r_x	Single agent x
r_s	The speed of each agent
C	Maximum capacity of each agent
$c_m(t)$	Load of agent m at time t, where $c_m(t) \leq C$
$l_m(t)$	Location of agent m at time t
Tasks	
$\mathcal{D} = \{1, ..., k\}$	The total set of delivery tasks
D_k =	A single delivery task $\in T$, defined by a 10-tuple (see below)
{ id,	Task id
v,	Starting vertex
u,	Goal vertex
t_r,	Release time of the task (>0)
t_a,	Time at which the task has been accepted
t_s,	Time at which the task has been picked up
t_f,	Time at which the task has been finished (dropped of)
t_d,	The task deadline
$P_{v,u}$,	A list of the shortest set of nodes to reach u from v
pos_{log},	A list containing the positions per timestep of the path
c_k }	The load of the task which is $\leq C$
t_e	The earliest dropoff time is the sum of t_r and the length of pos_{log} (usually noted as the sum of t_r and the length of pos_{log} and therefore not included in the tuple)

List of Figures

1 Introduction

1.1. Background

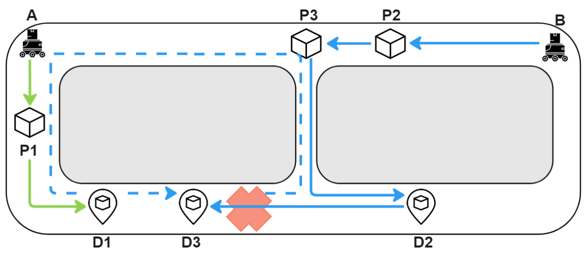
The United Nations has designated Tuesday, November 15th, 2022 as ”the day of 8 billion,” as the world population has reached this major milestone on this date. They call for action as the world’s population should do more than just exist. The entire world, not just the privileged part thereof, should thrive.[1] In order for the whole world to thrive many challenges have to be overcome. The first challenge is to provide sufficient food, proper housing and adequate healthcare for everyone. However, the increase in demand for healthcare has not been matched by an equal increase in supply. According to the World Health Organization, the shortage of healthcare workers will reach 10 million by 2030 mostly affecting the low- and lower-middle income countries.[2] The most obvious solution is to train more nurses and other healthcare workers, but this will require significant investments as well as a significant amount of time, both of which we do not have. The use of autonomous robots has the potential to drastically reduce the workload of healthcare workers and hence the shortage of healthcare workers.

On average, a nurse walks about 10 km per shift, mostly conducting ordinary pickup and delivery tasks [4]. This is a waste of their time, as they should be dealing with their patients rather than conducting mundane tasks that can be automated. Research has found that a single autonomous robot can perform a large number of transportation tasks, equivalent to 2.8 full-time jobs [34]. These robots can provide many types of transportation, from medicine to linen to meals to blood samples, and are able to work 24/7.

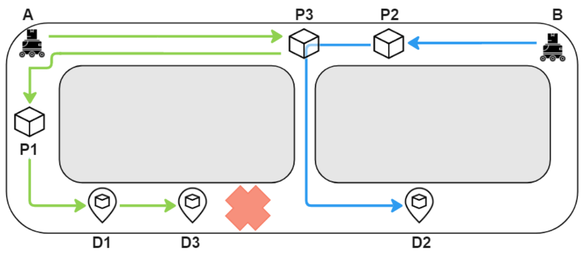
For these autonomous transportation robots to complete their tasks to the best of their ability, algorithms must be in place that handle task allocation and route planning of each robot. In a dynamic hospital environment, it is difficult to predict where and when these tasks must be performed. These types of problem in which multiple robots work together to complete constantly arriving and previously unknown pickup and delivery tasks are called online multi-agent pickup and delivery (MAPD) problems. In recent years, interest in these problems has increased, as applications such as autonomous mobility on demand, autonomous warehouses and same-day delivery services have gained in popularity [2, 8, 46]. The challenges remain the same, as previously unknown pickup and delivery requests must be fulfilled.

A common assumption in literature on the MAPD problem is that the agents know the time it takes to travel from location A to location B. Thus, not taking into account that many factors can influence the traversability of the paths the agents intend to take. In hospitals, hallway blockages may occur due to crowds, hospital beds being transported or due to stationary cleaning carts. In the urban environment, traffic, roadblocks, and accidents can all have a substantial impact on the agent’s travel times, just as robot failures do in autonomous warehouses. This thesis addresses the challenges of the multi-agent pickup and delivery problem in uncertain environments, with a focus on the hospital environment. By

[1] https://www.unfpa.org/8billion
[2] https://www.who.int/



(a) Reactive MAPD.



(b) Informed MAPD.

Figure 1.1: Problem instance of two agents (A,B) having to perform three pickups (P1-P3), with their coupled deliveries (D1-D3) in an uncertain environment. The red cross marks the location where blockages can occur. The solid lines portray the offline paths and if the reactive MAPD encounters a blockage it will travel along the dashed line.

addressing this problem, we can make progress towards a more sustainable and socially responsible healthcare system that provides adequate care to all people, regardless of their income or location.

1.1.1. Challenges of the MAPD problem in an uncertain environment

The uncertain environment presents many challenges to the already complex MAPD problem. One such challenge is to develop a mathematical model that captures the uncertainty of the real world. In the hospital setting, cleaning carts, crowds and hospital beds are just some of the factors that can cause differences between planned and actual travel times.

Another challenge is developing a MAPD algorithm that incorporates the uncertain environment model when computing routes and assigning tasks to agents. The algorithm should not only be able to compute the optimal current routes but should also consider the potential changes in travel times. As agents perform their tasks, they make numerous observations about the environment. An effective algorithm should take these observations into account when developing new routes and assignments.

Figure 1.1 depicts a MAPD problem instance where travel times may be affected by a blockage. A reactive MAPD algorithm plans the agents' paths as if travel times are constant, while an informed MAPD algorithm considers the potential blockage when planning its paths. In the case where the path is not blocked, the reactive MAPD produces a better result. However, if the path is blocked, the informed MAPD algorithm produces a better result, and the reactive MAPD algorithm may miss the deadline of D3.

Due to blockages, some parts of the environment can become temporarily unreachable. For example, elevators in hospitals can be out of service for an unknown amount of time, making it impossible for agents to complete tasks that span multiple floors. As a result, another challenge of the MAPD problem is that it must not only anticipate the possibility of the environment becoming disconnected, but also continue to execute its tasks in an efficient manner when this occurs.

1.2. Related Work

In this section, we first formally introduce the MAPD problem and describe some important characteristics of MAPD algorithms that are used to classify them. We then summarize a selection of state-of-the-art methods that solve the MAPD problem, followed by a brief review of how previous studies into routing problems have dealt with an uncertain environment. Lastly, we discuss the most relevant papers that address the MAPD problem in an uncertain environment.

1.2.1. MAPD

In the Multi-Agent Pickup and Delivery problem (MAPD), a fleet of agents must complete a stream of pickup and delivery tasks in a known environment. Tasks arrive continuously with previously unknown inputs for the pickup location, delivery location and release time, making it an online/lifelong problem. The problem consists of two parts: first, all tasks must be assigned to the agents, which is referred to as the Multi-Agent Task Allocation (MATA) problem. Second, the agents their paths must be computed, which is known as the Multi-Agent Path Finding (MAPF) problem [8]. In many cases, the MATA problem is performed before the MAPF problem, but the two problems are correlated and should be solved concurrently to improve the quality of the solution.

The MAPD problem can also be depicted as the online version of the vehicle routing problem (VRP) with pickup and deliveries. The VRP is an *NP*-hard problem and therefore the online version, which has to solve the VRP every time a new input arrives, is also *NP*-hard [22]. MAPD algorithms come in many variations, and in order to organize them, we compare them based on the following characteristics: single-agent vs. multi-agent paths and centralized vs. decentralized. The next paragraphs elaborate on these characteristics.

Single-agent path vs Multi-agent path problems
The majority of the MAPD problems can be divided into two groups:

1. **Single-agent path problems**, where the agents are autonomous robots/vehicles or drivers of vehicles and the routes that have to be scheduled only take their own paths and static/dynamic obstacles into account [2, 7, 36, 40, 42].
2. **Multi-agent path problems**, where the agents are likely to be warehouse robots and routes must be planned while accounting for inter-agent collisions [8, 24, 25, 33].

The methods used for solving the single-agent path problems differ from the multi-agent path problems as the multi-agent path problems require every agent's current route when computing a new path. In this master's thesis, we study the single-agent path problems and omit the multi-agent path problems.

Centralized vs Decentralized
When designing an algorithm to solve a routing problem, one of the first decisions to make is whether to use a centralized or decentralized approach. Initially, only the centralized approach was used, however, over the last decades, the decentralized approach has gained popularity [28]. The advantages and disadvantages of centralized and decentralized approaches are debated in the following paragraphs.

In the centralized approach, a single part of the entire system is responsible for managing all available resources [20]. This part can access all the available information and therefore easily compute an optimal solution. Many algorithms are developed from a centralized perspective rather than from a decentralized perspective. The downside of the centralized approach is that there is a single point of failure, as all communications and computations take place at one point [28]. Additionally, this central component is responsible for all calculations and broadcasts and therefore limits the scalability of the system. To overcome these problems, decentralized methods have been introduced in which no major element of the system is responsible for everything [38]. Decentralized systems are more robust and flexible, as they remain operational if one element fails and they can be scaled up more easily. Because of these factors, current literature favors decentralized approaches over centralized ones. Nettleton developed the formal definition of a decentralized system as [31]:

- There is no central agent/component required for the operation.
- There is no common communication facility; it is impossible to broadcast information to the entire team and only local communication between neighbours is aloud.
- The agents do not know the locations of every other agent, they are only aware of their local neighbors.

Recent papers also present hybrid solutions that attempt to incorporate decentralized elements into centralized systems [28].

1.2.2. Recent MAPD papers

Autonomous mobility on-demand (AMoD) systems are closely related to MAPD problems, as they too must handle online pickup and delivery requests. Lately, there has been a surge in interest in same-day delivery services and the dial-a-ride problem, resulting in more research on online MAPD problems [5, 19]. A state-of-the-art AMoD algorithm, that solves the MAPD problem on a large scale is the batch insertion algorithm of [2]. By first intelligently grouping requests that can be serviced together and matching them to vehicles close by, batch assignment can find solutions that efficiently use large capacity vehicles within practical computation time. Simonetto et al. provide a valuable alternative to the batch insertion algorithm by allowing only one new request per time period to be matched to a single vehicle [40]. This reduces the computation time, but at the expense of the quality of the solution, as combining multiple incoming trips to the same vehicle is no longer possible. Second, a federated (decentralized) architecture is proposed, which allows computation to be distributed among various sources using a linear assignment solver, reducing computation time while increasing communication.

1.2.3. Routing problems in an Uncertain Environment

The Canadian traveler problem (CTP) was one of the first to solve a routing problem in an uncertain environment [32]. In this scenario, a route between two vertices must be found in which the true cost of an edge is only revealed when the adjacent vertex of the edge is reached. A binary edge formulation is used as the edges can either be blocked or be unblocked. In the CTP of [32], a malicious adversary selects the edges to be blocked in such a way that the delay is maximized. Another popular form of the CTP is the stochastic CTP, where the blocked edges are chosen at random [16]. The Steiner-TSP with online edge blockages increases the number of goals that must be visited [48]. Furthermore, in contrast to this thesis, the blockages are non-recoverable, which means that the blocked edges remain blocked throughout the entire experiment, whereas we use recoverable blockages that can be traversed again after a stochastic amount of time. This problem has been extended to include multiple travellers and thus turning it into an extension of the vehicle routing problem [23]. Not only do blockages appear online in this thesis, but so do tasks consisting of coupled pickup and delivery locations in contrast to the known delivery locations of the multiple Steiner-TSP.

The authors of [30] do not use a binary formulation to simulate the uncertain environment, but present a different approach on how to handle the uncertain costs of the edges in a multi-agent task allocation problem. Here, the costs vary between an upper and a lower bound, rather than being blocked or free. In the risk-aware graph search of [9], all costs are drawn from an independent and identically distributed (i.i.d.) set of random variables. The true costs are only visible after arriving at the adjacent vertex, which also applies in our problem formulation. The agent has to find the fastest path to a final vertex, without knowing the costs of the edges. This strategy outperforms well-known heuristics, such as sampled A* and naive A*, by selecting paths with a high probability of low costs. In the colored traveling salesman problem, multiple TSP's have to be computed. This problem has been enhanced by introducing time-varying edge weights, transforming it into a dynamic problem [27]. Similarly to our work, the weights are varied according to a random distribution. Here, however, the weights are variable and we use a formulation with binary weights. A similar routing paper by [47] addresses the Multi-Robot Task Allocation (MRTA) problem while using queueing theory to model stochastic blockages. In this work, we consider the Multi-Agent Pickup and Delivery (MAPD) problem as opposed to the MRTA. Furthermore, we also show that if the inputs of the blockages are unknown maximum likelihood estimation can be used to estimate them. In addition, we also allow the graph to become disconnected for a stochastic amount of time by allowing the agents to wait at blockages.

1.2.4. MAPD in an Uncertain Environment

In most MAPD papers, tasks are the only dynamic element considered. However the online MAPD papers [14, 42] add additional dynamics. They encompass not only a dynamic environment, but also the possibility of vehicle failures and tasks being modified or even disappearing entirely. Ferruci & Bock's method, like ours, suggests changing only the weights of the busiest regions of the graph as it uses time-dependent variable costs for the main routes and highways [14]. Furthermore, they make use of a Tabu Search algorithm aided by a multiple stage neighborhood operator. This allows them to intensify and diversify the search space. Sun et al. use real traffic data to match vehicle speeds to time and place [42]. They develop an initial answer using a construction algorithm and compare the

Tabu Search technique with the adaptive large neighborhood search of [36]. Additionally, unassigned requests are handled via a dynamic insertion mechanism. Our proposed method does not depend on real data, but rather models blockages stochastically at the rate of blockage parameters. Moreover, we employ binary, recoverable cost changes, while [14] and [42] use variable formulations. To the best of our knowledge, our work is the first online MAPD problem that allows agents to wait at blockages as parts of the environment can become temporarily unreachable.

1.3. Research Objectives

This master's thesis aims to solve the challenges that arise when the environment in which the agents are travelling varies stochastically over time. This uncertain environment impacts the performance of the system, as the agent's ability to complete a task is affected by the changes in the environment. After conducting a thorough literature review, we found that very few MAPD problems include an uncertain environment and that even fewer include binary, recoverable blockages. Additionally, these types of blockages are also selected as they are well-suited to simulate blockages in a hospital setting. Thus, the first objective consists of effectively modelling an uncertain environment with binary, stochastic and recoverable blockages.

To the best of our knowledge, none of the works on the MAPD problem in an uncertain environment allows the environment to become disconnected. As a result, the second objective necessitates that the MAPD algorithm should continue to perform its tasks efficiently, even if the environment becomes disconnected. Finally, a MAPD algorithm must be designed that anticipates changes of the environment and thus outperforms reactive algorithms. To accomplish this, the algorithm should make use of all available information on the blockages, including the observations of the agents.

To encompass all of these objectives, the following research question has been formulated: "How can we efficiently assign pickup and delivery tasks to agents and compute their online routes in a dynamic environment consisting of stochastic, binary and recoverable blockages?"

