Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jon Elias Moen

Deep Reinforcement Learning for Supporting Ambulance Dispatch Decisions

Master's thesis in Computer science
Supervisor: Ole Jakob Mengshoel
June 2022

NTNU
Norwegian University of
Science and Technology

Jon Elias Moen

Deep Reinforcement Learning for Supporting Ambulance Dispatch Decisions

Master's thesis in Computer science
Supervisor: Ole Jakob Mengshoel
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

This master thesis explores the usage of Reinforcement learning (RL) and Proximal Policy Optimalisation (PPO) to the ambulance dispatching problem as a possible decision support tool for the Emergency Medical Service (EMS) in Oslo and Akershus municipality, Oslo University Hospital (OUH). The ambulance dispatching problem differs from a typical Vechicle Routing Problem (VRP) since patients (target locations) arrive stochastically (not known ahead of time), which makes the problem hard to solve. Furthermore, the EMS strives for optimal resource utilization, rapid response time, and good working conditions while at the same time experiencing an increase in demand. To solve this problem, a simulation model written in Python combined with Open Street Map (OSM) travel time estimation and simple synthetic incident data generation is implemented. Incident priority and incident queue are also put under consideration. Results show that the PPO model outperforms heuristic policies such as dispatching the closest ambulance by Haversine or Euclidean distance. Both when considering synthetic and real incident data. On the other hand, more work is needed for RL to be used as a decision support tool.

Sammendrag

Denne master avhandlingen utforsker bruken av forsterkningslæring og Proximal Policy Optimalisation (PPO) til ambulanseutsendelse-problemet, som et mulig beslutningsstøtteverktøy for Akutt medisin sentralen (AMK) i Oslo og Akershus, Oslo Universitets Sykehus (OUS).Ambulanseutsendelse-problemet er forskjellig fra et typisk ruteproblem (Vechicle Routing Problem (VRP)) siden pasienter (mål lokalasjoner) er stokastisk (ikke kjent på forhånd), som gjør at problemet er vanskelig å løse. I tilegg så strever AMK-sentralen med optimal ressursutnyttelse, kjapp responstid, og gode arbeidsforhold samtidig som at de opplever en økning i etterspørsel. For å løse dette problemet blir det det bygd en simuleringsmodell skrevet i Python, kombinert med Open Street Map (OSM)-reisetidsberegning og enkel syntetisk hendelses data generering. Hendelses-prioritet og kø blir også tatt i betraktning. Resultatene viser at PPO-modellen presterer bedre enn heuristiske retningslinjer, som å sende den nærmeste ambulansen basert på Haversine- eller euklidsk avstand. Dette gjelder både for syntetisk og virkelig hendelsesdata. Det kreves derimot mer arbeid for at RL kan brukes som et beslutningsstøtteverktøy.

Contents

Abstract . . . . **iii**

Sammendrag . . . . **v**

Contents . . . . **vii**

Figures . . . . **xi**

Acronyms . . . . **xv**

1 Introduction **1**

1.1 How the EMS system works 1

1.2 Problems with the current EMS system 4

1.2.1 Dispatching 4

1.2.2 Other 6

1.3 Scope and problem of solution 7

1.3.1 Similarities to other problems 7

1.3.2 Complexity 8

1.3.3 Scope 9

1.3.4 Contribution 10

1.4 Thesis Structure 11

1.5 Disclaimer 11

2 Literature Overview **13**

2.1 Allocation 14

2.2 Forecasting 14

2.2.1 Hermansen 15

2.2.2 Van De Weijer and Owren 15

2.2.3 Mannering 16

2.3 Reinforcement Learning 16

2.4 Reinforcement learning terminology 18

2.4.1 Actor Critic model 19
2.5 Proximal Policy Optimalisation (PPO) 20
2.6 Muti agent reinforcement learning 21
2.7 Dispatching 21
2.7.1 Algorithmic Dispatching 22
2.7.2 Reinforcement Learning Dispatching 22
2.8 Discussion 28
2.8.1 Reinforcement learning 28
2.8.2 Forecasting 30
2.8.3 Allocation 31
3 Tools and Datasets 33
3.1 Datasets 33
3.1.1 Road network dataset 33
3.1.2 Incident dataset 34
3.2 Tools 34
3.3 Dataset Preprocessing 34
3.3.1 Incident dataset 35
3.3.2 Open Street Map (OSM) dataset 35
3.3.3 OpenRouteService 38
4 Data Analysis 41
4.1 Incident data analysis 41
4.1.1 Reponse time 41
4.1.2 Incidents per location 43
4.1.3 Incidents over time 45
4.2 Open Street Map (OSM) dataset 47
4.3 Open route service 48
5 Method 51
5.1 Simulator model 51
5.1.1 Synthetic Incident generation 52
5.1.2 Ambulance simulation 53
5.1.3 Incident Queue 56
5.1.4 Small world setup 57

5.1.5 RL setup 57
5.2 Reinforcement learning model 60
5.2.1 Hyperparameter search 60
5.2.2 Environment 61
5.2.3 Evaluation 61
6 Experiments and results 63
6.1 Experiment 1 64
6.1.1 Results and discussion 64
6.2 Experiment 1.2 65
6.2.1 Results and discussion 66
6.3 Experiment 2 67
6.3.1 Results and discussion 69
7 Conclusion and Future work 71
7.1 Conclusion 71
7.2 Future work 72
7.2.1 Reinforcement learning (RL) 72
7.2.2 Simulation 74
7.2.3 Data analysis 75
7.2.4 Other optimization techniques 75
7.2.5 Limitations 75
Bibliography 77
A Additional Material 83
A.1 Make intersection graph (textual) 83
A.2 Make grid-cell graph (textual) 83
A.3 Makeshift survival function estimation 84
A.4 Breakdown of updating ambulances 84
A.5 Incident queue (textual) 85
A.6 All externally created links 88

Figures

1.1 Shows a simplified flow chart of how the EMS system works. This flow chart assumes all incidents are transported to the hospital. t-d, i-d, and h-l is time not used on driving (processing time), which is discussed further in chapter 4. 3

1.2 Scatter plot of euclidean versus shortest road network distance. The data is gathered from the Netherlands (Image source [5]) 5

1.3 Shows difference between Euclidean distance and Road network distance. Notice how the Euclidean distance is lower than the Road network distance. (Image source [8]) 5

1.4 Shows a particular case where an ambulance traveling to an urgent incident (Red arrow) could be redirected to an acute incident during travel. Since $d_2 < d_3$ and acute (A) has higher priority than urgent (H). The X and Y axis illustrates the spatial dimension, while the value of d shows the distance. 6

1.5 Confusion matrix of labeled priority and actual urgency. Compiled using data from Samdal *et al.* [10] 7

2.1 Simple overview of the research discussed in this chapter 13

2.2 Comparison of survival functions for cardiac arrest (Image source [27]) 15

2.3 The setup of reinforcement learning (Image source [33]) 17

2.4 Shows the actor critic setup. Policy is the actor, while the value function is the critic. (Image source [34]) 19

2.5 State action tree utilised in Mukhopadhyay *et al.* 27

3.1 Shows a simple example of the construction of graph H. The example only shows nine grid cells but is done for all grid cells. Each cell has a marked centroid and edges to other cells if a road connects them (Background image source [49]). 37

3.2 Shows the resulting road network graph of Oslo and Akershus. . . . 37

3.3 Shows an example of one isochrone from the base station in Bjørkelangen. 39

4.1 (Plot left) Average Responstime per priority, with marked 95% confidence interval. (Plot right) histogram plot of responstimes per incident. 42

4.2 Shows the inverse CDF of the response time for each priority. In other words, the resulting makeshift survival function 43

4.3 Shows a histograms of the different transition times with acute and urgent priority. (Plot Left) Time before starts driving to incident after a call is registered. (Plot Middle) Time before starts driving to hospital after arriving at the incident location. (Plot right) Time before ambulance is available after arriving at the hospital 44

4.5 Map of deviation from population linear regression. Both a zoomed in and full perspective. Full map available online https://joneliasmoen.github.io/popLinreg.html

4.6 Increase in volume of incidents weekly. Compared to interpolated increase in inhabitants 46

4.9 Hisogram of road network graph H degree. 49

4.10 Road network graph degree. Degree referes to the number of grids a certain cell can travel to by car 49

4.11 All isochrones from base stations in Oslo and Akerhus 50

4.12 Isochrones in City center Oslo, Norway. 50

5.2 Shows the main flow of the nonQueue version of the simulator . . . 57

5.3 Shows how the queue version of the simulation model handles unavailable ambulances. 58

5.4 This shows how the RL model interacts with the simulator, together with its input/output. Lastly elements in green shows the components the RL and simulator comprises of. 59

6.1 Shows considered small world (5km radius from Oslo Central station). The big black circle show the area considered, while the small black squares the grid-cells. Base stations are marked with blue circles, while hospitals in red. 63

6.2 Shows results for the different policies evaluated (**ex 1**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time and waittime 66

6.3 Shows results for the different policies evaluated (**ex 1.2**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time and waittime 68
6.4 Shows results for the different policies evaluated (**ex 2**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time. Inverse CDF is shown for survivability 70
6.5 Shows the reduction in mean response time (Y-axis) during training of the RL model in **experiment 2**. X-axis shows the total number of incidents trained on. (Screenshot from TensorBoard). 70

A.1 Shows visually how incidents during simulation while waiting for ambulance. Upwards arrows are negative, while downwards positive. Numbered according to step-wise list. 88

Acronyms

ACO Ant Colony Optimization. 75

ADP Approximate Dynamic programming. 22, 28, 29

API Application programming interface. 10, 38

CAD Computer aided dispatch. 1, 4, 7, 28, 52

CDF Cumulative distribution function. xii, xiii, 6, 42, 43, 52, 64–70, 84

DES Discrete event time simulation. 10, 51

DP Dynamic programming. 10, 18

DQN Deep Q-network. 28, 29

DVRP Dynamic Vehicle Routing Problem. 8, 75

EMCC Emergency Medical Communication Centre. 1

EMS Emergency Medical Service. iii, xi, 1–4, 6–9, 11, 13–15, 21–25, 34, 43, 45–47, 71, 72, 74, 75

GA Genetic Algorithm. 14, 16

GLM Generalized Linear Model. 75

LSTM Long short-term memory. 15, 26, 74

MAQR Multi-agent Q-network with Experience Replay. 10, 25, 28, 29

MC Monte Carlo. 18

MCTS Monte Carlo tree search. 27–29, 72

MDP Markov decision process. 10, 22–29

MDVRP Mutli Depot Vehicle Routing Problem. 7, 8, 75

MIP Mixed integer programming. 14

MLP Multi-layer perceptron. 15, 16, 60, 64, 68, 72

MTBF Mean time between failiure. 46, 52

OSM Open Street Map. iii, v, viii, 10, 33–36, 47, 51, 71

OUH Oslo University Hospital. iii, 1, 14, 42, 44

PBF Protocolbuffer Binary Format. 33, 37

PMF Probability mass function. 52

PPO Proximal Policy Optimalisation. iii, v, viii, 7, 9, 10, 18, 20, 29, 60, 61, 63–65, 71, 73, 74

RL Reinforcement learning. iii, v, ix, xiii, 7, 11, 13, 16–19, 21, 22, 28–31, 34, 51–53, 57, 60, 61, 63, 67, 69–75

SGD Stochastic gradient descent. 27

SMDP Semi Markov decision process. 26, 28, 29

SSB Statistisk Sentral Byrå. 34

TD Temporal differencing. 18, 19, 28

TSP Travelling Salesman Problem. 7, 8

VAE Variational autoencoder. 75

VRP Vechicle Routing Problem. iii, v, 7

VRPPD Vehicle Routing Problem with Pick-up and Delivering. 8

WGAN Wasserstein generative adversarial neural network. 16, 75

Chapter 1

Introduction

The Emergency Medical Service (EMS) in Oslo and Akershus municipality is handled by Emergency Medical Communication Centre (EMCC) at Oslo University Hospital (OUH). While there is high demand for fast response times, the resources they have at their disposal are limited. Because of this reason, optimal resource utilization is strived for.