1.4. Contributions

In this work, we use observations made by agents while traversing the environment to solve the online MAPD problem in an uncertain environment. The environment dynamics are modeled with stochastic, binary and recoverable blockages. We propose an online algorithm that actively recomputes the travel times of the parts that can become blocked. Furthermore, we make no assumptions about the connectivity of the environment, if agents are unable to track their paths due to blockages, we allow the agents to wait at these blockages. We show that our proposed algorithm outperforms naive algorithms even if noisy blockage parameter estimates are used. Lastly, if no initial information about the blockages is available, the blockage parameters are estimated by using the mean likelihood estimation technique.

1.5. Project Harmony

This master's thesis is conducted as part of Project Harmony, a collaboration of interdisciplinary organizations with the common goal to "Enhance healthcare with assistive robotic mobile manipulation." [3]. More specifically, the project seeks to aid healthcare workers by automating just-in-time delivery tasks. The project is located at the University Hospital of Zurich[4], which is why the map used to conduct the experiments is based on a floor plan of this hospital. To solve the problem efficiently, each organization is responsible for a different component of the problem. The planning and the scheduling of the tasks has been assigned to TU Delft. In this master's thesis we do not only aim to solve the task planning and routing, but also want to add a novel part: "How to assign tasks and compute routes in an uncertain environment".

[3]https://harmony-eu.org
[4]https://www.usz.ch/

1.6. Thesis Outline

The crux of this thesis is covered in Chapter 2, which is based on the scientific paper that will be submitted to the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023). Chapter 3 complements Chapter 2 as here, the parameters of the blockage model are estimated rather than known in advance, followed by Chapter 4, which describes and discusses additional approaches that did not significantly improve the quality of the solution. Next, Chapter 5 reflects on the main results in more detail. Lastly, Chapter 6 concludes the work and Chapter 7 offers suggestions for future work.

2 MAPD problem with stochastic blockages

This chapter covers the main chunk of this thesis as it is based on the paper that will be submitted to the IROS 2023 conference.[1] First, Section 2.1 provides the formulation of the problem, followed by Section 2.2, which defines the approach of this work. Section 2.3 introduces the developed MAPD algorithm and finally Section 2.4 covers the main implementation details and results.

2.1. Problem Formulation

Consider an environment that is described by a connected, undirected, weighted graph $G = (V, E, d)$ consisting of a set of vertices $V = \{1, 2, \dots\}$, where each vertex is described by a x-coordinate and a y-coordinate, and a set of edges $E = \{(i, j) : i, j \in V, i \neq j\}$. Edges have a positive nonzero weight $d(e_{i,j})$ which denotes the time it takes to traverse from vertex i to vertex j. A fleet of m homogeneous agents R = $\{r_1, r_2, \dots, r_m\}$ with capacity C must perform pickup and delivery tasks $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$, which arrive over time. Each delivery task $D = \{t_r, v, u, t_d\}$ consists of a release time t_r, the time at which the task becomes visible to the fleet of agents, a pickup v and a delivery vertex u in V and a deadline t_d before which the task must be completed. Furthermore, the environment is subject to a stochastic process X which determines the traversability of the edges. In particular, the stochastic process describes binary recoverable blockages that occur at predetermined sets of edges. The location of agent i at time t is marked as $l_i(t)$ and therefore its starting location is $l_i(0)$.

The quality of service for a single delivery task D is described by the cost function:

$$J(D) = \begin{cases} t_f - t_e & \text{if } t_f \leq t_d \\ M + (t_f - t_d)^2 & \text{otherwise,} \end{cases} \tag{2.1}$$

in which M is a large penalty, t_f is the time the delivery was finished, t_d the delivery deadline and t_e the earliest possible delivery time (see Figure 2.1). However, rather than focusing on a single delivery, we want to optimize the cost for all deliveries. So, let τ_i be a tour of agent i, which serves the set of deliveries $\mathcal{D}_i \in \mathcal{D}$. We can formulate the costs of this tour by summing the costs per delivery as follows:

$$c(\tau_i) = \sum_{D_j \in \mathcal{D}_i} J(D_j). \tag{2.2}$$

An assignment A maps agents to delivery tasks. We write this as a set of pairs between agents and delivery tasks, $A \subseteq \{(r_i, D_j) | r_i \in R, D_j \in \mathcal{D}\}$, with D_j being a single delivery. The assignment is complete if each delivery is exactly assigned to one agent. In addition, we note the total set of deliveries assigned to agent r_i as $\mathcal{D}_i$ and the tour that completes these deliveries as τ_i. The objective function

[1] https://ieee-iros.org/

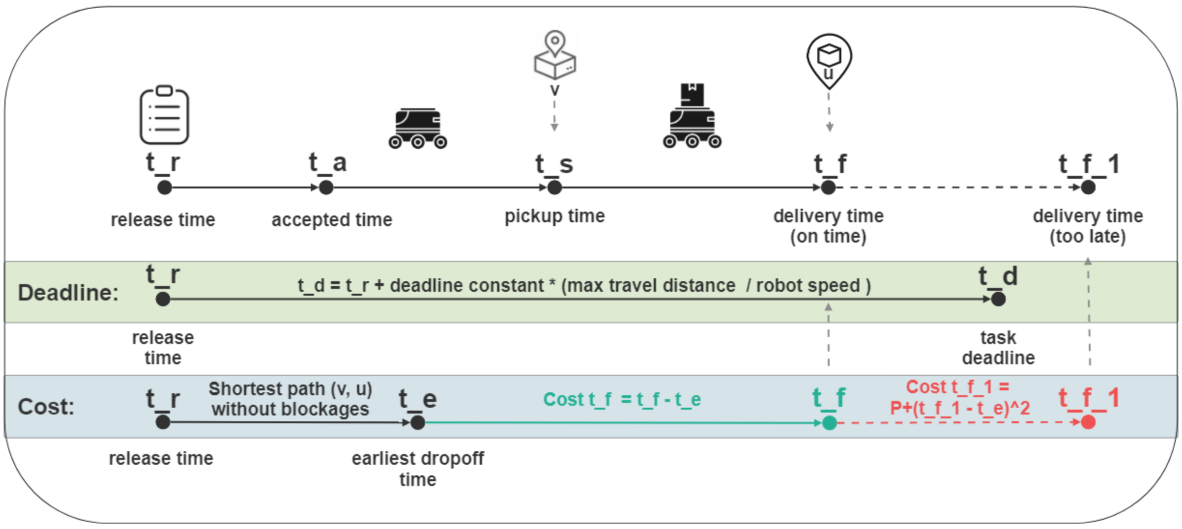


Figure 2.1: Computation of the cost and deadline

for the MAPD problem then becomes:

$$\begin{aligned} &\min_{A} \sum_{r_i \in R} c(\tau_i) \\ &s.t. \quad \tau_i \quad \text{serves delivery tasks} \quad \mathcal{D}_i(A), \\ &\mathcal{D}_1(A) \cup ... \cup \mathcal{D}_m(A) = \mathcal{D}. \end{aligned} \tag{2.3}$$

2.2. Approach

First, we model the uncertain environment, i.e., how blockages appear and disappear. Next, we derive an observation model for the blockages, which we use to compute the online edge costs of the graph.

2.2.1. Blockages

We model uncertainty in the environment with recoverable blockages, i.e., parts of the environment become untraversable for only a finite time. To define the locations of the blockages we let $B \subset 2^E$ be a collection of sets of edges. Each $b \in B$ describes a set of edges that are affected by the same blockage, i.e., this set of edges always becomes blocked and unblocked at the same time. Each blockage follows a stochastic process that regulates the traversability of the set of edges. Let $\Phi : B \times \mathbb{R}_{\geq 0} \to \{0, 1\}$ describe if blockage $b \in B$ is active $\Phi(b, t)$ = 0, or inactive $\Phi(b, t)$= 1 at time $t \geq 0$. If a blockage is active, all edges within the blockage set are removed from the graph G. We do not make any assumptions about the set of blockages B and thus allow G to temporarily become disconnected.

We model each blockage according to the $M/M/1/1$ queue [37]. By using this method we can model the blockages as independent random variables that follow a stationary process. Second, queueing theory has previously been utilized to model blockages [43], including rare blockages such as the Suez Canal blockage [17].

In a $M/M/1/1$ queue the arrival process (AP) is characterized by a Poisson point process with rate parameter λ, and the service time (ST) by another Poisson point process with rate parameter μ. The arrival process is used to model the arrival times of the blockages, and the service time gives us the amount of time the blockages remain active. Additionally, each blockage is modelled individually, so there is just one server, i.e., one blockage is processed at the time. There are no waiting lines as there can only be one blockage inside the system for each blockage. For each blockage location, we model the arrival process with λ, and the service time of the blockage with μ. To avoid queues from being generated, we alternate between the two Poisson processes until the experiment is finished.

2.2.2. Observation Model

When agents visit one of the vertices connected to the respective edges, they make observations about the states of the blockages. These observations are stored in a set $\Omega = \{(\Phi(b_1, t_1), b_1, t_1), (\Phi(b_1, t_2), b_2, t_2), ...\}$, which encapsulates the status of a blockage b being active $\Phi(b, t) = 0$ or inactive $\Phi(b, t) = 1$, the blockage b and the time of the last observation t. Due to the memoryless property of the $M/M/1/1$ queue we only record the last observation. Using Ω we can compute the probability that a blockage is active at time t, given that its last observation was at time t' as:

$$\mathbb{P}(\Phi(b,t) = 0|\Omega) = \begin{cases} p_b^1(t) \text{ if } \Phi(b,t') = 1 \\ \\ p_b^0(t) \text{ otherwise,} \end{cases} \tag{2.4}$$

where

$$\begin{aligned} p_b^1(t) =& \frac{\lambda}{(\lambda+\mu)}(1 - e^{-(\lambda+\mu)(t-t')}) \\ p_b^0(t) =& \frac{\lambda}{(\lambda+\mu)} + \frac{\mu}{(\lambda+\mu)} e^{-(\lambda+\mu)(t-t')} \end{aligned}$$

following the work of Rubino [37].

2.2.3. Online Edge Costs

The actual graph cannot be used to plan the paths of agents, as the edges are temporarily removed from the graph when a respective blockage is active. Therefore, we set up an online graph G_Ω, where we alter edge weights given the observation model, rather than deleting them. This graph never becomes disconnected as the edges are not deleted, but their weights are updated to incorporate the expected waiting time. This expected waiting time can be calculated for each blockage using the observation model which consists of the latest observations and the inputs of the blockage model, i.e. the $M/M/1/1$ queue.

First, let W be a random variable that depicts the waiting time at a blockage. When an agent is located at a vertex connected to one of the edges that can become blocked, it observes the state of this blockage. We can compute the current expected waiting time at this blockage given the properties of the $M/M/1/1$ queue and the observation.

$$\begin{aligned} \mathbb{E}(W|\Phi(b,0) = 0) = \mu^{-1} \\ \mathbb{E}(W|\Phi(b,0) = 1) = 0. \end{aligned} \tag{2.5}$$

If no agent is located at a vertex connected to blockage b, we can make use of the observation model to calculate the current expected waiting time of blockage b. The observation model gives us the probability that blockage b is currently blocked as $\mathbb{P}(\Phi(b,t)|\Omega)$ and the probability that it is free as $1 - \mathbb{P}(\Phi(b,t)|\Omega)$.

We employ the law of total expectation and equations (2.4) & (2.5), to obtain the current expected waiting time of blockage b as:

$$\begin{aligned} \mathbb{E}(W) =& \mathbb{E}(W|\Phi(b,0) = 0)\mathbb{P}(\Phi(b,0)|\Omega) + \\ & \quad \mathbb{E}(W|\Phi(b,0) = 1)(1 - \mathbb{P}(\Phi(b,0)|\Omega)) \\ =& \mu^{-1} \cdot \mathbb{P}(\Phi(b,t)|\Omega). \end{aligned} \tag{2.6}$$

Thus, we employ the observation model, the latest observations and the blockage parameters to predict the current expected wait times of each blockage location. Subsequently, we update the online graph, which is used to calculate the routes of the agents. In this online graph, for each edge that is in the set of edges of a blockage, we add the expected waiting time of this blockage to the cost of traversing the edge. For an edge e of blockage b the online edge cost follows:

$$d'(e,t,\Omega) = d(e) + \mu^{-1} \cdot \mathbb{P}(\Phi(b,t)|\Omega). \tag{2.7}$$

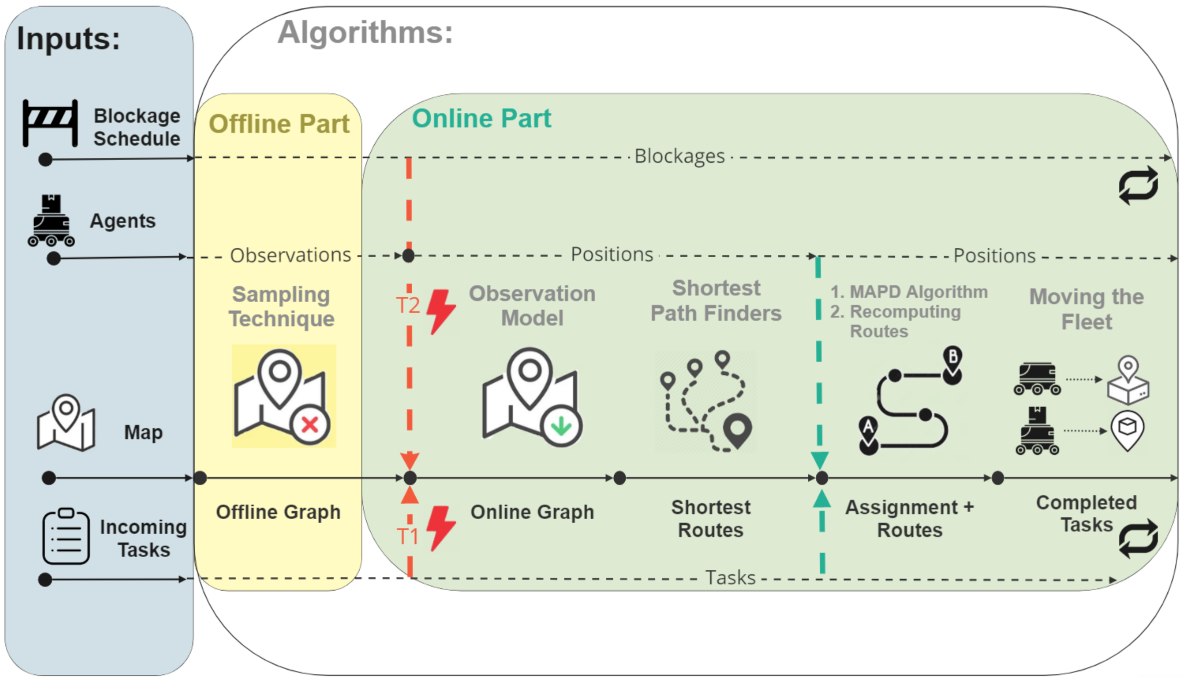


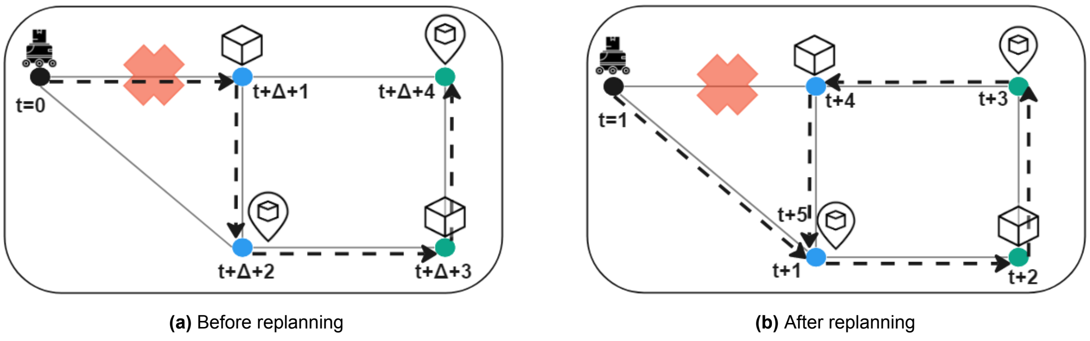
Figure 2.2: Overview of the MAPD problem

2.3. MAPD Algorithm

Throughout this section, we introduce a strategy to solve the previously stated MAPD problem. The goal is to use the agents' observations to ensure that each delivery task is assigned to the agent that leads to the highest service quality while the agents traverse along the most optimal routes. Algorithm 1 provides a structure on how to solve the problem, while Figure 2.2 portrays a schematic overview. First, at each timestep, all agents located at a blockage location make observations about the blockages, and the newly available tasks are pushed into the task queue Q. We update the online graph whenever at least one task is within the task queue and a rolling horizon period t_p has ended or an observation is made. A greedy assignment algorithm assigns the incoming tasks, while a cheapest insertion heuristic recomputes each robot's tour if an observation is made. Finally, at the end of each timestep, the fleet moves according to its (newly computed) routes.

Algorithm 1: MAPD with Stochastic Blockages

```
1  Input: Graph $G$, Task sequence $T$, Fleet $R$, Time horizon $t_h$, Observations $\Omega$, Rolling time
     horizon $t_p$, Time $t$, $A = \emptyset$ , $Q = \emptyset$
2  for time $t$ = 0 to $t_h$ do
3      $\Omega' \leftarrow$ Make_Observations($G, F$)
4      $Q \cup$ Retrieve_Tasks($T, t$)
5      if $t\%t_p = 0$ & $Q > 0 \vee \Omega' \neq \Omega$ then
6          $G_\Omega \leftarrow$ Update_Online_Graph ($G, \Omega', t$)
7          if $t\%t_p = 0$ & $Q \neq \emptyset$ then
8              $A, T \leftarrow$ Greedy_Assignment ($R, Q, G_\Omega$)
9          if $\Omega' \neq \Omega$ then
10             $T \leftarrow$ Recompute_Tours($R, G_\Omega$)
11     $R \leftarrow$ Move_Fleet($R, T, G, t$)
12     $\Omega = \Omega'$
```



(a) Before replanning **(b)** After replanning

Figure 2.3: A problem instance demonstrating the importance of constantly updating the most recent observation. The arrows represent the agents' most optimal tours. The blue delivery deadline is 10, the green delivery deadline is 5, and the expected waiting time at the blockage is Δ.

2.3.1. Computing Tours

Given a set of tasks, agents need to find a tour that minimizes the cost of service defined in Section (2.2). As the actual graph can become disconnected due to the blockages we compute the tours on the online graph which is always connected.

There are two scenarios in which we have to compute a tour. The first is when a new task has to be added to the agent's tour. In this case, we extend the ”cheapest insertion technique” of [39] by not adding a single vertex to the tour but by adding the pickup and delivery vertex at the same time with the constraint that the pickup vertex must be added first. The other constraint that also must be satisfied is the capacity constraint which cannot be exceeded. Due to the complexity of our cost function, we use the cheapest insertion technique rather than a TSP approximation algorithm.The second scenario is activated when an observation is made. Now, instead of simply adding a task to an existing tour, we completely replan the tour using the same insertion algorithm.

2.3.2. Waiting at Blockages

While agents are following their tours, they may run into blockages because the tours are planned on the online graph, which differs from the actual graph. As described above, a new tour is computed after a blockage has been observed. If the newly computed tour still traverses the blocked edge, the agent switches to a *waiting state*. In this state, the agent checks every timestep if the blockage has disappeared. Thus, at each timestep, a new observation is made and the routes are recomputed. This may seem unnecessary to recompute the tours while no new tasks have arrived nor are any new observations made. However, the optimal tour can change due to changes in edge weights of the online graph or to ensure that packages are delivered before their deadlines (see Figure 2.3). In practice, we do not replan at every timestep, but after a number of timesteps to reduce the computational burden or when the blockage has disappeared. As described in Algorithm 1 observations about other blockages or newly arriving delivery tasks can also alter the routes of waiting agents.

Figure 2.3 provides an example of why constant updating of the last observation is necessary in the waiting state. Here, an agent has to execute two deliveries with its starting location being at the left upper vertex. The green and blue deliveries have deadlines of $t = 5$ and $t = 10$, respectively. Let $\Delta = 0.5$ and let all edges be of length one. In the first scenario, $t = 0$, and so the minimum total time for delivering the packages is: $2.5 + 4.5 = 7$, in which both packages are delivered on time. The second scenario portrays the newly computed route that is assigned to the agent if the blockage has not disappeared at time $t = 1$. Now, a new optimal tour is computed to ensure that the green delivery is completed on time. In this case, the green delivery would not have been on time if the last observation had not been continuously updated and replanning had not been applied, resulting in a significant reduction in service quality.

2.3.3. Greedy Assignment

We now detail the assignment function from line 6 of Algorithm 1, which is fully specified in Algorithm 2. When a new task arrives, we must not only select the agent who incurs the least cost by accepting the task, but we must also determine the best order in which to perform the pickup and delivery actions. For this assignment we use a greedy assignment strategy that assigns the tasks one by one (see Figure 2.4). The algorithm iteratively selects the task with the earliest deadline and assigns it to the agent with the lowest cost. A task is rejected temporarily if the planning schedule of each agent is at its maximum capacity, i.e. the agents are hampered with a numerous amount of tasks.

Algorithm 2: Greedy Assignment

```
1  Input: Task Queue $Q$, Online Graph $G_\Omega$, Fleet $R$, Assignment $A$, Time $t$
2  Output: Assignment $A$, Rejected Tasks $Q'$
3  while Q is not empty do
4      $d_n \leftarrow$ Get_Earliest_Task($Q$)
5      for $r_i$ in $R$ do
6          $\tau_i$, c$_i \leftarrow$ Cheapest_Insertion($r_i, d_n, G_\Omega, t$)
7          $c^*, \tau^* \leftarrow$ Retrieve_Minimal_Cost($c_i, \tau_i, r_i$)
8      if $c^* < \infty$ then
9          $A \leftarrow$ Update_Assignment(r$^*$,$\tau^*$, $A$)
10     else
11         $Q' \leftarrow$ Add_Rejected_Task($Q'$, $d_n$)
12     $Q \leftarrow$ Remove_Task_From_Queue($Q$, $d_n$)
13 Return: $A$, $Q'$
```

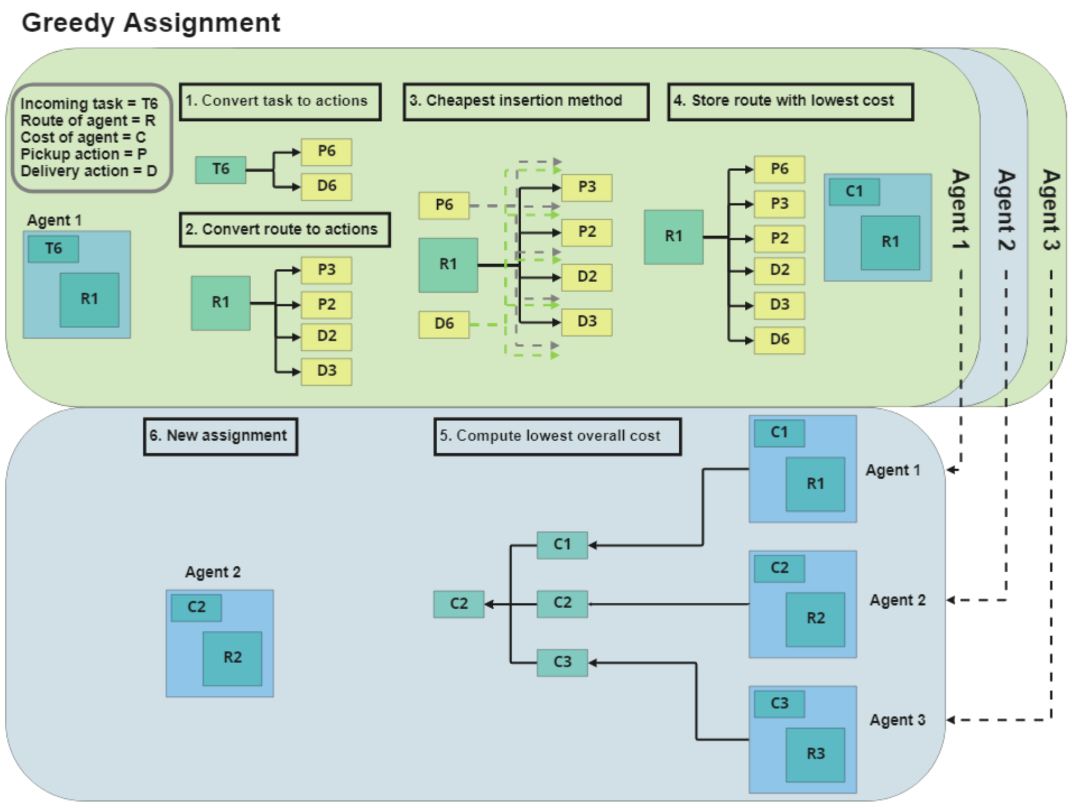


Figure 2.4: Greedy Assignment of a Task to an agent

2.4. Experimental Setup

Environment

We use two different environments to show that our *Proposed* algorithm can be applied to various maps. Figure 2.5 shows an environment based on a hospital floor plan and a second artificial map used for the experiments. The hospital environment spans over two floors, that are connected by vertices and edges with increased costs that represent elevators. Both environments feature multiple blockage locations. On the hospital map, four blockage locations were simulated and six on the artificial map (see Figure 2.5). For the scarce blockages experiment, the average blockage length μ^{-1} was varied to simulate short = 150 , medium = 200 and long = 250 blockages. The average arrival rate between blockages was kept constant at $\lambda^{-1} = 700$. The service rate and arrival rate were both varied in the frequent blockages experiment by using the same input for both parameters ($\lambda^{-1} = \mu^{-1}$) to simulate short = 100, medium = 200, and long = 300 blockages that appear more frequent. Furthermore, before the first task can arrive, the blockage model is simulated for 1000 timesteps, allowing the blockages to be active at the start.

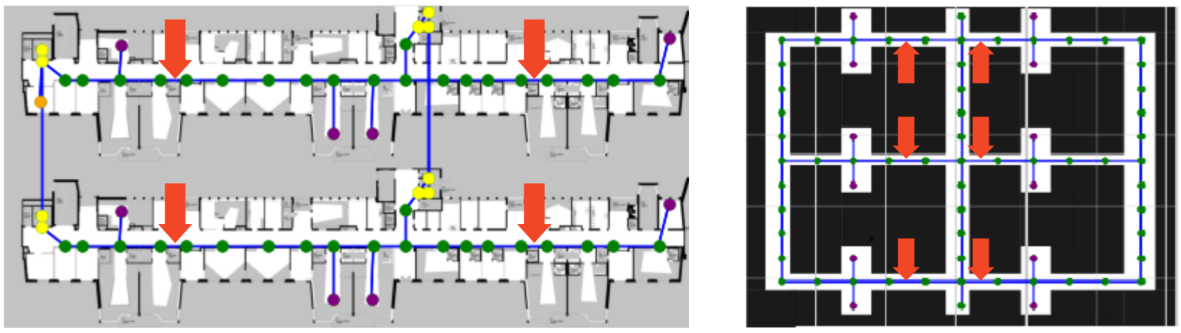


Figure 2.5: Maps of the environment, Left figure: 2 floors of the Hospital Map, right figure: artificial environment. Yellow vertices = elevators, purple vertices = pickup and delivery locations, orange vertex = hub. The red arrows indicate which edges can become blocked.

Tasks

The delivery tasks arrive online following a Poisson process. This stochastic process is commonly used in vehicle routing problems [6] and in queueing theory [35]. For each experiment four quantities of task loads (60, 80, 100, 120) were applied. Task loads indicate the expected number of tasks that must be completed by the agents; the actual number varies according to the Poisson process. For the hospital map, a hub and a set of pickup and delivery vertices are used, as shown in Figure 2.5. In our experiments, the pickup and delivery vertices are randomly drawn so that 75% of the generated tasks use the hub as the pickup or delivery location. The pickup and delivery vertices for the remaining tasks are drawn from a list of pickup and delivery vertices while $v \neq u$. For the artificial map, no hub is used. The deadline for a task is calculated using the release time t_r of the task, the maximum travel time max$_d$, which is the maximum travel time between any two vertices in V, and a deadline constant DC (5 in all experiments) according to: $t_d = t_r +$ max$_d \cdot DC$. The arrival times and deadlines of the tasks are within a 4000 timestep interval of (1000, 5000). Hence, new tasks are generated until a deadline of a task is outside this interval.

Remaining Parameters

The remaining parameters, such as the number of agents $m = 4$ and their capacity $C = 4$, are kept constant throughout the experiments. For each scenario, 40 repeats of experiments with different random seeds are conducted. Furthermore, after the task arrival interval is completed, an additional 5000 timesteps are simulated to ensure that each task is completed eventually.

Evaluation Criterion

The evaluation criterion used throughout this thesis is the rejection rate, which describes the percentage of tasks that have not been completed on time.

Lower Bounds
The experiments are designed such that, in the absence of blockages, the agents can complete all deliveries on time for all task load configurations. Therefore simulations have been run without any blockages, serving as a lower bound. Furthermore, we use an algorithm that immediately becomes aware of the locations and lengths of the blockages as they arrive, labelled *Revealed Blockages*. This algorithm serves as a tighter lower bound that we try to reach.

Naive Algorithms
We consider three naive algorithms: *Static*, *Optimistic* and *Pessimistic*. Each of the naive algorithms does not make use of the observation model. Therefore, the chance that a blockage is currently active is either 1 if the blockage was active at the last observation or 0 if the last observation of the blockage was inactive. If a blockage was active at the last observation each edge inside this blockage location is updated with

$$d'(e, t, \Omega) = d(e) + \alpha, \tag{2.8}$$

where α is a naive estimate of the waiting time. If the last observation was inactive the original edge weight is restored. The *Static* algorithm knows the average service rate of the blockage model and hence the naive estimate of the waiting time is equal to the average service rate of the observation model $\alpha = \mu^{-1}$. The *Optimistic* algorithm uses an overly optimistic estimate of the waiting time of just 1 timestep, assuming almost no waiting time, whereas the *Pessimistic* algorithm uses infinity, assuming the blockages are not recoverable.