In cardiac and circulatory arrest incidents, chances for resuscitation drop fast over time. This chance has an estimated drop of 10% every minute, making fast response crucial [1]. Even though Oslo University Hospital (OUH) has stated that the percentage of incidents with cardiac arrest is low, a fast response is vital, especially for acute incidents, see section 2.1. The EMCC also suffer from high turnover rates due to bad working conditions. At times ambulance drivers can work for long hours without breaks and socializing. Therefore resource management also needs to consider the human aspect of the task.

On top of this, OUH experience an increase in incidents over time. Since this is a multi-dimensional problem counteracting this increase is complex. However, the growth is primarily due to a population increase over time.

Additionally, the Computer aided dispatch (CAD) software which the EMS utilize for decision support can be improved. Especially in ambulance dispatch recommendations, as current CAD software has room for improvement.

Much of this information builds upon the work of Schjølberg and Bekkevold [2], who have interviewed ambulance personnel associated with Oslo University Hospital (OUH).

1.1 How the EMS system works

Below is a step-wise list of how the ems system works. Figure 1.1 also illustrates this process. The Triage hierarchy mentioned in the list has 3 different levels. The codes in this Triage are Acute (A), Urgent (H) and planned/unplanned regulars (V). These correspond to the **priority** of the incident, i.e how fast the ambulance should respond to that incident.

1. An incident occurs, and someone calls the EMS
2. A dispatcher at the EMS assesses the priority of the incident on a Triage hierarchy scale.
3. An ambulance close to the incident is dispatched to the location. The haste is defined by the given priority.
4. The ambulance arrives at the location and either treats the patient at the spot, or picks up the patient.

    4.1 If not treated on the spot, the ambulance delivers the patient to the closest hospital.

5. After treating the patient, the ambulance drives to its assigned base station.

Below is a list of definitions from the step-wise list given above.

- **Response time** is defined as the time from when EMS receives a call until an ambulance arrives at the given location. That means the time from step 1 all the way to step 4 ($t_r = t_5 - t1$).
- The **Ambulance dispatching** problem is the task of deciding which ambulance to send to which incident. Which corresponds to step 3 in the list.
- An **allocation** is defined by the number of ambulances assigned to each base station. An **allocation** can either be rigid or dynamic throughout time.
- If no ambulance is available, incidents enter a FIFO **incident queue** and awaits an ambulance. Once an ambulance is available, ambulances are dispatched to incidents from this queue.
- The **coverage** of an ambulance is defined as the spatial area said ambulance can travel to within a certain time limit.
- **Forecasting** is the task of predicting incident volume and location. This prediction occurs in both a space and time dimension (Spatio-temporal). Since incidents happen at a certain time at a certain location.

There are two different types of base stations: ambulance stations and standby points. Ambulance stations are well-established garages for ambulances with proper facilities. In comparison, standby points are temporary garages with less space for ambulances and humans. These standby points also have less established facilities, such as toilets and break rooms.

Various types of ambulances are utilized based on the nature of incidents and the prevailing road traffic conditions. However, in this work, only regular ambulances are assumed.

It is important to remember that special traffic laws apply to ambulances. Ambulance drivers are obligated to drive responsibly but are excluded from speed limits when it is justifiable. Other cars must yield to an ambulance with blue lights,

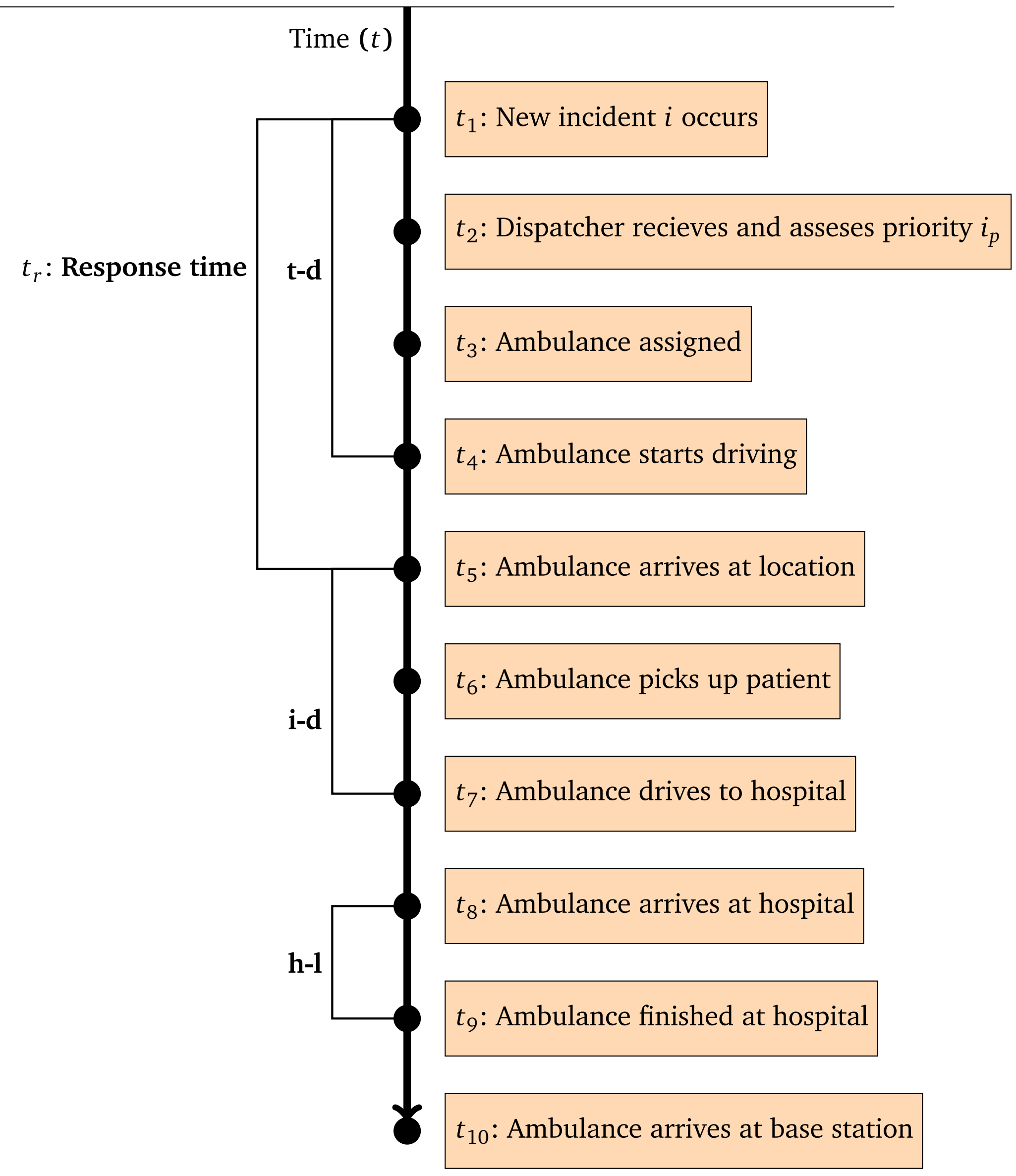
Time (t)
t_1: New incident i occurs
t_2: Dispatcher recieves and asseses priority i_p
t_3: Ambulance assigned
t_4: Ambulance starts driving
t_5: Ambulance arrives at location
t_6: Ambulance picks up patient
t_7: Ambulance drives to hospital
t_8: Ambulance arrives at hospital
t_9: Ambulance finished at hospital
t_{10}: Ambulance arrives at base station
t_r: **Response time**
t-d
i-d
h-l

Figure 1.1: Shows a simplified flow chart of how the EMS system works. This flow chart assumes all incidents are transported to the hospital. t-d, i-d, and h-l is time not used on driving (processing time), which is discussed further in chapter 4.

meaning ambulances can travel faster than a typical car. The travel time of the ambulances also depends on the incident priority, making travel time prediction nontrivial [3].

Digital solutions are incorporated to aid the ambulance dispatch process. Typically computers are both located at the dispatch center and in the ambulances. CAD software is to enter the location of the incident, assessed priority, and the choice of an ambulance to dispatch [4].

1.2 Problems with the current EMS system

1.2.1 Dispatching

The current methodology for ambulance dispatching has a very static approach to a stochastic environment. Many of the methods used today are rigid and follow a strict protocol.

Once an incident occurs, the decision on which ambulance to send (dispatching) is based on the Euclidean distance between the incident and currently available ambulances [1]. This protocol does not consider the city's road network or traffic congestion. An ambulance with a high Euclidean distance can be closer than an ambulance with a low Euclidean distance due to the structure of the road network. As shown on Figure 1.3, low Euclidean distance can result in high road network distance. Even though Euclidean is strongly correlated with road network distance, there are outliers [5]. A scatter-plot of Euclidean distance towards road network distance is shown in Figure 1.2

Typically incidents in the incident queue are served in a FIFO manner. The incident that has waited the longest in the queue is served first. When considering incident priority, this might not be the best approach since there might be an incident with higher priority further back in the queue. This results in a fast response time to the incident with low priority and a high response time to the incident with high priority.

Due to regulations, the EMS are restricted to always dispatch some ambulance once an incident occurs. It is, however, possible that an unavailable but soon-to-be available ambulance could have a shorter response time than the currently closest available ambulance [6].

Another aspect is that assigning ambulances to incidents can lead to ambulances getting drawn away from locations where the ambulance should have been positioned. In other words, always sending the closest ambulance pulls ambulances away from areas with a high likelihood of an incident occurring. This results in those locations having a longer response time [7].