2.5. Results

2.5.1. Scarce Blockages

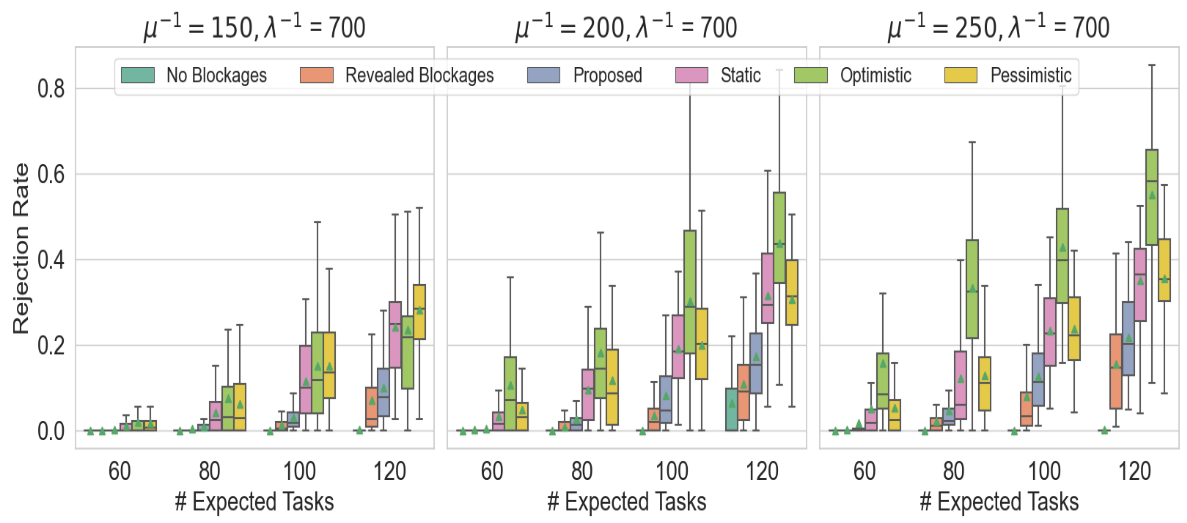
In this experiment, we vary the average time that a blockage remains active and compare our *Proposed* algorithm with the lower bounds and naive algorithms for scarce blockages with $\lambda^{-1} > 2\mu^{-1}$. The emphasis is on scarce blockages rather than on frequent blockages as we would expect in the hospital environment. Figure 2.6 shows the rejection rates of the different algorithms for the two maps.

Overall, the results portray that if *No Blockages* are implemented the expected 0% rejection rate is achieved, while the *Revealed Blockages* algorithm fails to complete all tasks in a timely manner but still performs second best. Our *Proposed* algorithm rejection rates are comparable to the *Revealed Blockages* algorithm rejection rates although the *Revealed Blockages* algorithm consistently produces better results.

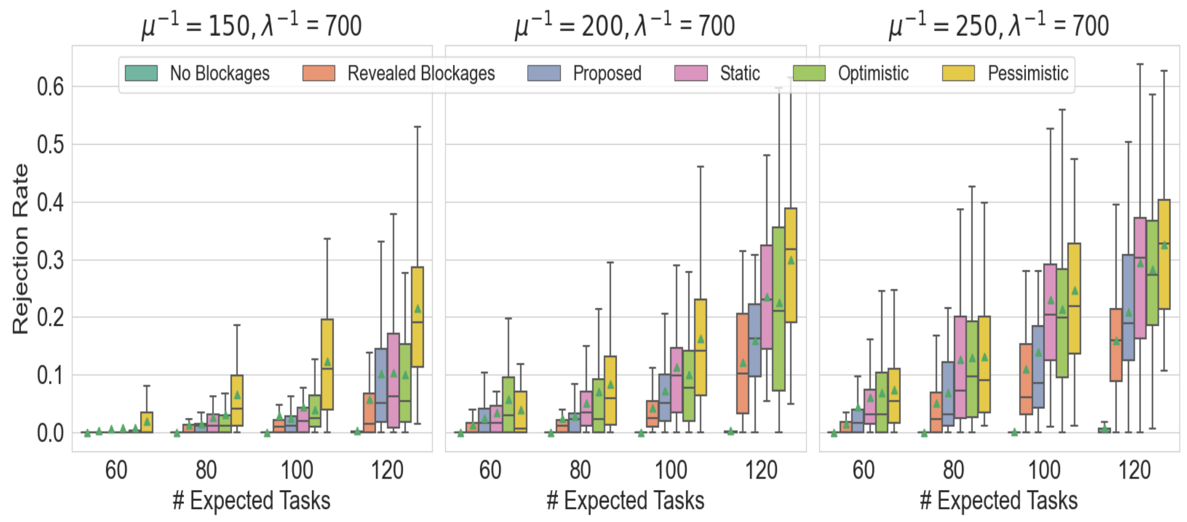
Across both maps our *Proposed* algorithm has average rejection rates ranging from 0 to 22%, while the *Static* average rejection rate reaches 35%, and the *Optimistic* and *Pessimistic* rejection rates even reach 55% and 36%, respectively. Furthermore, we discover that the gap between our *Proposed* algorithm and the *Revealed Blockages* algorithm is smaller compared to the gap between the naive algorithms and our *Proposed* algorithm. Hence, we conclude from the figures that our *Proposed* method outperforms or performs at least as good as the naive methods (*Static*, *Optimistic* and *Pessimistic*).

In the experiments conducted on the artificial map, we observe that our *Proposed* algorithm outperforms the three naive algorithms in all configurations of task loads and blockage parameters. In the experiments conducted on the hospital map, we see that when μ^{-1} = 150, our *Proposed* algorithm yields similar results compared to the *Optimistic* algorithm. This is due to the fact that in this scenario, waiting at a blockage is the preferred approach, and thus the *Optimistic* and our *Proposed* algorithms will often compute the same routes. As the average blockage time increases, the trade-off between taking a detour and waiting at a blockage becomes more difficult to solve. Hence, as μ^{-1} increases, the difference between our *Proposed* algorithm and the naive algorithms increases. Additionally, we observe that our *Proposed* algorithm is more effective on the artificial map compared to the hospital map. This can be explained by the fact that on the artificial map there are more alternative routes available and more blockages affecting these routes, making it more difficult to find the most optimal route.

In general, the *Static* and *Pessimistic* algorithms are rather risk averse, as they tend to avoid plan-



(a) Artificial Map



(b) Hospital Map

Figure 2.6: Results of the Artificial and the Hospital map with scarce blockages

ning through locations that were previously observed to be blocked. The *Optimistic* approach plans its paths as if there are no blockages and thus is extremely risk seeking. After a blockage has been observed, our *Proposed* algorithm avoids planning over this blockage (assuming that the blockage is sufficiently large). Over time the chance that this blockage is active decreases and therefore our algorithm will attempt to travel over the blockage again in order to find more optimal routes. Therefore, over time, our *Proposed* algorithm switches from being risk averse to risk seeking.

Furthermore, in both maps, but especially in the artificial one, we observe that as μ^{-1} increases, the average rejection rate of the *Optimistic* algorithm increases faster than the other naive algorithms. This is not surprising given that the average blockage time increases and that the *Optimistic* algorithm always waits at a blockage until it disappears. Because the artificial map has six blockages compared to the hospital map's four, the likelihood of an agent waiting at a blocked edge is greater, and the rejection rate of the *Optimistic* algorithm increases faster. On the hospital map, the cost of taking detours is higher than in the artificial map due to the large costs of the elevator edges. If the blockages are relatively short $\mu^{-1} = 150$ and the costs of taking a detour are relatively large, we observe that the difference between the *Pessimistic* algorithm and the other naive methods is the greatest (see left Figure 2.6b).

2.5.2. Frequent Blockages

In the previous experiment, blockages are rather rare than common as $\lambda^{-1} > 2\mu^{-1}$. This section presents additional experiments to show that our *Proposed* algorithm also outperforms naive methods if blockages appear more frequent. Figure 2.7 portrays the rejection rates when the average arrival interval between blockages is drastically decreased to $\lambda^{-1} = 100$, $\lambda^{-1} = 200$ and $\lambda^{-1} = 300$ while the service rate parameters are selected so that blockages are active 50% of the time. These results show that on both maps the *Proposed* algorithm still outperforms the naive methods. However, if the system becomes overloaded ± 40% rejection rate on the hospital map, we note that the differences between the naive and our *Proposed* method decrease.

Further, we observe a trend that as we increase both the arrival and service rates, the rejection rates increase, while the average time a blockage is active remains the same. Setting λ^{-1} and μ^{-1} to 1 results in an average wait time for agents that is nearly the same as if there were no blockages and therefore a low rejection rate. However, if we set λ^{-1} and μ^{-1} to infinity, any blockage that occurs during an experiment is likely to drastically increase the rejection rate.

The differences between the baseline methods our *Proposed* algorithm and the naive methods are larger in the artificial map and are rather small in the hospital map. This can be explained by the fact that taking detours is less likely to improve the performance of the algorithm on the hospital map due to the extra travel time caused by using the elevators. Moreover, agents on the hospital map have fewer possible detours to choose from when observing a blockage than on the artificial map where the number of possible detours is greater.

2.5.3. Noisy Blockage Parameter Estimates

To demonstrate the effect of noisy estimates of the service rate μ^{-1} and the arrival rate λ^{-1} on the performance of our algorithm, we repeat the experiment with $\mu^{-1} = 200$ and $\lambda^{-1} = 700$ on both maps. The average service rate has been overestimated and underestimated by increments of 50 timesteps, while the average arrival rate uses increments of 100 timesteps.

Service Rate

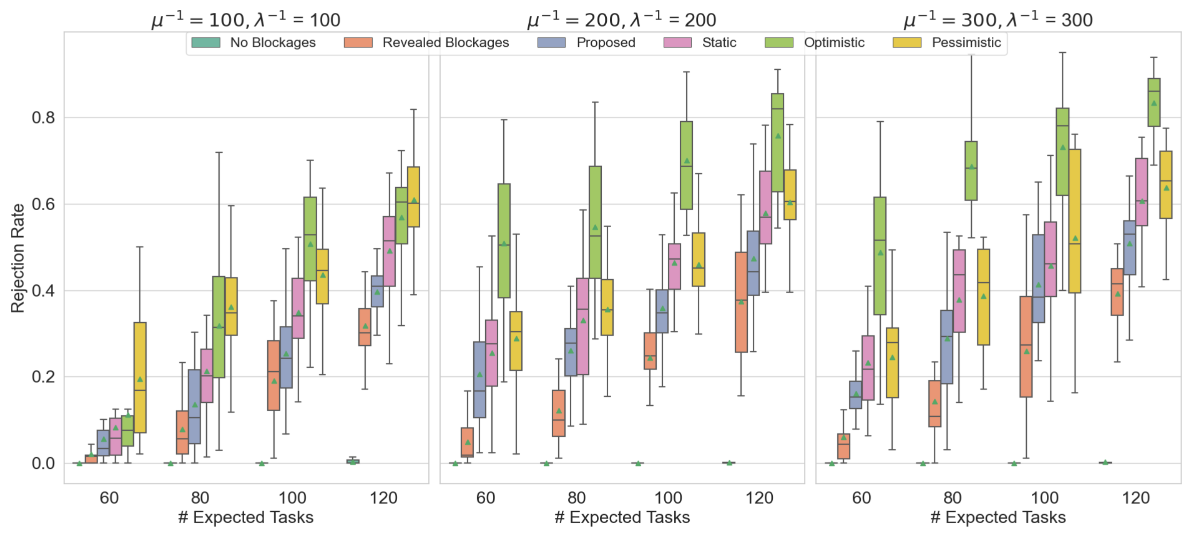
From Figure 2.8a, we note that, in general, over- and underestimates of the service rate have limited effect on performance and that the average rejection rate of the noisy estimates is lower than the naive methods of Figure 2.6a. The results of the hospital map in Figures 2.8b and 2.6b also portray this. Further, we observe that the wrongful estimates have a larger effect on the artificial map and that they almost perfectly portray the convex V shape that we would expect. This V shape is less visible on the hospital map results. This greater effect can be attributed to the fact that there are more potential detours on the artificial map, as well as more blockages for which the service rate is over- and underestimated.

Additionally, we find that the worst results are obtained when we heavily underestimate the service rate for lower task loads on the artificial map. However, in high task loads, over- and underestimates of the service rate do not appear to have a significant impact on performance.

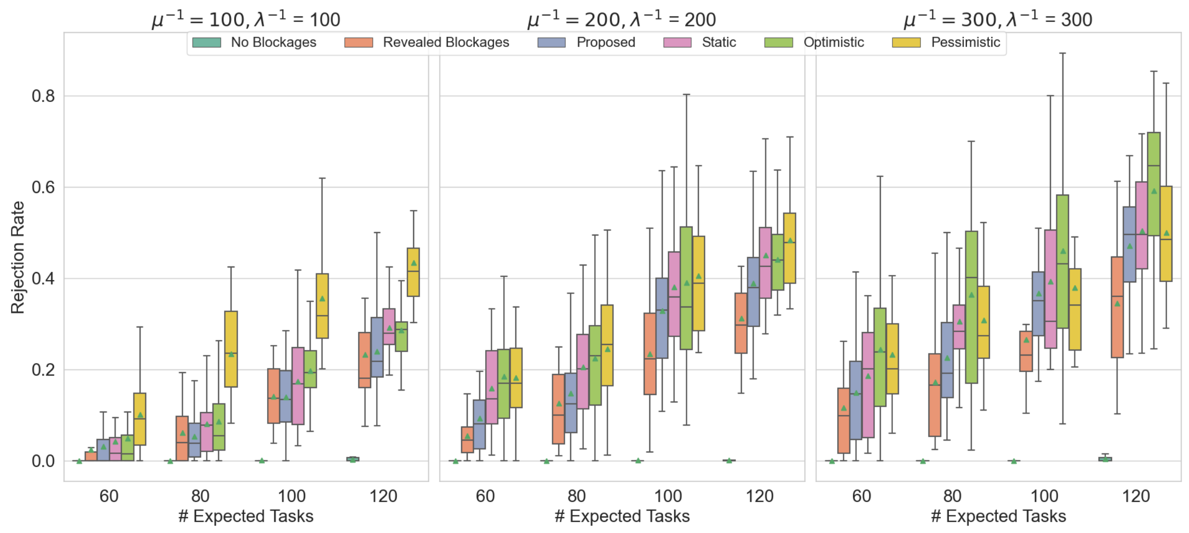
Arrival Rate

In Figures 2.8a and 2.8b, we see that over- and underestimating λ^{-1} appears to yield comparable results. Furthermore, decreasing the estimation of the arrival rate means that the algorithm expects more blockages and becomes more risk averse, and increasing the estimation leads to a more risk seeking algorithm. However, the convex V shape of the service rate of the artificial map is not found. No conclusions can be drawn from these results because the differences in average rejection rates remain within ± 3%.

Overall, we can state that the incorrect estimates of μ^{-1} and λ^{-1} have a small influence on the rejection rate of the system. The difference between the correct mean rejection rate and the over- and underestimates of λ^{-1} stayed within 3% and for μ^{-1} within 5%. The larger deviation of μ^{-1} can be attributed to the fact that μ^{-1} is increased and decreased by 50%$(100/200)$ and that λ^{-1} is increased or decreased by only 29%$(200/700)$.