One complex aspect which has similarities to a paradox also can happen. In this situation, an ambulance is dispatched to a low-priority incident and drives to-

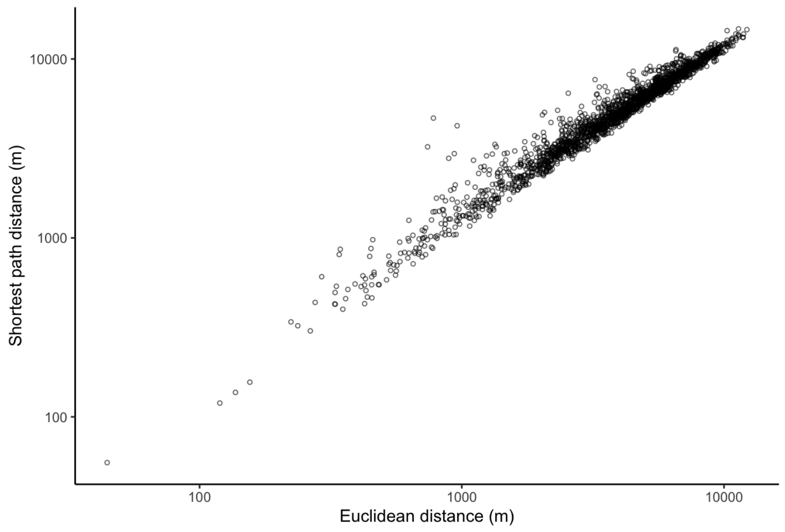

Figure 1.2: Scatter plot of euclidean versus shortest road network distance. The data is gathered from the Netherlands (Image source [5])

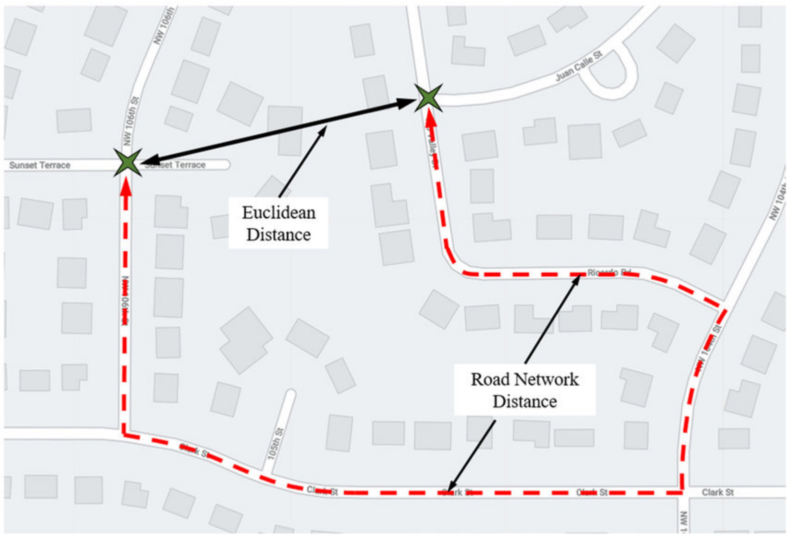

Figure 1.3: Shows difference between Euclidean distance and Road network distance. Notice how the Euclidean distance is lower than the Road network distance. (Image source [8])

wards it. While driving, another incident with higher priority occurs close to the ambulance but not close to a base station. Figure 1.4 illustrates this situation, where d is some distance to each incident. Now the question is, does the ambulance reroute to the higher priority, or should an ambulance from the base station be dispatched to the new incident? This problem involves carefully prioritizing the incidents, their probability of resuscitation, and distance. It also depends on

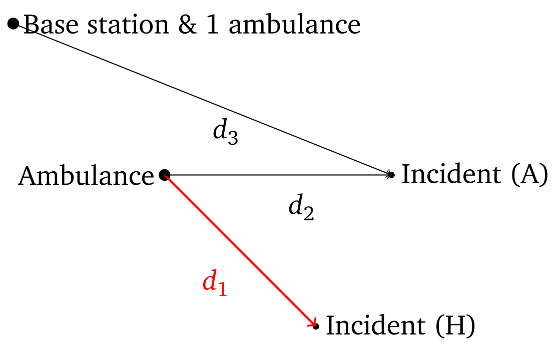

Figure 1.4: Shows a particular case where an ambulance traveling to an urgent incident (Red arrow) could be redirected to an acute incident during travel. Since $d_2 < d_3$ and acute (A) has higher priority than urgent (H). The X and Y axis illustrates the spatial dimension, while the value of d shows the distance.

the value of d_1, which defines how close the en-route ambulance is to its assigned incident.

The Norwegian Directorate of Health has set goals on how fast the response time should be. These goals are quantifiable and depend on the incident's priority and the population density. In literature, this is called Coverage and revolves around the CDF of the response time. These goals are listed below [2] [9].

1. In **densely** populated areas 90% of acute incidents should have a response time lower than **12 minutes**
2. In **sparsely** populated areas 90% of acute incidents should have a response time lower than **25 minutes**.

Unfortunately, the EMS in Oslo and Akershus do not meet these goals.

1.2.2 Other

Often incidents are labeled as acute due to the uncertainty of the call (step 1) since the caller provides little or uncertain information. Assessing an incident as acute eliminates the risk of arriving later if labeled as urgent, resulting in over-labeling incidents as acute. For example, in 2015, 74-80% of incidents were over-labeled, and 20-30% were under-labeled in the southeast part of Norway [10]. Figure 1.5 shows a confusion matrix of the assessed and actual urgency, which is also adjusted for availability, response time, and transport time.

Often incidents are labeled as acute due to the uncertainty of the call (step 1) since the caller provides little or uncertain information. Assessing an incident as acute eliminates the risk of arriving later if labeled as urgent, resulting in over-labeling incidents as acute. For example, in 2015, 74-80% of incidents were over-labeled, and 20-30% were under-labeled in the southeast part of Norway [10].

Figure 1.5 shows a confusion matrix of the assessed and actual urgency, which is also adjusted for availability, response time, and transport time. However, the data quality of the estimates in the figure is low. Since the existing CAD system data is inconsistent and inaccurate for scientific research, which is a common problem among other studies assessing the accuracy of medical dispatch [11]. There also exists no consensus on standards to asses the accuracy.

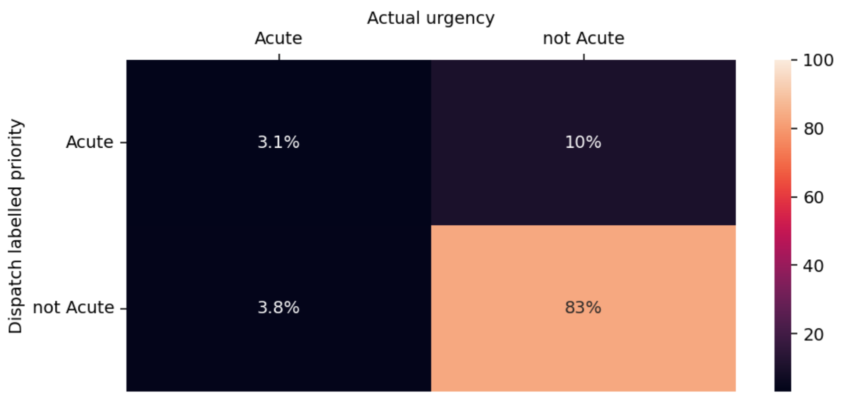

Figure 1.5: Confusion matrix of labeled priority and actual urgency. Compiled using data from Samdal *et al.* [10]

1.3 Scope and problem of solution

The main focus is on ambulance dispatching, in other words, optimizing the choice of which ambulance to dispatch to a given incident. The aim is to improve the response times according to the quantifiable goals of the Norwegian Directorate of Health while also accounting for incident priority and queue. This work will use Reinforcement learning (RL) and the PPO algorithm to achieve this goal. This system will then provide a recommended dispatch action as a decision support tool for the EMS.

1.3.1 Similarities to other problems

The ambulance dispatching problem is closely related to the Travelling Salesman Problem (TSP), Vechicle Routing Problem (VRP) and Mutli Depot Vehicle Routing Problem (MDVRP). While researchers consider these problems tractable when employing heuristics, it is crucial to note that particular significant distinctions exist.

To begin with, in TSP and MDVRP, all the target locations (cities) are known ahead of time. In an EMS setting, the target locations (incidents) are unknown ahead of time, and ambulances are dispatched immediately on occurrence. Consequently, this results in a tiny time window for planning.

Further, TSP and MDVRP consider that a vehicle can visit multiple locations before returning to their depots. In an Emergency Medical Service setting, multiple ambulances can be dispatched to a single incident (location) before potentially proceeding to the hospital and ultimately to a base station (depot). It can be the case that an ambulance can visit multiple incidents (locations); however, the maximum carrying capacity of each ambulance is unknown. Furthermore, this relies on many unknown factors of the actual urgency of the incident.

The ambulance dispatching problem can be seen as a combination of MDVRP, Vehicle Routing Problem with Pick-up and Delivering (VRPPD) and Dynamic Vehicle Routing Problem (DVRP). Since locations (incidents) can occur during the execution of a route (DVRP), and patients are sometimes delivered to the closest hospital before returning to their depot (VRPPD). When accounting for the capacity of the ambulances, each ambulance type, ambulance relocation, incident priority, incident queue and traffic congestion the ambulance dispatching problem differs from these problems. Making the ambulance dispatching problem hard to solve with heuristics.

Further, ambulance dispatching is similar to ride-hailing/sharing problems. In these problems, taxis or similar vehicles serve customers with stochastic origin and destination. The goal for these vehicles is optimal Coverage together with fast response times. However, in these problems, taxis drive around proactively anticipating customers and have no depots.

Lastly the ambulance dispatching problem has close relation to the ambulance allocation problem [2] and incident forecasting [12][13][14].

1.3.2 Complexity

Solving the ambulance dispatching problem is not an easy task. As further examined in chapter 2, many aspects of improving the EMS pipeline struggle due to the high dimensionality of the problem and its temporal instability. Below is a list of reasons solutions to these problems struggle.

1. Incidents occur distributed in space (spatial), time (temporal), and priority, which makes accurate forecasting hard to solve. Some success has been shown in predicting the total number of incidents. Still, these methods struggle with underestimating predictions, lack of causal features (such as population shifts, traffic congestion, and road construction), over/under dispersion of the data, feature selection (Such as appropriate contextual information), data quality and size, interpretability of data-driven approaches, and temporal instability [14] [4]. When considering the spatial and priority aspect, prediction methods struggle. Due to different spatial patterns for each pri-

ority and spatial granularity [15] [12].

2. Many papers have shown that sending the closest ambulance to every occurring incident is myopic optimal [16] [17][1][6][18]. In other words, there is room for improvement in the current methodology for ambulance dispatching.
3. Due to the dynamic nature of the spatial pattern of EMS incidents, the ambulance dispatching policy also exhibits strong dynamism. Which simple rule-based approaches often fail to capture [17].
4. As mentioned in section 1.1, realistic ambulance travel time prediction is nontrivial.