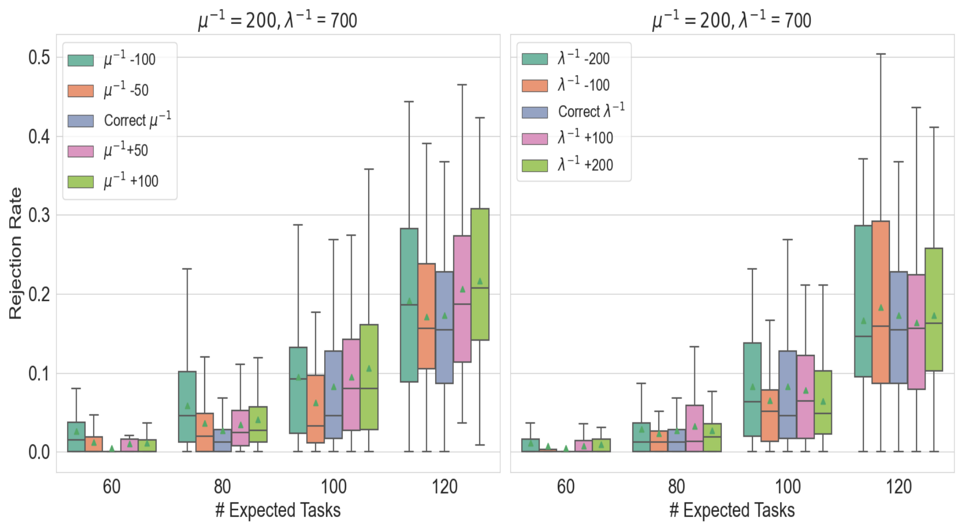
(a) Artificial Map



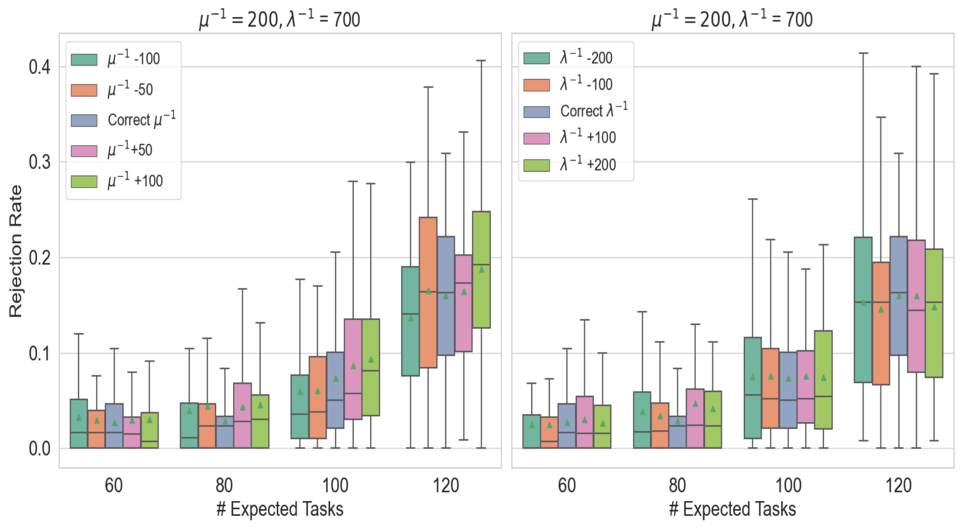
(b) Hospital Map

Figure 2.7: Increased arrival rate of the blockages

In our approach we assume that μ^{-1} and λ^{-1} are known, but these results show that small over- and underestimates of the blockage parameters do not drastically decrease the performance of the algorithm. Thus, even when one of the blockage parameters is incorrectly estimated, our *Proposed* algorithm performs better or at least as well as the naive methods.



(a) Artificial Map



(b) Hospital Map

Figure 2.8: Noisy blockage parameter estimates

3 Parameter Estimation

In the previous chapter, the parameters of the blockage model are assumed to be available, which is a strong assumption. These parameters may or may not be known in real-life experiments. Therefore, this chapter builds on the previous chapter by estimating the blockage parameters λ and μ rather than assuming that they are known. Section 3.1 introduces the technique for estimating the parameters, while Section 3.2 explores the maximum a posteriori probability estimation to increase the quality of the solution if some initial knowledge is known. Lastly, Section 3.3 describes the setup of the experiment and evaluates the results.

3.1. Maximum Likelihood Estimation

Over the years, many work has been done on estimating the parameters and states of queues [3]. Clarke was one of the first to estimate the parameters of an $M/M/1$ queue by using the maximum likelihood estimation [10]. Since then many research has been conducted on parameter estimation of the $M/M/1$ queue [29, 41], however few works specifically address the $M/M/1/1$ queue.

Before deriving the maximum likelihood estimates, we must first identify the observation scheme that can be used to estimate the parameters. In the experiments of Chapter 2, blockage locations are only observable when an agent is located at an adjacent vertex. Thus, neither full observation nor periodic observation schemes can be implemented as the times at which the agents traverse the blockage locations are unpredictable. However, because the agents have full communication, their observations on the same blockage location can be combined. Now, despite the fact that each blockage location follows its own independent blockage model, we can merge the estimations of all blockage locations because it is known that the input parameters are all the same and that they are independent of each other. Further, we assume that the observations are also independent of each other.

When an agent passes a blockage location, it records the status of the blockage (blocked or free) as well as the time of observation. Following each new observation, an interval is calculated that includes the elapsed time between the new observation and the previous observation as well as the two states of the observations. These intervals are classified into four types of intervals: blocked to blocked (b,b), blocked to free (b,f), free to free (f,f), and free to blocked (f,b). Then, for each interval we calculate the probability that the two states have been observed with the elapsed time as:

$$P(b, b, t_{int}) = \frac{\lambda}{(\lambda + \mu)} + \frac{\mu}{(\lambda + \mu)} e^{-(\lambda+\mu)(t_{int})} \tag{3.1}$$

$$P(b, f, t_{int}) = \frac{\mu}{(\lambda + \mu)} (1 - e^{-(\lambda+\mu)(t_{int})}) \tag{3.2}$$

$$P(f, f, t_{int}) = \frac{\mu}{(\lambda + \mu)} + \frac{\lambda}{(\lambda + \mu)} e^{-(\lambda+\mu)(t_{int})} \tag{3.3}$$

$$P(f, b, t_{int}) = \frac{\lambda}{(\lambda + \mu)} (1 - e^{-(\lambda+\mu)(t_{int})}) \tag{3.4}$$

We can compute the likelihood by making use of the assumption that all of the observations are independent of each other and thus the intervals are independent of each other. Hence, the likelihood function of the n intervals with times $(t_1, \ldots, t_n)$, previous states $(x_1, \ldots, x_n)$ and new states $(y_1, \ldots, y_n)$ is given as

$$\mathcal{L}(\lambda^{-1}, \mu^{-1}; (t_1, x_1, y_1, t_2, x_2, y_2 \ldots, t_n, x_n, y_n) = \prod_{i=1}^{n} P(x_i, y_i, t_i) \tag{3.5}$$

After having defined the likelihood function, the maximum likelihood estimates of λ and μ can be found with

$$\hat{\lambda}, \hat{\mu} = \underset{\lambda,\mu}{\text{argmax}} \prod_{i=1}^{n} P(x_i, y_i, t_i) \tag{3.6}$$

3.2. Maximum a posteriori probability estimation

When given an initial distribution of blockage parameters, we can use the maximum a posteriori probability (MAP) estimation method to obtain better estimates if this distribution is not uniform. First, the probability functions of λ and μ have to be gathered. If we assume that the probability density functions of λ and μ both follow a normal distribution. Then we obtain

$$\hat{\lambda}, \hat{\mu} = \underset{\lambda,\mu}{\text{argmax}} \mathcal{N}(\lambda_\mu, \lambda_{\sigma^2}))\mathcal{N}(\mu_\mu, \mu_{\sigma^2})) \prod_{i=1}^{n} P(x_i, y_i, t_i). \tag{3.7}$$

As the probability density functions of λ and μ are not given, they should be tuned to get the best results. After some initial tests without using MAP, we found that the arrival rate deviates heavily at the start of each experiment by taking values close to the minimum or maximum values of the range for $\lambda^{-1} = (1, 1000)$ see the left Figure 3.1. Furthermore, we observed that the estimates of the service rate take on values close to 1 at the start of each experiment. If the service rate is heavily underestimated, the agents make more observations on blockages as they attempt more routes over active blockages. In practise, the agents wait at the blockages until they disappear which gives them more information on the average blockage length than if the agents take detours right from the start of the experiment.

Now, in order to get rid of the large deviations of λ^{-1}, the MAP is initialized with a normal distribution for λ of $\mathcal{N}(500, 250)$. Since there is no prior information on the service rate and an underestimate is more desirable than an overestimate of the service rate, no normal distribution is applied for μ^{-1}.

3.3. Results

This experiment is conducted solely on the artificial map, while both the arrival and service rates of the blockages are varied from 100-700 with increments of 200. The proposed algorithm of Chapter 2 with the correct estimates is used as the lower bound of this experiment, whereas the naive method that we want to improve uses the proposed algorithm with an optimistic estimate of $\mu^{-1} = 50$ and $\lambda^{-1} = 500$.

In both the MLE and MLE + MAP methods, the service and arrival rates are not constant, but are updated every time a new route has to be computed. To minimize the computational effort, the maximum arguments of the service and arrival rates are calculated for each combination of μ^{-1} and λ^{-1} within the range of 1 to 1000, incrementing in 250 steps for both μ^{-1} and λ^{-1}. Therefore, when a route must be computed, the updated estimates of the service and arrival rates are obtained from equations (3.1) and (3.2) by evaluating all 250 x 250 combinations.

Not all experiments are portrayed and discussed as when the rejection rates of the lower bound become larger than 40% the system can be described as heavily overloaded and therefore no reasonable service quality can be achieved.

Now, let us examine the results by first looking at how the estimates of the service rate and arrival rates of the MLE and MLE + MAP develop over time in Figure 3.1. We observe that after applying the MAP, the large deviations of the arrival rates at the start of each experiment are removed. Additionally,

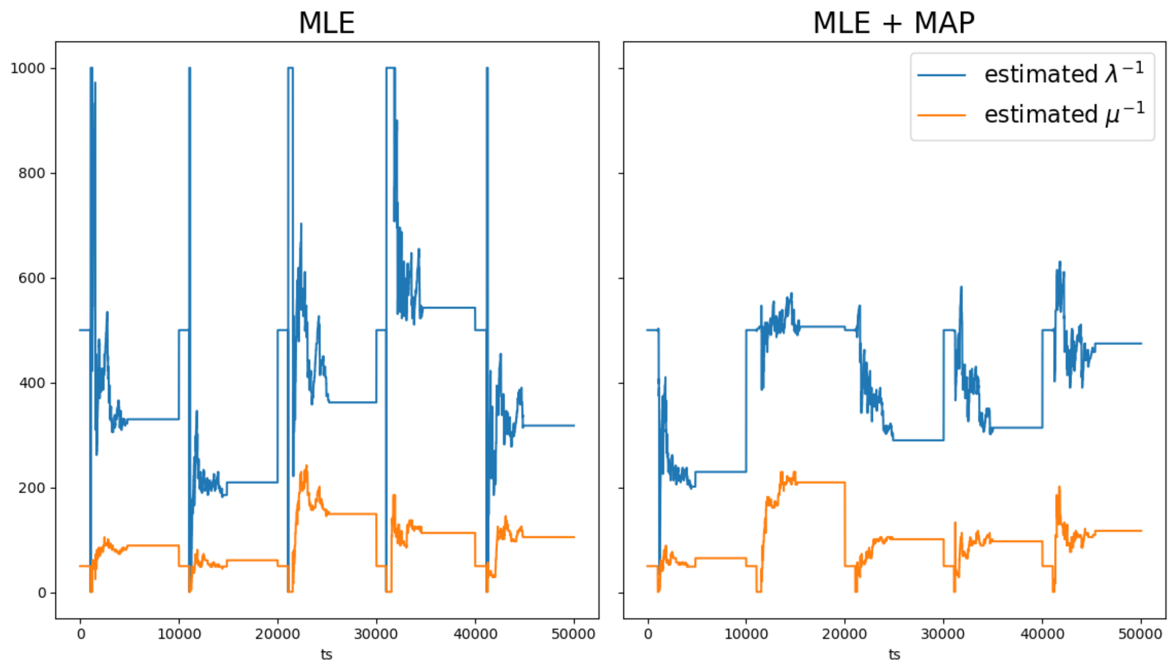


Figure 3.1: MLE and MLE + MAP estimates of 5 experiments (each experiment is 10k timesteps) on the artificial map with $\mu^{-1} = 100$, $\lambda^{-1} = 300$ and an average task load of 120

apart from the deviations, the average estimates for both the service and arrival rates are similar for both methods. When considering the performance of this experiment in Figure 3.2, we notice that the box plots for both methods largely overlap and that the average and median rejection rates are slightly higher for MLE + MAP.

After reviewing the results of Figures 3.2 - 3.4, we observe that MLE and MLE + MAP show a similar overall performance. This could be attributed to the large variation, since only ten experiments were conducted due to the large computational burden or because MAP does not significantly improve or decrease performance. Further experiments with different normal distributions should be conducted in order to support or refute these hypotheses.

In general, we notice that the optimistic estimates clearly performs worse than the MLE and MLE + MAP estimates in Figures 3.3 and 3.4, only in Figure 3.2 do the optimistic estimates achieve competitive results. As expected, the correct estimates outperform all methods in the large majority of experiments. If the service rate is small $\mu^{-1} = 100$ the optimistic estimates perform very similar to the MLE and MLE + MAP methods as the preferred strategy is to wait at the blockages. As the service rate of the blockages increases, the performance of the optimistic estimates begins to deteriorate significantly compared to the MLE and MLE + MAP methods. Now, taking a detour becomes more effective and therefore the optimistic method starts to perform worse. Furthermore, when the system becomes heavily overloaded and the rejection rates of the correct estimates surpass 40%, the differences between the rejection rates of MLE, MLE + MAP, and the optimistic estimates decrease. Figures 3.3 and 3.4 demonstrate this phenomenon with values of $\mu^{-1} = 300$ and $\lambda^{-1} = 500$ as well as $\mu^{-1} = 500$ and $\lambda^{-1} = 500$.