In the case of ambulance dispatching, solutions have been found using a variety of techniques. Below is a list of problems in previous solutions.

1. The priority of an incident is often ignored [19] [20]. Accounting for the priority of the incidents has been shown to increase survivability [21].
2. Standard decision-theoretic models have been evaluated for the problem but are intractable due to the large dimensionality [4].
3. Some papers assume that response time is independent of priority. [21][16]
4. Changing traffic conditions is often ignored. Also, the different types of EMS vehicles for road conditions in different scenarios are not accounted for [20].

1.3.3 Scope

This thesis focuses on implementing the PPO algorithm to the ambulance dispatching problem while considering incident priority, queue, and a changing number of ambulances. Furthermore, an implementation of a simulator model in Python and a framework for generating synthetic data is also provided. . An emphasis is also made on Data analysis (4) to provide insight into the datasets provided.

In this implementation, some simplifying assumptions are considered. They are similar to the assumptions made by Schjølberg and Bekkevold [2].

1. Ambulances are assumed to always travel to the closest hospital after attending an incident. In a real-world scenario, an ambulance can treat an incident on the spot, and a given hospital might also be understaffed or unsuitable for the given patient. However, this consideration is overlooked due to insufficient information.
2. Ambulances are considered reactive, meaning ambulances only move in reaction to incidents. This differs from a proactive perspective, where ambulances patrol around to cover vital areas.
3. One ambulance is assumed to have a capacity for one incident. Furthermore, one ambulance is assumed to be needed for every incident.
4. Only Acute (A) and Urgent (H) incidents are considered ($i_p \in \{A, H\}$). Ig-

noring Planned Regulars (V).
5. Dispatch actions are only performed when incidents occur or there are incidents in the incident queue.
6. Due to lack of information, travel times are assumed to be independent of incident priority.
7. The uncertainty in the labeling of the incidents is not considered; Since there are no clear indications of the real urgency before an ambulance arrives. Van Barneveld *et al.* [22] argues that dispatchers can't know if an incident needs hospitalization before an ambulance arrives.

1.3.4 Contribution

Previous	**This**
Usage of Markov decision process (MDP), Dynamic programming (DP), and Multi-agent Q-network with Experience Replay (MAQR)	Implementation with Proximal Policy Optimalisation (PPO)
Commonly ignoring incident priority	Accounting for incident priority
Usage of costly travel time API.	OSM based travel time. Reproducible
Some dataanalysis	Extensive dataanalysis
FIFO incident queue.	Choice of any incident in queue
Discrete event time simulation (DES) simulation	Skips forward until next incident.
	Day / night time allocation consideration

Table 1.1: Shows main contributions compared to previous implementations

Table 1.1 shows the main contributions compared to previous implementations. From this the usage of PPO, case study of Oslo and Akershus, and incident priority is most vital. A substantial effort was made on the simulation model and data analysis. Details on each of these contributions are shown throughout the thesis.

The implementation of PPO to the problem of ambulance dispatching has not been done before. Bélanger *et al.* [23] argues there is a need for experimentation with new models. Holler *et al.* [24] applied the same algorithm to a similar problem and achieved promising results.

The literature some of the literature considers both ambulance relocation and dispatching problems. However, there is little consideration of changing the number of available ambulances during day and night shifts.

It is common to ignore incident priority to reduce the complexity of the problem. However, this implementation considers incident priority since it affects the dispatching order of the ambulances.

Considering the possibility of overfitting historical data, a synthetic incident generator is built. Testing the model on both historical and synthetic data ensures proper validation.

1.4 Thesis Structure

This thesis is divided into the following chapters: Introduction, Literature Overview, Tools and Datasets, Data Analysis, Method, Experiment and Results, and Conclusion. The chapters follow the order of implementation, meaning each chapter relies on the information given in previous chapters. A breakdown of each chapter is given below.

chapter 2 provides a literature and background overview. First, the literature on other aspects of the EMS pipeline using the same dataset is outlined. Then the background for the implementation is outlined together with previous solutions to the ambulance dispatching problem using RL. Lastly, a discussion of previous implementations is given.

chapter 3 provides an overview of the tools and datasets utilized for the data analysis (chapter 4) as well as the implementation of the RL and simulation model (chapter 5). This chapter also includes the data pre-processing methodology.

chapter 4 provides a data analysis into the datasets to provide insight and extract knowledge for synthetic incident data generation. This data analysis aims to provide additional insights beyond previous implementations using the same dataset.

chapter 5 provides an overview of the methodology for the simulator, the RL model, and its evaluation.

chapter 6 explains the setup of each experiment, its result, and discussion. The methodology of these experiments is primarily covered in the previous chapter.

chapter 7 wraps up the thesis with a conclusion of the result and future work.

1.5 Disclaimer

This thesis continues the pre-project delivered by the same author in autumn 2022. This is because the insights gained from the pre-project are highly relevant to this thesis. Especially the literature overview (chapter 2) and data analysis (chapter 4) chapters are similar to the pre-project.

Chapter 2

Literature Overview

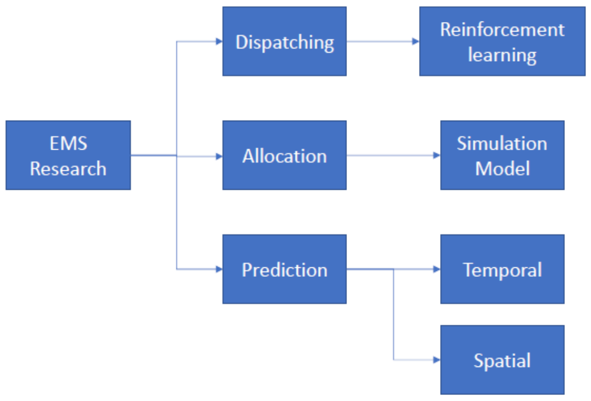

Figure 2.1: Simple overview of the research discussed in this chapter

The main streams of EMS research can be divided into allocation (section 2.1), forecasting (section 2.2), and dispatching (section 2.7). A simple overview is provided at the end of the chapter (Figure 2.1). Bélanger *et al.* [23], Neira *et al.* [25] and Mukhopadhyay *et al.* [4] are the most recent review papers on the ambulance dispatching problem and EMS research. This chapter also covers the background for RL.

In machine learning, there is a difference between online and batch learning. When batch learning is performed on a model, it is trained on a given dataset in batches and later deployed. After deployment, the model does not improve its predictions based on data gathered during deployment. Moreover, it does not retrain the model after deployment. This contrasts with online learning, where the model is continuously trained during deployment based on prior predictions.

2.1 Allocation

Schjølberg and Bekkevold [2] utilised Genetic Algorithm (GA) and other evolutionary methods to find optimal ambulance allocation. Previous methods utilized Mixed integer programming (MIP), which has higher time complexity than evolutionary methods. The allocations are evaluated in a Discrete event simulation model built in Java. This simulation model utilized the work of McCormack and Coates [26].

The allocations to be optimized were static and independent of the time dimension. Both a night and a daytime allocation were sought.

Clustering techniques were to find allocations proportionate to the population around the base stations (population-proportionate). Results showed that this technique performed better over a long-term simulation of one year. However, whether this is an artifact of overfitting by the GA method or if the population-proportionate is always optimal is discussed. From a short-term perspective (less than three months), the optimized allocations performed better than the other methods. This indicates that an optimized ambulance allocation will work well in a short time perspective but needs to be updated to work well continuously.

The number of ambulances used for simulation and optimization was experimented with. Results showed that the number of ambulances in Oslo and Akershus could lower without a severe increase in average response time. However, whether this is true is discussed due to the non-linear relationship between response time and chance of survival (survivability). The authors argue that the allocations would be more optimal if a survival function were given by OUH.

As a side note, a survival function models the probability of patient survival as a function of response time. These functions are investigated in literature for cardiac arrest, where a fast response time is vital. Examples of such survival functions are shown in Figure 2.2. The shape of a survival function depends on many factors, such as specific types of incidents, age, health, and the availability of medical care in ambulances.

2.2 Forecasting

The forecasting task is spatio-temporal, meaning the problem has two dimensions: location (spatial) and time (temporal). Having such a large dimensionality makes predictions hard, especially with high granularity. There are different definitions for high/low granularity for both the spatial and temporal aspects. But in the EMS setting, the literature defines one hour as high temporal granularity.

The previous master theses of Hermansen [13] and Van De Weijer and Owren [12] have focused on prediction using the same dataset provided here. Both build upon the MEDIC method and the work by Setzler *et al.* [28] and Zhou [29]. The

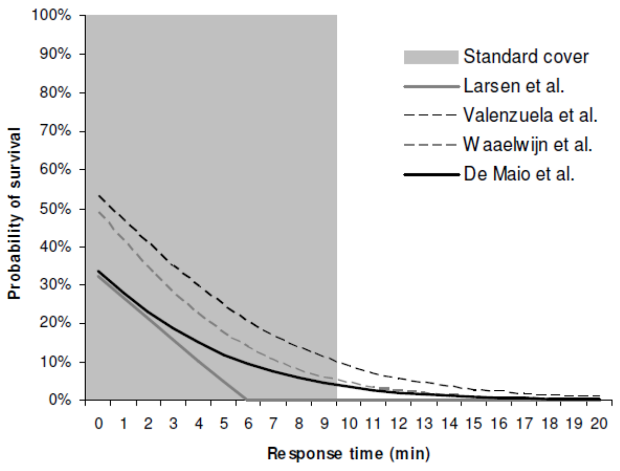

Figure 2.2: Comparison of survival functions for cardiac arrest (Image source [27])

MEDIC method is used by the EMS department at Charlotte-Mecklenburg, North Carolina. It forecasts by averaging over historical data.

2.2.1 Hermansen

Hermansen [13] [14] studies a complete and a split approach to the problem. The complete approach directly forecasts each location, while the split approach has a separate model for the total volume and the spatial distribution. Both Multi-layer perceptron (MLP) and Long short-term memory (LSTM) models were investigated. Online learning is utilized instead of batch learning for all of the models.

When comparing the evaluations of the split and complete approach on the spatial distribution and volume separately, the split approach performs better. The complete approach performs marginally better when comparing the evaluations on the spatial distribution and the volume combined. Overall the split approach produces more helpful information than the complete approach.

Weather also has an impact on call volume [30]. Nevertheless, results showed the models without weather data performed better on volume evaluation—conversely, the best model for the combined evaluation used weather data. It might be possible that the weather variability is captured in the time-dependent features of models not using weather data.

2.2.2 Van De Weijer and Owren

Van De Weijer and Owren utilized different time series decomposition techniques in combination with MLP to predict the total volume of incidents over time. The same split approach is used as Hermansen [13]. Finally, another model was used, which used the GA to optimize the weights of a Poisson neural network.

For spatial prediction, GA is used to aggregate the spatial locations in the dataset to optimize the performance of an MLP. In addition, an MNIST pre-trained Wasserstein generative adversarial neural network (WGAN) was also utilized for spatial prediction.