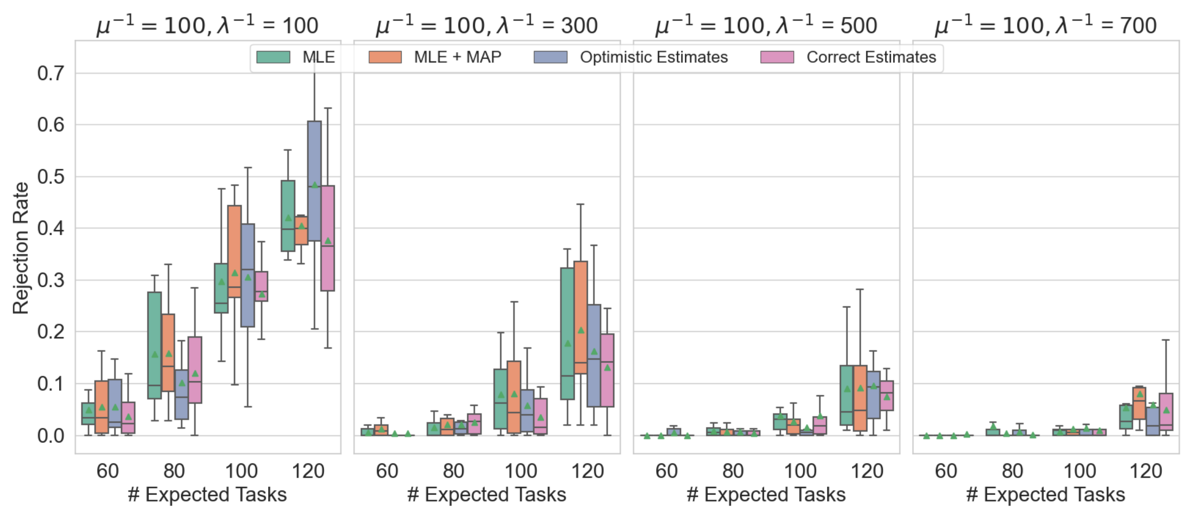


Figure 3.2: Parameter estimation on the Artificial map with $\mu^{-1} = 100$

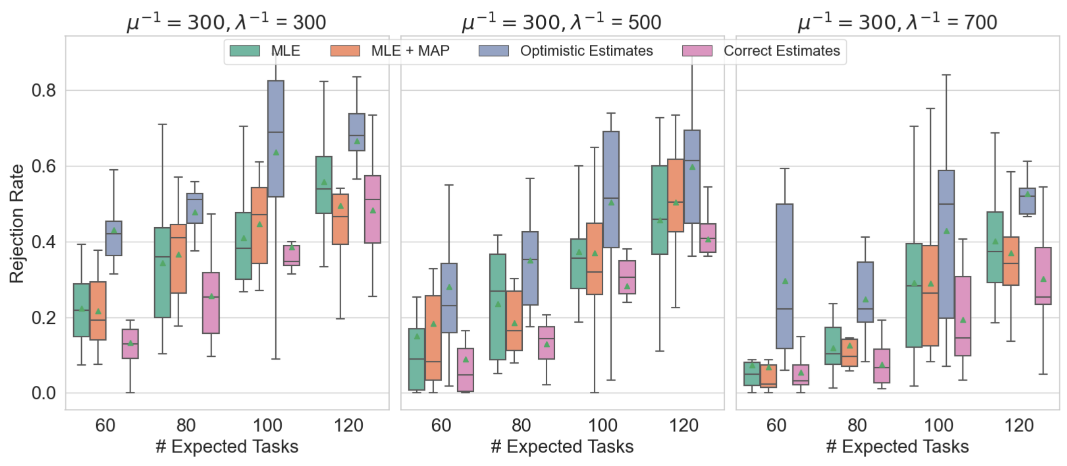


Figure 3.3: Parameter estimation on the Artificial map with $\mu^{-1} = 300$

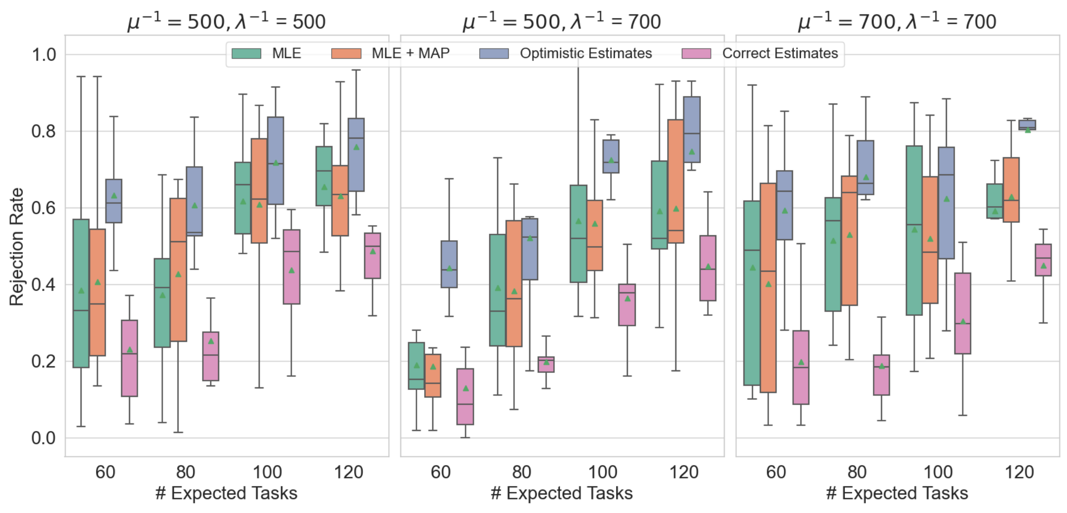


Figure 3.4: Parameter estimation on the Artificial map with $\mu^{-1} = 500$ and $\mu^{-1} = 700$

4

Additional Methods

This section discusses additional methods that were implemented but not included in Chapter 2. The batch assignment method is examined first followed by the single-step and the multi-step prediction models.

4.1. Batch Assignment

An integral component of the MAPD algorithm is the assignment of incoming tasks to agents. In this section, we investigate the batch assignment algorithm of [2] that may be used as an alternative to the greedy assignment method of Section 2.3.3. To this end, we explain the batch assignment process in Figure 4.2 and evaluate the results.

Batch assignment seeks to intelligently group requests to match them to agents close by. By making use of prepossessing steps large problems can be simplified, allowing them to be formulated as an integer linear programming problem, and thus making them solvable within reasonable time. We can examine the batch assignment of Figure 4.2 which involves two agents and four incoming tasks. First, we check if each agent can add an incoming task to its current route while still completing each delivery prior to its deadline. We use the cheapest insertion method, as described in Section 2.3.1, to add the a trip to the current route. After determining the feasible agent-task pairs, an agent trip graph (AT-graph) is generated (see Figure 4.2).

Next, we use the AT-graph to calculate all possible combinations of the new tasks that lead to feasible trips. For each agent, we assess the feasibility of all combinations of the incoming tasks that are inside the agents AT-graph. For instance, in Figure 4.2, we add tasks 3 and 4 to the agent-task pair A1-T1, using the cheapest insertion method. Task 2 is not added as A1-T2 is not found in the AT-graph of agent A1. We keep adding tasks until either the trips become unfeasible or all incoming tasks have been assigned. In practice, trips only become unfeasible if an agent is full, meaning that a large amount of tasks have already been assigned to an agent. Because there are no guarantees that any agent can complete an incoming task on time due to the uncertain environment, tasks are not rejected if their delivery time is scheduled after the deadline. To complete this step, we store all feasible trips with their associated costs.

Then, we formulate the ILP problem with the constraints of the problem, the cost function of (2.2) and a dummy robot that serves all unassigned tasks. This ILP is solved by making use of a Gurobi optimization solver[1]. Finally, we assign the tasks to the agents, and if a task has been assigned to the dummy robot, this task will be handled in the next assignment round.

Results

Figure 4.1 shows that the greedy assignment and the batch assignment results do not differ much. After carefully examining the results, we notice that the mean rejection rates of the batch assignment

[1]https://www.gurobi.com/

method are slightly higher, making the greedy assignment method preferable. We expect the batch allocation to outperform the greedy allocation as the number of tasks increases. However, the maximum number of tasks in the experiments of [2] and our experiments differ enormously and therefore cannot be compared (120 vs 460.000 tasks).

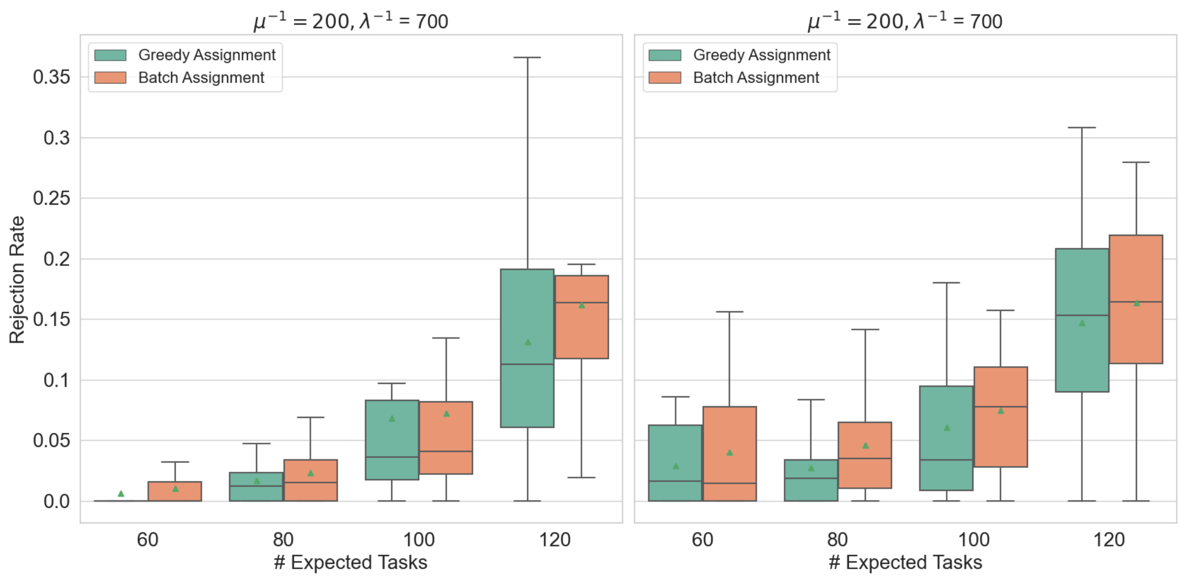


Figure 4.1: Comparison of the Greedy assignment and the Batch assignment, left figure shows the artificial map, right figure shows the hospital Map

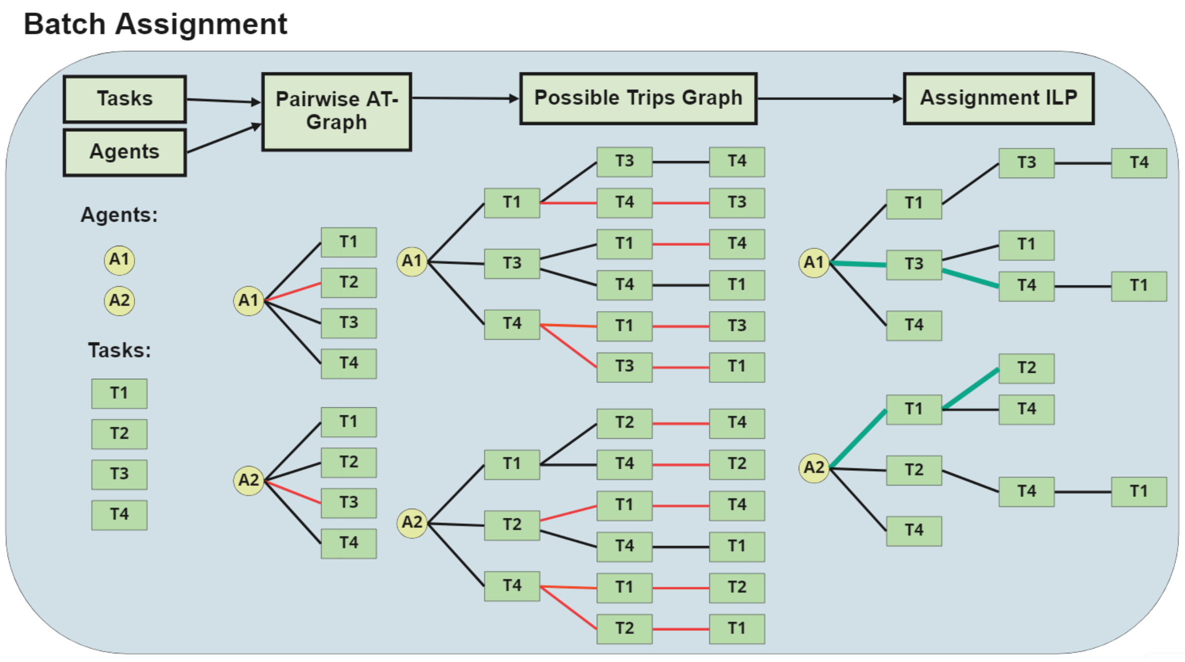


Figure 4.2: Batch assignment of four tasks to two agents

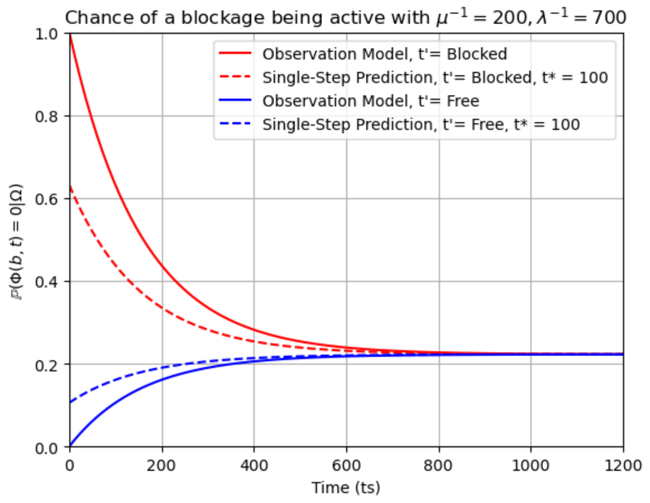


Figure 4.3: Chance of a blockage being active with $\lambda^{-1} = 700, \mu^{-1} = 200$, minimum travel time to the blockage is $t^* = 100$

4.2. Single-Step Prediction Model

The observation model of Chapter 2 is used in combination with the blockage model parameter inputs to estimate the current waiting time at each blockage. However, an agent can at most be adjacent to one blockage and must travel a bounded time before arriving at another blockage. Therefore, these estimations can become outdated while an agent is traveling towards the next blockage. Therefore, the prediction model is designed to provide a more informed estimate of the expected waiting time at each blockage for each agent individually. This more informed estimation necessitates the addition of a new input, namely, the location of each agent.

This additional input allows us to calculate the minimum bounded time for an agent to travel to each blockage, assuming that no blockages are active. This minimum travel time t^* is then incorporated into the observation model as follows:

$$\mathbb{P}(\Phi(b,t) = 0|\Omega) = \begin{cases} p_b^1(t) \text{ if } \Phi(b,t') = 1 \\ \\ p_b^0(t) \text{ otherwise,} \end{cases} \tag{4.1}$$

where

$$\begin{aligned} p_b^1(t) =& \frac{\lambda}{(\lambda+\mu)}(1 - e^{-(\lambda+\mu)(t-t'+t^*)}) \\ p_b^0(t) =& \frac{\lambda}{(\lambda+\mu)} + \frac{\mu}{(\lambda+\mu)} e^{-(\lambda+\mu)(t-t'+t^*)} \end{aligned}$$

This results in a more optimistic estimate whenever the last observation of a blockage was active and a more pessimistic one if the last observation was free. Figure 4.3 provides an example of how the chance of a blockage being active is influenced by the single-step prediction model if $t^* = 100$.

Finally, the online edge costs of Section 2.2.3 are updated for each agent individually rather than once for all. These additional calculations increase the computational burden, but due to the improved Floyd-Warshall algorithm described in Section A.4 this increase is minimal.