The volume forecast performed well. However, the spatial prediction models struggled. The GA aggregated MLP outperformed the historical average by a thin margin. While the WGAN struggle due to the sparsity of the spatial distribution. The spatial predictions improved by the inclusion of covariates and by spatial aggregation; this, however, comes at the cost of precision. The thesis also found that the number of people most frequently in need of ambulances is roughly the same per location throughout time.

2.2.3 Mannering

For the specific case of highway accidents Mannering [31] argue over the temporal instability of model parameters due to instability of driver behavior. Since driver behavior is largely temporally unstable, the model parameters forecasting incidents are also temporally unstable. In other words, the model parameters cannot be constantly optimal over time which can be solved by online learning.

Driver behavior is affected by a lot of individual and social factors. These factors include driver age, risk-taking behavior, and macroeconomic conditions. Macroeconomic conditions have been shown to be a strong predictor of risk-taking behavior, which again influences driver behavior.

Since driver behavior is temporally unstable, the changing dynamics of urban environments (such as traffic, population shifts, and road construction) highlight online learning's importance.

2.3 Reinforcement Learning

Reinforcement learning (RL) is a paradigm in machine learning which differs from unsupervised and supervised learning. It models intelligent agents interacting with an environment to maximize its reward [32].

In this environment, states, possible actions, and rewards are outlined. Given a state of the environment s it transitions into a state s' after performing action a and receives reward r. The goal is to perform actions through time t to maximize an episode's discounted reward. An episode is a series of states and actions in an environment instance. An episode can terminate when a terminal (goal) state is achieved, or a maximum number of iterations limit is reached. When the environment is non-deterministic, the state transitions from s to s' with action a with

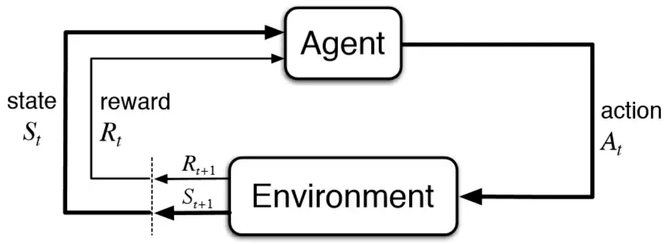

Figure 2.3: The setup of reinforcement learning (Image source [33])

a probability $P(s \rightarrow s'|a)$ or often denoted $P(s'|s,a)$.

Most of RL theory assumes environments satisfy the Markov property. The Markov assumption holds in a stochastic system that possesses this property. The Markov assumption assumes that future states depends only on the current state, and not on several past states. In other words, it has no memory of past states. i.e the state transition probabilities are only dependent on the given state.

The action a is modeled through a policy π learned through time. Policies utilize a state-value function $V(s)$, which models the desirability for an intelligent agent to be in state s (following policy π). The state-value function is the expected future rewards when actions are performed from that state. These future rewards are discounted, meaning future rewards affect the current state value with a coefficient γ, which reduces over future time t with γ^t. With an infinite time horizon, the state-value function is defined in equation 2.1.

$$V_\pi(s_{t=0}) = \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t \cdot r_t] \tag{2.1}$$

A policy can also be modeled through an action-value function (Often called Q-values, utilized in Q-learning). Which essentially is the same as the state-value function, given that action a is performed in state s. The action-value function is defined in the Equation 2.2.

$$Q_\pi(s,a) = \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t \cdot r_t | a_s = a] \tag{2.2}$$

With perfect estimations, the optimal action is the action which maximises the state-value function or the action-value function. i.e $a(s) = \max_a Q_\pi(s,a)$

The policy π, can be represented in different ways. In environments with small state space, the representation can be a dictionary mapping the state and action to a value. It can also be represented as a deep neural network for environments with ample state space. This would then take a state as input and return an expected future reward distribution among possible actions (Q-values).

The estimation of the policy varies depending on the type of RL. Monte Carlo (MC) methods and Temporal differencing (TD) are experience-based while Dynamic programming (DP) is model-based. Experience-based methods update values and probabilities from experience throughout actions in episodes. While model-based methods assume transition probabilities $P(s \to s'|a)$ and rewards r for all states are known.

Most of RL theory revolves around DP and the Bellman equation. The Bellman equation is a constraint that should hold for each state in the environment and is the basis for policy updates in DP. Equation 2.3 shows the Bellmann equation for a state-value policy, which requires a given model for transition $P(s \to s'|a)$ and $R_{s'}$. Let $\pi(s,a)$ be the probability of choosing action a in state s following policy π.

$$V_\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s \to s'|a)[R_{s'} + \gamma V_\pi(s')] \tag{2.3}$$

The goal in RL is to find a policy $\pi(s)$ such that all Bellman constraints for all states hold. Under such optimal conditions, the optimal policy is defined under the Bellman optimality equation $V_*(s) = \max_a \sum_{s'} P(s \to s'|a)[R_{s'} + \gamma V_*(s')]$. In other words, doing action a, which satisfies this optimality equation, is always the most optimal. Given a perfect model as explained above, state values can be propagated backward from $\forall_{s'} V(s')$ to $V(s)$, thus iteratively solving the optimal state-value function and policy. This process is called value iteration, and the policy is the action that satisfies the optimality equation.

If a perfect model is not given, generalized policy iteration (GPI) can be applied, which defines a more general way of learning an optimal policy under unknown circumstances. It switches between policy evaluation (PE) and policy improvement (PI).

2.4 Reinforcement learning terminology

The distinction between offline and online RL methods needs to be made clear. The main difference is that in online methods (such as PPO) the replay buffer is completely disregarded after it has been trained on. This differs from offline setups, where the replay buffer is continuously grown and trained on.

Furthermore, there is a distinction between on-policy and off-policy methods. While off-policy methods (such as Q-learning) has one behavior policy and one target policy, on-policy methods only have one policy. Off-policy methods use a separate policy to explore the environment (behavior policy) from the trained policy (target policy). On-policy methods explore the environment with the same policy which is taught.

A general problem in RL is stability. An epoch of training might impose undesirable large updates to the policy. Which makes the policy get stuck in local optima.

2.4.1 Actor Critic model

The actor-critic model is a policy gradient method composed of two models, extending the TD and RL theory. The actor represents the policy π while the critic the state value function $V(s)$. The critic evaluates how well the actor performs in the environment through TD-error δ (Also denoted A for advantage in literature). These two models are trained concurrently, so the critic becomes better at evaluating the actor while the actor's policy improves over time. Figure 2.4 shows the setup of the actor-critic model [32].

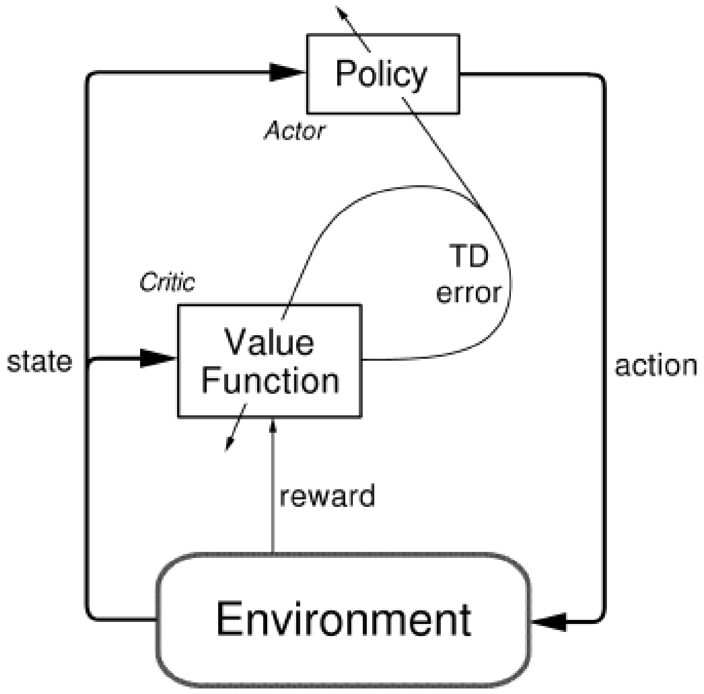

Figure 2.4: Shows the actor critic setup. Policy is the actor, while the value function is the critic. (Image source [34])

Assume that the actor (policy π) just performed action a in state s of the environment, received reward r, and transitioned into s'. The critic evaluated the states with $V(s)$ and $V(s')$. The actor is a Q-network meaning it produces an action-value distribution $\pi(s)$ for state s, denoted $\pi(s,a)$ for a specific state action value.

The TD-error critiques $V(s)$ and action a in s with Equation 2.4. This also takes the current state value approximations of the critic and discount factor γ into consideration. $V(s)$ should be as close to $r + \gamma v(s')$ as possible, hence the error of $V(s)$ becomes δ.

$$\delta = r + \gamma V(s') - V(s) \tag{2.4}$$

The model performance data is stored in a replay buffer by using the initialized actor-critic model to perform actions in the environment. The replay buffer contains $(s, r, s', v(s), v(s'))$ pairs and is used to train the neural networks. This replay buffer is sampled in batches of size T

How the actor is trained depends on the type of actor-critic model. The advantage actor-critic model (which relates to the PPO algorithm) is covered in this case.

Assuming the actor Q-network π has trainable parameters θ, updated based on the policy gradient and TD-error δ. The critic is trained by using the loss δ^2 while the actor is a bit more complicated.

The loss function $J(\theta)$ is approximated using a batch from the replay buffer with size T according to Equation 2.5. With learning rate α the actor parameters θ is updated by gradient decent; $\theta = \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$.

$$J(\theta) = \sum_{t=0}^{t=T} \log \pi_\theta(a_t, s_t)\delta_t \tag{2.5}$$

2.5 Proximal Policy Optimalisation (PPO)

The PPO algorithm is an on-policy, online policy gradient method. It is similar to the advantage actor-critic model but ensures stability by restricting its value updates to not deviate too much from the old policy. It does this by clipping the probability ratio between the new and old policy [35].