4.3. Mutli-Step Prediction

The planning horizon of the single-step prediction is limited to the near future. However, the planned routes generally span over a wider planning horizon. In the routes that are computed, a blockage location is likely to be traversed more than once. To account for this, the multi-step prediction model updates the expected waiting times of blockages each time a route travels over one.

For example, if we construct a route that plans to traverse a blockage location twice, first at $t = 150$ and then at $t = 350$, the expected waiting time (edge cost) at the first passage must differ from that at the second passage. If the last observation of this blockage was active the edge cost decreases and if the last observation was free the edge cost increases (see Figure 4.3). The observation model is still in place, but now t^* changes every time we plan to travel through a new blockage.

Previously, the improved Floyd-Warshall algorithm (see Section A.4), has been used to compute and store the shortest path of each pair of vertices alongside with their costs. However, storing every possible combination of shortest paths between any two vertices based on the most recent observations, each robots current position, and the scheduled times of passing blockage locations necessitates large amounts of storage and computational power. Furthermore, the MAPD algorithm does not require the shortest paths between each pair of vertices, but only between the agent's current location and the pickup and delivery vertices. Thus, a new algorithm should be implemented that only computes the shortest paths between these vertices and the agent's location while updating the edge costs of edges affected by the blockages at each passage.

A well-known algorithm used in many optimization problems to find the shortest path between two

Algorithm 3: Dynamically Weighted Dijkstra Algorithm

```
1  Inputs: $G = (V, E, d)$, source vertex $s$, destination vertex $d$, Observation Model $\Omega$
2  for every vertex $v$ do
3      Time($v$) $\leftarrow \infty$
4      Parent($v$) $\leftarrow NULL$
5  Time($s$) $\leftarrow NULL$
6  Push(0,$s$) $\rightarrow Q$
7  while $Q \neq \emptyset$ do
8      $u \leftarrow$ Pop_Vertex_With_Minimum_Time($Q$)
9      for every neighbor $v$ of $u$ do
10         $t_{int} \leftarrow$ Time($u$)
11         $alt \leftarrow$ Time($u$) + Travel_Time($u, v$,$\Omega$,$G$,$t_{int}$)
12         if $alt$ < Time($v$) then
13             Time($v$) $\leftarrow alt$
14             Parent($v$) $\leftarrow u$
15             if $v$ is $\neq d$ then
16                 Push($alt$, $v$) $\rightarrow Q$
17 Return: Time[], Parent[]
```

vertices is Dijkstra's algorithm [12]. However, the weights of the edges in this problem may not change over time. This issue is solved by the recently published algorithm of [45], Dynamically Weighted Dijkstra's algorithm, which allows the costs of the edges to vary over time. This algorithm start at a source vertex and iteratively visits neighbouring vertices with the lowest edge costs (travel times) until a destination vertex is reached (see Algorithm 3). The cost of traversing to a neighboring vertex v is computed with the observation model and t_{int}, which is the planned time it takes to reach the vertex u, whose neighbors are being explored. Most edges remain constant over time, as only the costs of the edges affected by the blockages are updated.

4.4. Comparison of Observation Model vs Prediction Models

Since the single-step and multi-step prediction models consider an additional input, we expect that they perform better or at least as good as the regular observation model. However, based on Figure 4.4, we must conclude that neither the single-step nor the multi-step prediction model outperforms the regular observation model. The following paragraphs discuss possible explanations for this behavior.

Single-Step Prediction Model

One possible explanation for the poor performance of the single-step prediction model could be that the single-step prediction model is too risk-seeking. For instance, when the last observation of a blockage is active, the expected waiting time at this blockage is lower than the expected waiting time of the observation model as a larger time interval is used in equation (4.1). On the other hand, the single-step prediction model computes a larger expected waiting time than the observation model if the last observation was free. Hence, when choosing between a ”free” and a ”blocked” route, the one-step prediction model is more likely to take risks than the observation model, and thus plans across ”blocked” edges more often. This is illustrated in Figure 4.3, since the difference between the ”blocked” and ”free” lines is smaller in the single-step prediction model than in the observation model.

If an agent attempts to traverse a previously blocked edge, but it is still blocked, the memoryless property of the algorithm calculates a 100% probability that the blockage is active, leading to an overestimation of the expected waiting time and therefore results in taking a detour, instead of waiting at the blockage. This phenomenon also occurs in the observational model, but is less likely to occur because it is more risk averse.

Multi-Step Prediction Model

When planning a route in the multi-step prediction model, the timestep increases every time we plan through a new blockage. Thus, after a certain amount of time, the expected waiting time of an active blockage becomes the same as the expected waiting time of a ”free” blockage. The planning in the nearby future can therefore be seen as informed whereas in the far future we add an expected waiting time to every blockage location independent of the last observation which can be seen as less informed. As a result, the multi-step prediction model generalizes too quickly, which could explain why the results are so poor. Second, the phenomenon previously explained above of the single-step prediction model also affects the multi-step prediction model. Therefore, the overall performance of the multi-step prediction model is worse than both the observation model and the single-step prediction model.

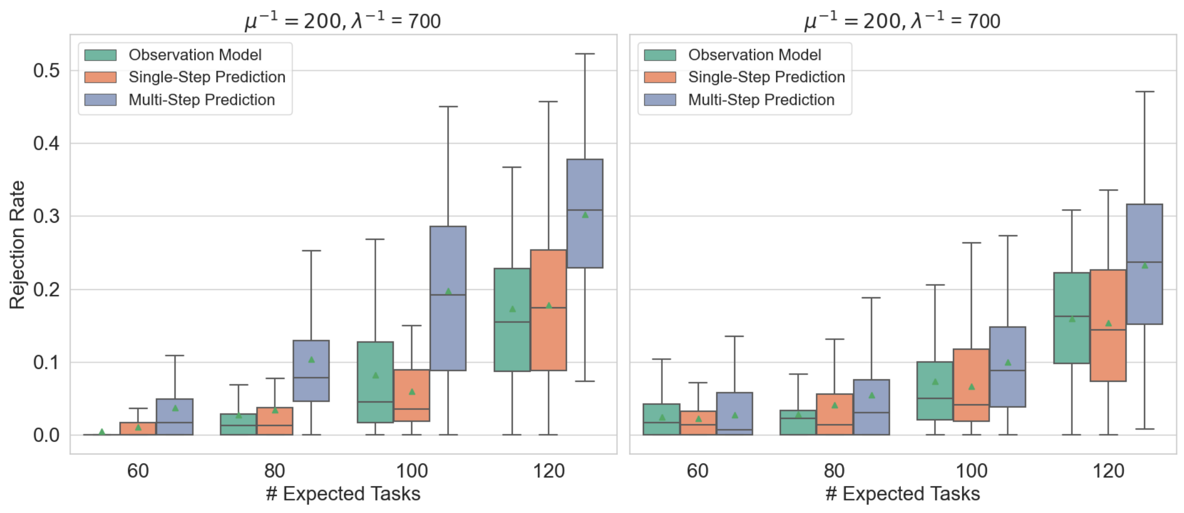


Figure 4.4: Comparison of the Observation Model and the Prediction models, left figure shows the artificial map, right figure shows the hospital Map

5 Discussion

We investigated the MAPD problem with stochastic, binary, and recoverable blockages throughout this thesis. First, an observation model was developed that records blockages and uses the input parameters of the blockages to estimate the current expected travel times of the edges affected by a blockage. Then, an online MAPD algorithm computes the shortest routes and assigns tasks to the agents, resulting in the highest service quality. Numerous experiments on multiple maps and with various blockage parameters have been carried out to demonstrate that our proposed algorithm outperforms naive algorithms. Furthermore, it is demonstrated that even when the inputs to the blockage model are unknown, the maximum likelihood estimation technique can be used to derive estimates that outperform naive methods.

During the experiments, the proposed method appears to anticipate the locations of the blockages by leveraging the observations and blockage parameters, mainly outperforming naive algorithms in terms of rejection rate. The small difference between the static, optimistic and proposed method on the hospital map for $\mu^{-1} = 150$, $\lambda^{-1} = 700$ and $\mu^{-1} = 100$, $\lambda^{-1} = 100$ can be attributed to the fact that for short blockages, taking a detour is unlikely to improve performance, so, therefore, the strategies and the results of the optimistic, static and proposed method are similar. These high detour costs can be explained by the fact that detours on this map must use the elevator.

In all experiments, the difference in the rejection rate between the proposed method and the naive methods increases as the average length of the blockages increases. When this happens, it generally becomes more advantageous to take detours and therefore we see that the performance of the optimistic method drastically decreases. The static and pessimistic approach favor ”free” alternative paths over ”blocked” paths, while our proposed method will over time explore ”blocked” paths if their expected travel time is less than the expected travel time of the ”free” paths. This time-dependent strategy leads to the discovery of more optimal paths. However, when an agent attempts to traverse a ”blocked” path and the blockage has not disappeared, the proposed algorithm disregards the last observation and treats this blockage as a new blockage. This may lead to taking a detour if one is available while waiting at the blockage would have been the preferred option.

The higher rejection rates of the single-step and multi-step prediction models of Chapter 4 can also be explained by the lack of memory in the system. These approaches also only take the last observations into account and disregard any observations made beforehand. Therefore, they too are unable to distinguish if a blockage is observed for the first time or that it has already been observed before. A memory-based system might be able to successfully distinguish this and thus outperform the memoryless system. In practice, the prediction models plan over ”blocked” paths faster than the proposed method, increasing the chance of observing the same blockage and incorrectly treating it as a new one. On the other hand, prediction models can also find more optimal paths if the previously observed blockages have already disappeared.

The use of the online graph and waiting state ensures that no assumptions need to be made regarding the connectivity of the graph. Hence, the agents always plan their paths on a connected graph even if the actual graph becomes disconnected. Further, experiments have shown that over- and underestimates of the blockage model parameters do not strongly affect the performance of the system. The parameter estimation model of Chapter 3 substantiates this claim since the estimates do not have to be perfect to outperform either the proposed method with optimistic inputs or the naive methods. As a result, the proposed method is quite robust and so can be applied to a wide range of applications even when the parameters are unavailable.

Lastly, using the batch allocation method from Chapter 4 instead of the greedy allocation method did not produce a discernible improvement in performance. This is most likely due to the fact that this method necessitates a greater number of tasks and agents before it a difference becomes noticeable.

6 Conclusion

The vast majority of research on multi-agent pickup and delivery problems considers a static environment. However, when this problem is solved in an urban or human-centric environment, agent travel times are affected by a wide variety of inputs. In order to complete the first objective of modelling an uncertain environment with stochastic, binary and recoverable blockages, a blockage model based upon the $M/M/1/1$ queue has been set up to mimic deviations in travel times. The $M/M/1/1$ queue has been picked because of its stationary, independent and memoryless characteristics.

To the best of our knowledge no MAPD algorithm allows the environment to become disconnected, therefore the second objective has been formulated to not only allow the environment to become disconnected but to also design a MAPD algorithm which anticipates this and that continues to execute its tasks efficiently when this occurs. In the actual graph, blocked edges are temporarily removed from the graph, disconnecting the graph and therefore it cannot be used for route planning. Hence, the online graph is used for route planning, in which blocked edges are not removed from the graph, but their weights are altered to represent the expected current travel times. The observation model leverages observations of agents and the inputs of the blockage model to compute these expected current travel times. Further, if an agent attempts to traverse an edge which is blocked, it computes the expected cost of waiting at the blockage and the expected cost of taking a detour. If the expected cost of waiting at the blockage is lower than the expected cost of the detour, the agent waits at an adjacent vertex until the blockage disappears. If the cost of the expected detour is lower, the agent traverses this detour. By continuously computing the most optimal routes on the online graph and allowing the agents to wait at blockages, the MAPD algorithm effectively executes its tasks even if the environment has become disconnected.

To fulfil the last objective of anticipating the blockages and outperforming reactive methods, the MAPD algorithm uses the observation model and the blockage model inputs to calculate the fastest routes and a greedy assignment method to find the most optimal allocations in terms of quality of service. By conducting a wide variety of experiments we showed that our proposed algorithm outperforms the naive methods in terms of rejection rate even if the inputs of the blockage model are over- and under-estimated. Additionally, a maximum likelihood estimation has been applied to obtain estimates of the blockage model parameters given the observations with stochastic intervals.

7 Future Work

Several improvements of the proposed algorithm are elaborated on throughout this chapter as well as some limitations of the experiments. First, only simulation experiments have been conducted, whereas real-life experiments should be performed to validate the results. Real-life experiments should also be carried out to reinforce or to overturn the assumptions that were made throughout this thesis, such as that blockages are independent of each other and that they usually occur in the same places. Furthermore, the actual tasks and their deadlines may have a significant impact on the number of agents required to complete the deliveries on time.

Since we expect more autonomous robots to be added to the hospital environment over time, field experiments should show whether it is necessary to consider the paths of other agents when computing a new path. At the same time, future efforts should seek to add decentralized components to the system to make it more scalable and less prone to the ”single point of failure problem” [28]. If the amount of tasks and agents is heavily increased we would expect the batch assignment method of Chapter 4 to start outperforming the greedy assignment method. Therefore, experiments with larger fleets and task sets should be carried out to confirm or refute this hypothesis.

One possible solution to improve the quality of the current results is to allow tasks to be reallocated over time as this is not possible while using the current MAPD algorithm. Therefore, the algorithm must be altered to allow tasks to be redistributed after they have been assigned. Additionally, if agents can exchange packages, they can collaborate to improve service quality. To enable this, new algorithms should be developed. In simulation experiments, however, this may seem beneficial, but in practice many other challenges may arise. Currently, a linear cost function is applied before the deadline, after which a quadratic cost function is used to ensure that delayed tasks are prioritized. Future research could look into different cost functions to increase the total number of on-time deliveries, include prioritized packages, or reduce the number of agents used.

To improve the current solution generated by the MAPD algorithm, different techniques can be applied, such as tabu search [18] and adaptive large neighborhood search [36]. These local-search meta-heuristics, have been applied in recent MAPD problems to enhance the quality of the solution if exhaustive search methods are not applicable. Sun et al. demonstrate that adaptive large neighborhood search improves solution quality more, but also requires more computational power [42].

In our approach, we first selected a queueing model and simulated blockages according to this model. Unlike this, the collected data from real experiments could also be used as a starting point to select a blockage model, as in [13]. Lastly, as explained in the discussion section, a disadvantage of the applied observation model is that it is memoryless and as soon as a new observation is made, the previous is discarded. Future studies can attempt to resolve this issue by adapting a new blockage model that includes memory and hence uses all/multiple observations. If the blockage model contains memory, we expect to get better estimates of the actual travel times. Overall, the performance of the MAPD algorithm therefore should also increase.

References

[1] Evan Ackerman. *Robots in Hospitals*. 2022. URL: `https://harmony-eu.org/` (visited on 03/31/2022).

[2] Javier Alonso-Mora et al. “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment”. In: *Proceedings of the National Academy of Sciences of the United States of America* 114 (3 Jan. 2017), pp. 462–467. ISSN: 10916490. DOI: `10.1073/pnas.1611675114`.

[3] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. “A survey of parameter and state estimation in queues”. In: *Queueing Systems* 97.1 (2021), pp. 39–80.

[4] Jakob E Bardram and Claus Bossen. “Mobility work: The spatial dimension of collaboration at a hospital”. In: *Computer supported cooperative work (CSCW)* 14.2 (2005), pp. 131–160.