While the PPO model trains the actor, a hyperparameter ϵ defines a trust region. The trust region $1+\epsilon$ and $1-\epsilon$ defines how much the new policy π_θ can deviate from the previous policy π_{θ_k}. The ratio of change in probability is defined by Equation 2.6, high $b(\theta)$ means that action a_t is much more likley in π_θ than in π_{θ_k}. The clipping of this ratio is performed by $c(\theta, \epsilon) = clip(b(\theta), 1-\epsilon, 1+\epsilon)$, which ensures $b(\theta)$ comprises of values between $1-\epsilon$ and $1+\epsilon$.

$$b(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_k}(a_t, s_t)} \tag{2.6}$$

The loss function of the PPO algorithm (Equation 2.7) is similar to the loss function of the advantage actor-critic model (Equation 2.5) but ensures stable policy updates. The main logic is that when the performed action a was reasonable $(\delta_t > 0)$, the policy update is limited to the trust region. The incentive to move $b(\theta)$ beyond the clipping range is removed.

$$J^{clip}(\theta) = \sum_{t=0}^{t=T} min(b(\theta)\delta_t, c(\theta,\epsilon)\delta_t) \tag{2.7}$$

It is worth noting that the minimum operator of Equation 2.7 cannot be simplified to just $c(\theta,\epsilon)\delta_t$. Since when $b(\theta) > 1+\epsilon \implies c(\theta,\epsilon) = 1+\epsilon$ and $\delta_t < 0$ leads to $b(\theta)\delta_t < c(\theta,\epsilon)\delta_t$. This happens when a large update has been performed, which increases the probability of taking worse actions $(b(\theta) > 1 + e \wedge \delta < 0)$, and therefore $b(\theta)\delta_t < 0$ is not limited by the clipping. Hence the minimum of these two terms is performed.

Equation 2.7 is to be maximized, and the critic is updated the same way as in the advantage actor-critic model.

In context of the ambulance dispatching problem, this ensures small updates to the policy π making it suitable for strongly stochastic environments.

2.6 Muti agent reinforcement learning

Multiple ambulances are involved in the context of the ambulance dispatching problem. It is a cooperative multi-agent reinforcement learning problem, where each agent in the environment aims to maximize its reward in cooperation. However, in other multi-agent reinforcement learning problems, the agents can be in a competition setting.

In a competition setting, the reward for each agent opposes each other. In such a setup, complex policies can arrise and has similarities to Game theory. However, in a cooperation setting, the reward is shared by each agent, leading to coordination to maximize the system-wide reward.

The collaborative setting for EMS has different perspectives of planning. It is possible to consider a policy for each ambulance. Meaning that each ambulance acts independently based on its local observations. This is also called decentralized planning.

Further centralized planning considers all the ambulances and all the current incidents. In other words, a centralized (policy) considers the system-wide observation and chooses an available ambulance to dispatch. This thesis implements such a centralized policy.

2.7 Dispatching

A literature overview of algorithmic and RL implementations to the problem of ambulance dispatching is outlined below.

2.7.1 Algorithmic Dispatching

Other optimization methods have been applied to this problem besides RL. These are methods such as linear programming [36], mixed integer programming [37], or tabu-search [38].

However the search space is very large, specifically there are $m!^{2n}$ possible dispatch orders for m ambulances and n zones [21].

Bandara *et al.*

Bandara *et al.* [21] sought a EMS dispatching methodology to increase patient survivability by incorporating the priority of the incidents. The probability of survival is modeled by a survival function ($S(t_R)$) of the response time (t_R). See Equation 2.8 and Larsen et al in Figure 2.2

$$S(t_R) = \max[(0.594 - 0.055 * t_\mathrm{R}); 0] \tag{2.8}$$

The study looked at 2x2 zones and solved the optimal policy with complete enumeration and commercial optimizer software. Further, it is assumed that ambulances are only dispatched from base stations, a constant poisson rate λ, and that response time is independent of priority.

The results indicated that the optimal policy largely depends on the demand balance between the different zones. When the demand between the zones was balanced, the optimal policy was always to send the closest ambulance, but not when there was an imbalance. In this case, the closest ambulance is always dispatched to acute incidents but not necessarily to urgent incidents. The optimal policy balances the lowest distance and ambulance busy probability for urgent incidents. Any urgent incidents from any of the zones are served by the closest ambulance in a lower-demand zone. This increases the overall average response time but increases the patient survivability.

This optimal policy is more intuitive for a small toy example, but the complexity increases for a realistic scenario. At the same time, the computation becomes intractable with higher granularity. Because of this reason, a heuristic was developed to simulate the optimal policy for higher granularity scenarios. However, this heuristic is deterministic. It yields the same urgent dispatch order for each zone and does not consider a changing Poisson rate for each zone.

2.7.2 Reinforcement Learning Dispatching

Mainly previous methods use some form of a MDP model [39][7][16], but other methods of RL have been applied. Such as Q-learning [40] and Approximate Dynamic programming (ADP) [1][6]. These implementations differ in considerations, methods, assumptions, and metrics. Most of these optimize with either coverage, survivability, or response time more directly. A common trend is to optimize for response time and coverage compared to survivability.

Many Reinforcement learning papers exist on vehicle fleet management and ride-sharing/hailing. Most commonly, articles focus on taxi fleet management [41] [42]. Papers focusing on Medical dispatching are covered here.

There are several benefits of Reinforcement learning over algorithmic techniques.

1. Reinforcement learning can better capture stochastic and dynamic environments.
2. Reinforcement learning also works well when spatial predictions are hard [42].
3. Reinforcement learning can work online (depending on the algorithm). Meaning it can adapt to changing circumstances.
4. Training a Reinforcement learning model can be slow. However once trained, inference time is fast, and the model can be updated after deployment.

From a reinforcement learning perspective, this problem is still complex. The action space and state spaces are ample, the environment is stochastic and dynamic, and multiple agents (ambulances) are involved in the environment.

Keneally *et al.*

Keneally *et al.* [19] implemented a MDP to solve the dispatch problem of military aeromedical assets. In other words, the dispatch of military medical evacuation helicopters to locations in combat areas.

This problem differs slightly from an urban EMS problem in two distinct ways. First, for a given call for evacuation, several incidents can occur with different priorities at the same location. Secondly, the response time of helicopters is more closely correlated with Euclidean distance.

The problem's mathematical formulation is similar and provides valuable modeling insights, especially since incident priority is considered a part of the reward function. Similarly to the Norwegian EMS system, incidents are labeled on a three-part priority scale. However, the response time limits of each priority level are more extended due to the problem it solves. Since the formulation of the reward function is more complex than for this application, a general outline is formulated.

The priority for a single patient for evac is modeled as a number $k \in \{1,2,3\}$ where the urgency decreases with the value of k. Each of these priorities has an associated desired time limit RT_k. Rewards are also given based on the value of k, defined as r_k, which follows the constraint $r_1 > r_2 > r_3 \geq 0$. By combining these rewards with the count of patients per priority for a single call for evac $c = (c_1, c_2, c_3)$, their respective probabilities q, a multi nominal probability distribution

f, and $\alpha = \sum c$, a utility function $u(c)$ is computed. See Equation 2.9. This utility is only received if the response time is lower than RT_k.

$$u(c) = \sum_{h=1}^{h=3} r_h c_h f(c_h|\alpha, q) \tag{2.9}$$

The reward function uses a probability distribution f over the priorities due to possible priority classification errors made by the dispatcher (Over-labelling). This work builds on McLay and Mayorga [16], which considered optimal ambulance dispatch in an urban EMS with consideration of classification errors of priorities.

An assumption is made that the state-transition function of the MDP model can be solved in closed form and follow a memoryless distribution. This is a shortcoming since it is assumed that the optimal policy is stationary. One policy is assumed to be optimal irrelevant of the time dimension.

McLay and Mayorga

The impact of patient priority classification errors on the optimal dispatching policy was studied by McLay and Mayorga [16]. A MDP was trained, which is scaled to a discrete-time problem using uniformization.

It is assumed that response time is independent of the assessed priority and that the patient's absolute risk is known retrospectively (After an ambulance has arrived). When the MDP makes its decisions, it does not see the patient's absolute risk, but it is known after a patient is treated, which is reflected in its reward function.

A three-part-priority scale was utilized, and two different cases were investigated. In these cases, the priorities are mapped into a high and a low-risk class, corresponding to over and under-responding to incidents. In the first case, only acute incidents were considered to be high risk (under-responding). While in the second case, both acute and urgent were regarded as high risk (over-responding). All other priorities, which are not high risk, were considered as low risk.

Based on these high and low-risk classes, there is uncertainty about the actual risk level of the patient (known retrospectively). A variable α is used to adjust this uncertainty about the prospectively assessed priority. α denotes the proportion of acute incidents being genuinely high-risk to urgent incidents being truly acute. The impact of patient classification errors on the dispatching policy was studied by varying this variable. When α is low, there is a high rate of classification error.

The state and action is defined over ambulances, while rewards are over incident coverage. In the state, an ambulance takes on the value 0 if available and the incident location it serves if it is busy. The reward function is defined as the probability that an actual high-risk patient H^{T} is covered at location l, given an ambulance j responds to the incident with assessed risk level $h \in \{H, L\}$. The event R is where a patient is covered within a 9-minute time limit. This reward function relies on the quality of the priority classifications, which is determined by α. See Equation 2.10.

$$u_{ij}^{h} = P_{R|H^{\mathrm{T}} \cap h \cap i \cap j} P_{H^{\mathrm{T}}|h \cap i \cap j} \tag{2.10}$$

The results indicated that it is better to over-respond when there is a high rate of classification errors. Specifically, when $\alpha < 0.45$, it is better to over-respond. i.e., treating urgent incidents as acute. This is because there is a high probability that the urgent labeled incidents are actually acute. However, this also implies that under-respond is better when there are better classifications ($\alpha > 0.45$). In other words, urgent incidents should only be considered as low risk when there is great certainty.

With exponentially distributed response times, the MDP model is only slightly better than always dispatching the closest ambulance. When there are high classification errors, the closest ambulance is almost always dispatched by the policies. Across all values of α, there is only a slight improvement in coverage.

It is however argued that exponentially distributed response times is unrealistic. With log-normal distributed response times which is more realistic there is a statistical improvement in coverage by the model.

It is pointed out that consideration of multiple types of EMS vehicles, identification of scalable heuristics, and work-load balance needs further research. Especially since it was shown that the optimal policy most of the time resembled a priority list of the incidents, which can be expressed with algorithms and heuristics.

Liu *et al.*

The work of Liu *et al.* [40] built a Multi-agent Q-network with Experience Replay (MAQR) (off-policy) to tackle the ambulance dispatch problem. Experience replay is a technique in reinforcement learning which stores previous experiences of the model in the environment, which is then later used to train the model in a supervised manner.

Each ambulance is represented as an agent in the environment. In order to reduce the computational burden, every ambulance with the same base station is set to share the same policy. This means they designed one policy for each base station instead of each ambulance. This reduces the computational burden heavily since it greatly reduces the number of policies to train.

The state and actions in the environment are based on a grid representation with a time horizon of 5 minutes. The action in the environment was defined as the grid the given ambulance would travel to. The state representation is, however, more complex. It consisted of three components: available ambulance distribution, ambulance request distribution, and waiting time distribution. It does not represent the acuteness of the ambulance requests.

The reward for each agent is shared in the environment. This is done to reduce the computational burden of the problem. The reward itself is defined as a weighted sum between the number of incidents at the grid zone, round trip time, and the sum of waiting time at the grid.

To further reduce the computational cost, a single Q-network represents the policy for each base station (and further for each ambulance). Each base station is then distinguished using its ID.