[5] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. “Dynamic pickup and delivery problems”. In: *European journal of operational research* 202.1 (2010), pp. 8–15.

[6] Francesco Bullo et al. “Dynamic vehicle routing for robotic systems”. In: *Proceedings of the IEEE* 99.9 (2011), pp. 1482–1504.

[7] Andrea Camisa, Andrea Testa, and Giuseppe Notarstefano. “Multi-Robot Pickup and Delivery via Distributed Resource Allocation”. In: (Apr. 2021). URL: `http://arxiv.org/abs/2104.02415`.

[8] Zhe Chen et al. “Integrated task assignment and path planning for capacitated multi-agent pickup and delivery”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5816–5823.

[9] Jen Jen Chung et al. “Risk-aware graph search with dynamic edge cost discovery”. In: *The International Journal of Robotics Research* 38.2-3 (2019), pp. 182–195.

[10] A Bruce Clarke. “Maximum likelihood estimates in a simple queue”. In: *The Annals of Mathematical Statistics* 28.4 (1957), pp. 1036–1040.

[11] Robert B Cooper. “Queueing theory”. In: *Handbooks in Operations Research and Management Science* 2 (1990), pp. 469–518.

[12] Edsger W Dijkstra et al. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[13] James Dong and Ward Whitt. “Using a birth-and-death process to estimate the steady-state distribution of a periodic queue”. In: *Naval Research Logistics (NRL)* 62.8 (2015), pp. 664–685.

[14] Francesco Ferrucci and Stefan Bock. “Real-time control of express pickup and delivery processes in a dynamic environment”. In: *Transportation Research Part B: Methodological* 63 (2014), pp. 1–14.

[15] Robert W Floyd. “Algorithm 97: shortest path”. In: *Communications of the ACM* 5.6 (1962), p. 345.

[16] Dror Fried et al. “Complexity of Canadian traveler problem variants”. In: *Theoretical Computer Science* 487 (2013), pp. 1–16.

[17] Johannes Gast et al. “Analysis of the Suez Canal blockage with queueing theory”. In: *Adapting to the Future: How Digitalization Shapes Sustainable Logistics and Resilient Supply Chain Management. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 31*. Berlin: epubli GmbH. 2021, pp. 943–959.

[18] Fred Glover. “Future paths for integer programming and links to artificial intelligence”. In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[19] Sin C Ho et al. “A survey of dial-a-ride problems: Literature review and recent developments”. In: *Transportation Research Part B: Methodological* 111 (2018), pp. 395–421.

[20] Xiao Jia and Max Q.-H. Meng. “A survey and analysis of task allocation algorithms in multi-robot systems”. In: *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2013, pp. 2280–2285. DOI: `10.1109/ROBIO.2013.6739809`.

[21] David G Kendall. “Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain”. In: *The Annals of Mathematical Statistics* (1953), pp. 338–354.

[22] Allan Larsen. “The dynamic vehicle routing problem”. PhD thesis. Institute of Mathematical Modelling, Technical University of Denmark, 2000.

[23] Henan Liu, Huili Zhang, and Yi Xu. “The m-Steiner Traveling Salesman Problem with online edge blockages”. In: *Journal of Combinatorial Optimization* 41.4 (2021), pp. 844–860.

[24] Minghua Liu et al. “Task and path planning for multi-agent pickup and delivery”. In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2019.

[25] Hang Ma et al. “Lifelong multi-agent path finding for online pickup and delivery tasks”. In: *arXiv preprint arXiv:1705.10868* (2017).

[26] Amgad Madkour et al. “A survey of shortest-path algorithms”. In: *arXiv preprint arXiv:1705.02044* (2017).

[27] Xianghu Meng et al. “A Dynamic Colored Traveling Salesman Problem With Varying Edge Weights”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[28] Alejandro R Mosteo and Luis Montano. “A survey of multi-robot task allocation”. In: *Instituto de Investigacin en Ingenierła de Aragn (I3A), Tech. Rep* (2010).

[29] SP Mukherjee and Shovan Chowdhury. “Maximum likelihood and Bayes estimation in M/M/1 queue”. In: *Stochastic Modeling and Applications* 8 (2005), pp. 47–55.

[30] Changjoo Nam and Dylan A. Shell. “U3. When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time”. In: vol. 2015-June. Institute of Electrical and Electronics Engineers Inc., June 2015, pp. 1249–1254. ISBN: 9781479969234. DOI: `10.1109/ICRA.2015.7139351`.

[31] Eric Nettleton, Hugh Durrant-Whyte, and Salah Sukkarieh. “A robust architecture for decentralised data fusion”. In: *Proc. of the international conference on advanced robotics (ICAR)*. 2003.

[32] Christos H Papadimitriou and Mihalis Yannakakis. “Shortest paths without a map”. In: *Theoretical Computer Science* 84.1 (1991), pp. 127–150.

[33] Ana Carolina LC Queiroz et al. “Solving Multi-Agent Pickup and Delivery Problems Using a Genetic Algorithm”. In: *Brazilian Conference on Intelligent Systems*. Springer. 2020, pp. 140–153.

[34] Mohammed Owais Qureshi and Rumaiya Sajjad Syed. “The impact of robotics on employment and motivation of employees in the service sector, with special reference to health care”. In: *Safety and health at work* 5.4 (2014), pp. 198–202.

[35] A Ravi Ravindran. *Operations research and management science handbook*. Crc Press, 2016.

[36] Stefan Ropke and David Pisinger. “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”. In: *Transportation science* 40.4 (2006), pp. 455–472.

[37] Gerardo Rubino. “Transient analysis of Markovian queueing systems: A survey with focus on closed forms and uniformization”. In: *Queueing Theory 2: Advanced Trends* (2021), pp. 269–307.

[38] 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 (Jan. 2020), pp. 152–173. ISSN: 02786125. DOI: `10.1016/j.jmsy.2019.12.002`.

[39] Armin Sadeghi and Stephen L Smith. “Heterogeneous task allocation and sequencing via decentralized large neighborhood search”. In: *Unmanned Systems* 5.02 (2017), pp. 79–95.

[40] Andrea Simonetto, Julien Monteil, and Claudio Gambella. “Real-time City-scale Ridesharing via Linear Assignment Problems”. In: (Feb. 2019). DOI: `10.1016/j.trc.2019.01.019`. URL: `http://arxiv.org/abs/1902.10676%20http://dx.doi.org/10.1016/j.trc.2019.01.019`.

[41] Saroja Kumar Singh and Sarat Kumar Acharya. “Equivalence between Bayes and the maximum likelihood estimator in M/M/1 queue”. In: *Communications in Statistics-Theory and Methods* 48.19 (2019), pp. 4780–4793.

[42] Baofeng Sun et al. “Dynamic pick-up and delivery optimization with multiple dynamic events in real-world environment”. In: *IEEE Access* 7 (2019), pp. 146209–146220.

[43] Rik W Timmerman and Marko AA Boon. “The Fixed-Cycle Traffic-Light queue with multiple lanes and temporary blockages”. In: *arXiv preprint arXiv:2112.11292* (2021).

[44] Ismail H Toroslu. “Improving The Floyd-Warshall All Pairs Shortest Paths Algorithm”. In: *arXiv preprint arXiv:2109.01872* (2021).

[45] Piyush Udhan et al. “Vehicle Route Planning using Dynamically Weighted Dijkstra’s Algorithm with Traffic Prediction”. In: *arXiv preprint arXiv:2205.15190* (2022).

[46] Marlin W Ulmer et al. “The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times”. In: *Transportation Science* 55.1 (2021), pp. 75–100.

[47] N. Wilde and J. Alonso-Mora. “Online Multi-Robot Task Assignment with Stochastic Blockages”. In: *61st IEEE Conference on Decision and Control (CDC)*. Accepted, to appear. 2022.

[48] Huili Zhang and Yinfeng Xu. “The k-canadian travelers problem with communication”. In: *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*. Springer, 2011, pp. 17–28.

A Additional Implementation Details

This chapter elaborates more on why the $M/M/1/1$ queueing method is used as the blockage model. Additionally, the characteristics of the Poisson process are discussed and lastly, the technique for finding the shortest paths is provided, as it was omitted from the previous chapters.

A.1. Blockage Model

To understand how the blockage model has been reached, some basic principles of queueing theory are necessary. First and foremost, we must recognize that we are dealing with a single-station queuing system. In such a system, there is only one queue and in our system we also only use one server. To illustrate this system, consider a bank where one person enters the bank (waiting room) and checks to see if there is a customer at the counter (if the server is busy). If the queue is empty, the customer proceeds to the counter (in the server), otherwise the customer enters the waiting room.

To obtain the inputs of this queueing system random distributions have been applied, the first random distribution being the arrival process (AP), which tells us when the customers arrive. The second distribution is the service time (ST), this distribution tells us how long it takes for each customer to be served, i.e., the times when the customers leave the system. In our problem formulation the arrival process tells us when a blockage arrives and the service time tells us how long the blockage remains active. The arrival process and service time are the first two parameters of the Kendall notation, which was developed to efficiently organize queueingn models [21]. The other parameters are the number of servers used (NS), the capacity (CAP), being the total number of customers in the entire system, and finally the service discipline (SD). The service discipline describes the order in which customers are handled; in most cases, the first come first serve (FCFS) principle is used. In this master's thesis, we also use this service discipline. In the next section, we dive deeper into how we use the Kendall formulation to effectively design stochastic, binary and recoverable blockages.

A.2. M/M/1/1 queue

A well-known and widely used queueing model is the M/M/1 queue [11]. This single server queue possesses many of the characteristics that we require for modeling the blockages as it is stochastic, independent, and can be tuned so that the blockages are recoverable. However, this model allows multiple blockages inside the system as a newly arriving blockage is placed into the waiting room. This blockage will be processed right after the first blockage has disappeared. In real life, we do not expect blockages to influence each other and therefore blockages are not allowed to wait in the system. So, the capacity of our queueing model is set to one, to prevent queues from being formed, thus turning the $M/M/1$ queue into the $M/M/1/1$ queue.

Now, let us have a look at the random distributions of the $M/M/1/1$ queue. First, the arrival process of the blockages has to be set up. In the hospital environment, it is difficult to predict when exactly blockages will occur. Therefore, the exponential distribution has been picked, which is an independent and identically distributed (iid) process with blockages arriving via a homogeneous Poisson process. Additionally, the process is stable throughout the entire experiment and is memoryless. The Poisson process is also referred to as the Markovian and/or Memoryless process, resulting in the first M of the M/M/1/1 queue [35].

Second, comes modelling the service time, which in our case represents the time the blockages are active. Again, little information about this time is known beforehand, so another exponential distribution is used to sample the service times of the blockages. After the service time is completed, the blockage disappears. This gives us the second M of the $M/M/1/1$ queue. Both the arrival process and the service time parameters do not vary over time and therefore the $M/M/1/1$ queue is continuous, constant and independent.

Finally, by combining the arrival process, the service time and the single server with a capacity of one and applying Kendall's notation we arrive at the $M(\lambda)/M(\mu)/1/1$ queue or the $M/M/1/1$ queue. The blockages are created according to the $M(\lambda)$ Poisson process and subsequently the $M(\mu)$ Poisson process tells us how long the edges are blocked. We alternate between the two Poisson processes to ensure that no queue can be formed and that the maximum amount of blockages in the system is one.

A.3. Poisson Processes

As both the arrival process and the service time are determined by the Poisson process this section elaborates more on why this process has been picked. One of the primary reasons is the homogeneous Poisson process's simplicity, as it only requires the average time between events to be configured. This parameter is also known as the rate or density. Despite knowing the average time between events, the actual time between these events is sampled at random. The AP and ST are now sampled with positive mean rates, thus the values drawn from these distributions range from 0 to positive infinity.

Next, let us define the exponential distribution mathematically: A random continuous variable X has an exponential distribution with *rate parameter* $\lambda > 0$ if the probability density function PDF is stated as follows X exp(λ):

$$f_X(x) \equiv \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{A.1}$$

Furthermore, each random variable is accompanied by a cumulative distribution function (CDF). The CDF defines the probability that the random variable X is smaller than or equal to x. For the exponential distribution the CDF is given by:

$$F_X(x) \equiv \begin{cases} 1 - e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{A.2}$$

Now, after having defined the processes mathematically, the average amount of time between the arrivals of the blockages can be seen as $1/\lambda$ and the average service time is portrayed as $1/\mu$.

A.4. Improved Floyd-Warshall

In Chapter 2, Figure 2.2 provides an overview of the inputs and the algorithms used to solve the MAPD problem. Most of the algorithms have already been covered in the previous chapter. However, the algorithm that computes the shortest routes has been left out. The next paragraph elaborates on why the Improved Floyd-Warshall has been chosen to find the shortest routes.

Dijkstra's algorithm is one of the first that comes to mind when computing the shortest paths between two vertices in a static, weighted graph [26]. However, it has a time complexity of $\Theta(|V|^2)$, and when computing a large number of paths, it can be computationally advantageous to compute and store all shortest paths between any two vertices in a graph. The Floyd-Warshall algorithm, which has a time complexity of $\Theta(|V|^3)$, is a well-known algorithm used to compute all the shortest paths between any two vertices [15]. In terms of time complexity, the Floyd-Warshall algorithm is preferred when the number of paths to be computed exceeds the number of vertices. A large number of shortest paths between pickup and delivery vertices and the agent's locations must be computed in the MAPD problem. Hence, we compute all the paths at once and store them.

Let us now investigate the Floyd-Warshall algorithm. It is simple to use but extremely effective. It begins by constructing an adjacency matrix, the weights of which are initialized with the weights of the edges, and if no edge is available, the weight is set to infinity. Subsequently, it uses three for loops to iterate over the vertices and updates the adjacency matrix if a shorter path is found. Because of the three for loops its complexity is $\Theta(|V|^3)$. In large, sparse graphs the second and third for loop are mostly looping over ∞ values and therefore should be skipped (it is checking edges that do not exist). This is precisely what Torusla proposes in the Improved Floyd-Warschall algorithm (see Algorithm 4) [44]. In the artificial map, as well as the hospital map of Figure 2.5 a large number of vertices has been set up with few incoming and outgoing edges. Thus, the Improved Floyd-Warshall is particularly usefull on these maps. When analyzing the algorithm it can be observed that the use of the inlist and outlist for each vertex is where the faster computation is gained compared to the regular Floyd-Warshall algorithm. In addition, more for loops are used but the size of each for loop is much smaller if the graph is sparse.

Algorithm 4: Improved Floyd-Warshall Algorithm

```
1  Inputs: Vertices $N$, Adjacency matrix $A$
2  for $i \to N$ do
3      for $j \to N$ do
4          if $(i \neq j) \wedge (A[i,j] \neq \infty$ then
5              $out[i] \leftarrow out[i] \cup \{j\}$
6              $in[j] \leftarrow in[j] \cup \{i\}$
7  for $k \to N$ do
8      for each $i \in in[k]$ do
9          for each $j \in out[k]$ do
10             if $A[i,j] > A[i,k] + A[k,j]$ then
11                 if $A[i,j] = \infty$ then
12                     $out[i] \leftarrow out[i] \cup \{j\}$
13                     $in[j] \leftarrow in[j] \cup \{i\}$
14                 $A[i,j] \leftarrow A[i,k] + A[k,j]$
```