To train the Q-network, they made a simulation model. This was done by compressing the real-world data into a single day and then sampling using a binomial distribution with $p = 1/361$. This method works to make a general real-world simulation; however, it might not capture the seasonality of the volume and spatial pattern of incidents. Since the volume and spatial pattern of incidents heavily depend on the time of day and month.

Different heuristics for ambulance dispatching is compared towards the trained model to evaluate the results. The most realistic algorithms are location-based allocation (LBA), which assigns ambulances to their nearest request, and time-based allocation (TBA), which assigns ambulances to the request with the longest waiting time. The results of this paper showed that their method outperformed the two mentioned heuristic algorithms.

Mukhopadhyay *et al.*

Mukhopadhyay *et al.* [7] implemented a Semi Markov decision process (SMDP) and a parametric survival model for a complete pipeline for responder dispatch. In other words, both forecasting of incidents and a dispatch model were implemented. A grid-based environment is also assumed. A Long short-term memory (LSTM) model is also used to predict traffic on the road network with data retrieved from Here maps API.

Compared to other papers, the main contribution is implementing a complete system in a tractable online manner. Prior work on incident prediction has relied on batch learning, which Mukhopadhyay *et al.* claims fails to capture the latest trends in the data. The same applies to the dispatch model; prior work utilizing a MDP is intractable since the state transition probabilities are computed using canonical policy iteration. This method is prolonged and needs to be retrained if the parameters of the environment are changed.

A critical distinction from prior work using canonical policy iteration is that one policy is not trained to be optimal for the entire state space. The policy does not need to be optimal in states where there is no possible action to perform. For example, there are many states where there is no incident to dispatch an ambulance or ambulance to dispatch to an incident. Instead of learning an optimal policy for all states, a sub-MDP is trained around the neighborhood of the state where an action needs to be performed (decision-making state).

A grid-based representation represents the state and actions. The state consists of a tuple with information about incidents, ambulances, and environmental factors. The incidents waiting to be served are represented by their grid-id and sorted by their time of occurrence. The ambulances contain information about their location and their availability. The actions correspond to sending ambulances either to their base station or incidents.

Similar to Liu *et al.* [40], a simulation model is built to train the dispatch model. In this case, the simulation model combines the online incident prediction and ambulance response time models. The incident prediction model uses Poisson regression, and its parameters are updated after each simulation week. While the simulational model uses actual incident data, the prediction model is used as a generator to produce likely incident spatiotemporal distributions into the future. The same goes for the ambulance response time model, which is used as a generator and by the simulator.

There are several ways to handle the spatial distribution using Poisson regression. Such a model can either be trained for all the locations, one model for each location, or by learning the spatial distribution using hierarchical clustering on the data (which is explained in their prior work [17]).

The incident prediction model is set up such that it is online. After a week has passed when going through the dataset of incidents, the latest week is used to update the model using Stochastic gradient descent (SGD). These newly trained coefficients are only used if the loss has decreased.

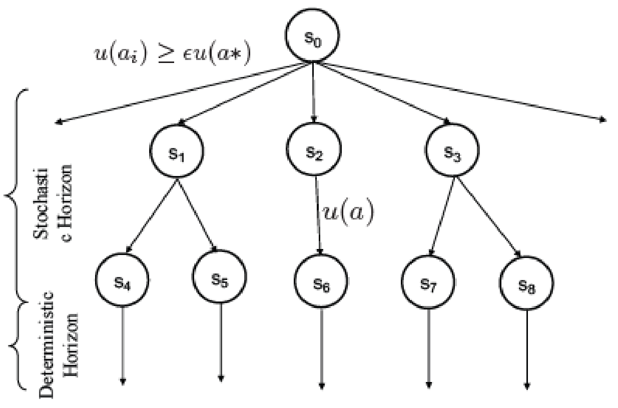

Figure 2.5: State action tree utilised in Mukhopadhyay *et al.*

To train the sub-MDP the paper utilizes a Monte Carlo tree search (MCTS) with a stochastic and deterministic horizon (Figure 2.5). In addition, the incident prediction model and ambulance response time model are used as generators to build the Monte Carlo Tree into the future. The depth of the tree determines these horizons, which again define candidate actions. The stochastic horizon is at the top of the tree, while the deterministic is at the bottom. Each action at least ϵ times better than a heuristic-based action is considered in the stochastic horizon. This heuristic action assigns ambulances to their nearest request. While in the deterministic horizon, only this heuristic-based action is considered. The Monte Carlo Tree is built to a certain depth.

To find the best overall action, multiple incident chains are generated and for each chain, the MCTS is built. The estimated rewards for each action are then averaged over each chain. The best action is then the action with the lowest average response time.

Results showed that with the trained policy, few incidents had different actions than the heuristic-based. Less than 1% of incidents benefited from their policy, however, 38s were saved on average. Abbreviation studies showed that by reducing the number of base stations, the number of incidents impacted by their policy increased.

2.8 Discussion

A discussion among the literature is outlined below. This draws a line between the methodology (chapter 5), data analysis (chapter 4) and this literature overview.

2.8.1 Reinforcement learning

Citation	**Year**	**Method**	**Focus**	**Reward**	**Environment**	**Training method**	**Online policy updates**
Schmid [1]	2012	ADP	Pure dispatching	Response time	CAD data / Syntetic	Value iteration	No
McLay and Mayorga [16]	2013	MDP	Patient classification errors	Coverage	CAD data	Uniformasation	No
Nasrollahzadeh *et al.* [6]	2018	ADP	Dispatching / relocation	Priority adjusted response time	CAD data / Syntetic	Apprximate policy iteration	No
Mukhopadhyay *et al.* [7]	2019	SMDP	Complete pipeline	Response time	CAD data / Syntetic	MCTS	Yes
Liu *et al.* [40]	2020	MAQR	Pure dispatching	Wait time Count incidents Round trip time	CAD data / Syntetic	Experience replay	No
Elfahim *et al.* [43]	2022	DQN	Pure dispatching	Response time	CAD data	Experience replay	No
Hua and Zaman [44]	2022	TD	Pure dispatching Augmented transition probabilities	Response time	N/A	Policy iteration	No

Table 2.1: Table of most recent research on the ambulance dispatching problem. Using RL

Table 2.1 provides a table of the most recent research on the application of RL to the problem of ambulance dispatching. Some of these papers are not described in detail in this chapter.

Below is a discussion about the previous solutions provided and possible new solutions.

Model

In older literature, it is more common to use MDP models; a more common trend is to use more sophisticated model-free models such as ADP, MAQR, Deep Q-network (DQN) and MCTS.

All of these models face the issue of the ambulance dispatching problem being a non-homogeneous Markov decision process. In other words, the state transition probabilities have temporal instability.

The MDP of the ambulance dispatching problem can also be seen as partially observable (POMDP). Since the state, which includes the current incident locations, might not unveil enough information about the future incident distribution.

An assumption for the usage of MDP in general is that the Markov property holds for the environment, which might not be the case for the ambulance dispatching problem. Since the future spatial distribution of incidents might depend on several past states rather than only the current state (Markov property). This is why papers like Mukhopadhyay *et al.* [7] use techniques like MCTS, which relaxes the Markov property.

Furthermore, Mukhopadhyay *et al.* [7] uses SMDP, which fits better to the problem of ambulance dispatching than MDP. Since in SMDP the time between decision-making states (ambulance needs to be dispatched) can be random [45]. In MDP, this time (sojourn time or time-step) is assumed to be static for each state.

Considering all of this, policy gradient methods [46] (such as PPO), other model-free RL methods (such as MAQR and Deep Q-network (DQN)), and MCTS might suit the problem more. These methods relax the Markov property and approximate the state transition probabilities.

Reward

Few mentioned papers consider the incidents' priority in their reward function. However, some papers consider incidents' priority for a different context, such as military evac Keneally *et al.* [19], patient classification errors McLay and Mayorga [16], and algorithmic dispatch [21].Nasrollahzadeh *et al.* [6] is the only found RL paper which directly incorporates the priority of the incidents into the reward. This thesis draws inspiration from Bandara *et al.* [21] to incorporate incident priority.

In a more realistic ambulance dispatching scenario, the work of Keneally *et al.* [19] can be used to model multi-casualty incidents in combination with classification errors. This is however, outside the scope of this thesis.

Interestingly Liu *et al.* [40] used waiting time in the reward; this is wise since it directly correlates with response time. Furthermore, it is monotonically increasing over every state an incident is waiting for an ambulance, which depends on how long the response time is.

State/action space

A huge part of implementing a RL dispatch system is defining its state and action space. The state and action space is vast since there are many locations where incidents can occur and ambulances can travel to.

Action space

It is possible to define the action space among ambulances instead of among locations since there exist fewer ambulances than locations. However, this must be retrained if the number of ambulances increases. Liu *et al.* [40] used a grid-based actions while Mukhopadhyay *et al.* [7] and McLay and Mayorga [16] used a selection of ambulances.

State space

Interestingly Liu *et al.* utilized probability distributions on locations among available ambulances, incidents, and waiting time.

The state is then (N_g, N_g, N_g) large with N_g number of grids in the environment. This can be interpreted as an image with 3-channels, which is helpful for neural networks. This thesis draws inspiration from this concept.

A simple way of representing the state space as an allocation. In other words, with N_b base stations, $S = (f_a k)_{k<N_b}$, and $f_a k$ is the number of ambulances at base station k. However, the location of en-route ambulances must also be captured since the best ambulance may be en route to its base station.

2.8.2 Forecasting

The previous work also uses one-hot encoded time features. Experimentation with cyclical time features can be performed. Cyclical time features use trigonometric functions to capture the cyclical nature of time in the features.

It is also possible to utilize auto-encoders for spatial prediction. Autoencoders are part of a new paradigm in machine learning called Self-supervised learning. It can be trained without human-labeled data and reduces high-dimensional data into a small vector. This can then be used to reduce the large dimensionality.

Estimating the spatial distribution by the time since the last incident occurred at a given location is also possible. This fact is utilized in subsection 5.1.1.

Since spatial distribution is the hard part of the prediction, aggregation is essential but comes at the cost of lost precision. Some way to cluster locations to base stations with minimal travel time, such that the spatial predictions match the responsibility areas for each base station. This is looked at in subsection 3.3.3.

2.8.3 Allocation

Since the population-proportionate allocation performed better than the optimized solutions, an assumption can be made that population plays a crucial role in incident spatial distribution.

The simulation model built by Schjølberg and Bekkevold [2] can be used in RL context. However, this model is written in Java which has limited Machine Learning support. Because of this reason, a simulation model is implemented in Python and is explained in the method (chapter 5).

Chapter 3

Tools and Datasets

This chapter outlines the tools and datasets used in this thesis. Which also includes the data preprocessing of the datasets, and general methodology for chapter 4.

3.1 Datasets

There are two big datasets used in this thesis. The first is a dataset with ambulance incidents retrieved from OUS, while the other dataset contains the road network of Oslo and Akershus, retrieved from OSM. These datasets are discussed further in the sections below.

3.1.1 Road network dataset

OSM is an open-source map tool. It is freely available for all interested parties, and the maps it provides are also updated regularly. The maps can also be downloaded as a Protocolbuffer Binary Format (PBF) file, which then can be analyzed for research purposes. OSM provides one such map file for Oslo and one for Akershus. These two files were merged to create a complete map of Oslo and Akershus.[1]

The road network is represented as a directed graph $G(V, E)$. In this graph, the nodes V represent a location on the map, while the edges E represent the road between them. The nodes do not provide any helpful meta-information other than latitude and longitude on a road segment in Oslo and Akershus. On the other hand, the edges give a lot of detailed information such as speed limit, width, number of lanes, type of road, oneway road, surface type, road name, smoothness, and much more. These nodes and edges describe the road's topography and meta-information in the spatial dimension.

[1]Available: https://github.com/JonEliasMoen/Amb-public

Only the subset of the dataset containing roads for driving will be considered, as the entire dataset includes roads for walking, cycling, and driving.

The dataset does not contain any traffic information, which heavily influences travel time.

3.1.2 Incident dataset

The Incident dataset is an anonymized dataset of ambulance incidents, meaning the location of the incidents is aggregated into 1x1 Km grids of Oslo and Akershus. The dataset contains 2597 such grids with about 752k incidents ranging over eight years from 2001 to 2019 (2001, 2002, 2005, 2015-2019). The grids in the dataset are combined with population data from SSB.

It contains extensive meta-information about the incidents and ambulances dispatched. For example, it includes which type of ambulance was sent, ambulance id, grid location of the incident, and incident priority. It also contains timestamp information about the different stages of the EMS process discussed in chapter 1.

The dataset does not contain any information about where the ambulance drives from. A likely scenario is that the ambulance drives from the closest base station. Still, it is also possible that the ambulance could have been assigned while traveling to either a base station, hospital, or another incident.

3.2 Tools

The programming language Python with the Anaconda distribution is utilized for this thesis. The most standard packages used are Pandas, GeoPandas [47], Numpy, Networkx, and Matplotlib. The Open Street Map (OSM) dataset is handled by the packages Osmium, Osmnx, and Pyrosm.

For map visualization purposes, several packages are utilized—mainly the package plotly together with API connection to Mapbox. For more extensive visualizations, Geopandas, folium, and OpenRouteService API are employed. The OpenRouteService API again uses OSM.

For optimization of distributions, the package SciPy and Fitter is utilized. Stable Baselines is used for the RL implementation. [48].

3.3 Dataset Preprocessing

This section outlines the data preprocessing of the datasets mentioned earlier in the chapter.

3.3.1 Incident dataset

The same data preprocessing as Schjølberg and Bekkevold [2] is applied. Which means that the data ranging from 2015 to 2018 is included, and incidents which happen with identical timestamps and location is treated as the same incident.

3.3.2 Open Street Map (OSM) dataset

To represent the environment of the incident dataset, the OSM dataset is utilised. This represents the road network of Oslo and Akerhus, with a detailed directed graph G.

A lot of preprocessing was performed to reduce the number of nodes and edges in the graph. There were a lot of redundant nodes and edges only to describe the exact shape of the roads, which made the graph intractable for travel time estimation.

Make intersection graph

In order to make $G(V,E)$ tractable, It was reduced to roughly only contain nodes at road intersections. Edges then represented a continuous road from one intersection to another. *makeIntersectionGraph* (Algorithm 1) shows how this was performed, an elaborate textual explanation is also given in section A.1. This process takes a road network graph G and returns an intersection-only graph G_i.

Make grid-cell graph

The incident dataset uses a spatial granularity of 1x1km grid cells; therefore, G_i is similarly divided. Each node in G_i has a latitude and longitude, which is used by a spatial join from GeoPandas to label each node by their grid cell id l. Mathematically speaking G_i is divided into subgraphs $\forall_l G_i|l$. Where $G_i|l$ is the intersection-only graph for the l incident dataset grid cell.

Due to the *makeIntersectionGraph*, grid cells that contain a road but no intersection would contain no nodes since only intersection nodes are kept in G_i. For example, this happens in grid cells with highways, which may have long stretches without any intersection. Due to this, the *makeIntersectionGraph* was performed such that these grid cells would contain at least one node.

A new graph $H(V_h, E_h)$ is derived from $\forall_l G_i|l$. The nodes V_h take on coordinate values k, corresponding to which subgraph $G_i|k = l$ it represents. Given nodes $A, B \subset V_h$ with coordinate values k_a and k_b, an edge E_h is added to H if there exists any edge which connects $G_i|k_a$ and $G_i|k_b$. This is shown in Equation 3.1, a simple example in Figure 3.1, and a textual explanation in section A.2. Notice in the figure how not all cells are connected.

Algorithm 1 *makeIntersectionGraph*. The $G|con$ statement denotes part of G where the condition con is true. The $\cup$ operator is used to denote the set Union operator (merge) of graphs

```
Require: OSM graph $G(V,E)$. $E$ has direction features $i,j \subset V$.
Ensure: Intersection graph $G_i$
 1: $G_o = G$                                  ▷ Copy original graph
 2: G = $G|i \neq j$                           ▷ Remove self referring edges from G.
 3: u,c = uniqueValuesAndCounts(E)
 4: G2 = $G|c \leq 2$   ▷ Remove intersection nodes by having degree bigger than 2.
 5: G = $G|c > 2$                              ▷ Keep only intersection nodes in G
 6: $S$ = connectedComponents(G2)              ▷ Find all connected components in G2.
 7: k = 0
 8: for $s \in S$ do
 9:     $E_{G2}(l) \subset s = k$              ▷ Label edges by index in S
10:     k = k + 1
11: end for
12: $G2$ =groupBy($E_{G2}(l)$) ▷ Group G2 by s. (1 component = 2 nodes and 1 edge)
13: $G = G2 \cup G$                            ▷ Recombine graphs
14: $S$ = connectedComponents(G)               ▷ Find connected components in G.
15: $G_i = S_0$                                ▷ Init $G_i$ by first subcomponent
16: for $s \in S$ do                           ▷ Merge subcomponents by original graph $G_o$
17:     con = $\exists E_{G_o} \Rightarrow$ connected(s, $G_i$) ▷ Exist edge in $G_o$ which connects s to $G_i$
18:     if con then
19:         $G_i = G_i \cup (G_o|con) \cup s$  ▷ Add these edges to $G_i$
20:     end if
21: end for
22: return $G_i$
```

$$Connected(G_i|k_a, G_i|k_b) \Rightarrow E_h(A,B) \tag{3.1}$$

Further, the value of $E_h(A,B)$ is defined as the average time it takes to travel from any node in $G_i|k_a$ to any node in $G_i|k_b$. This time is based on the speed limit and length of the roads ($t = \frac{s}{v}$). Which then provides the minimum time it takes to travel along any road. This average is based on a sample size of the number of nodes in the representative subgraphs. Specifically, the sample size is $\frac{u}{2}+1$, where u is the number of nodes (intersections) in one of the subgraphs.

These travel times ($E_h(A,B)$) were validated towards the travel times by Open-RouteService and Google Maps. The centroids of two grid-cells is used as input to OpenRouteService and Google Maps. When the travel times of Google Maps was not affected by traffic, the estimates was within $\approx$ 5m to that of grid-cell graph $H(V,E)$. Generally it was most equal to the travel times provided by Open-RouteService, which is also based on OSM.

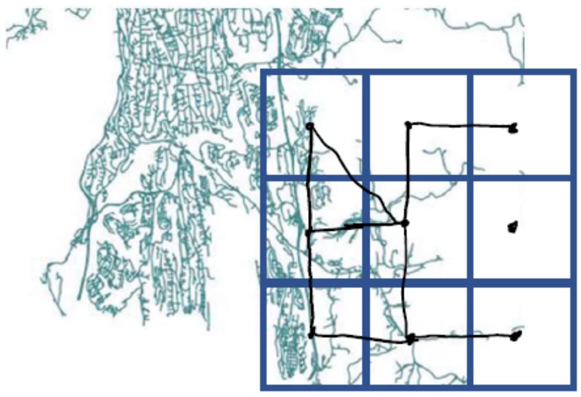

Figure 3.1: Shows a simple example of the construction of graph H. The example only shows nine grid cells but is done for all grid cells. Each cell has a marked centroid and edges to other cells if a road connects them (Background image source [49]).

The result is a graph H, which represents the environment. Each node is a location where incidents can happen, and the edges represent if it is possible to travel by car between these locations and the amount of time it takes. Graph H is shown in Figure 3.2, and is utilized throughout the rest of this work.

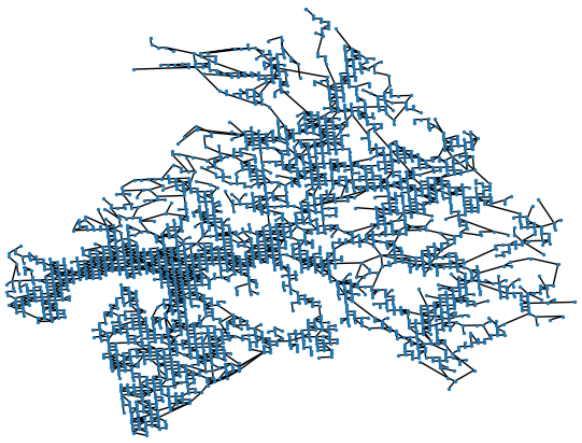

Figure 3.2: Shows the resulting road network graph of Oslo and Akershus.

If the appropriate Protocolbuffer Binary Format (PBF) file is given, this preprocessing methodology can also be applied to any area.

3.3.3 OpenRouteService

Isochrones can be derived by collecting the base stations from the incident dataset and utilizing OpenRouteService API. Isochrones are polygons where the shape on a map shows how far a particular form of travel can come traveling from a given geographical location within a given time limit. Typically several such overlapping polygons are evaluated for different discrete time limits. It is essentially a spatio-temporal discretization of travel times.

These isochrones are utilized to estimate the coverage of the ambulances stationed at their base station. Overall these isochrones give a visual explanation of the concept of coverage in the context of the ambulance dispatching problem. Furthermore these also potentially be used in future work to divide Oslo and Akershus into distinct spatial responsibility areas for each base station.

Mathematical explanation

Given a starting location s, some other location L, a travel-time function $T(s, L)$ and n time limits $T_l = (t_1, t_2, t_{...}, t_n)$ the isochrone which location L belongs to can be described mathematically. The continuous travel-time function returns how long it takes to travel from location s to L in the same unit as the time limits T_l. Location L belongs to isochrone i if $t_{i-1} \leq T(s, L) \leq t_i$. It can be considered a discretization of the travel time to each location.

There are some particular case scenarios. For example, if $T(s, L) > t_n$ means location L doesn't belong to any isochrone. Further, with k starting locations $s = (s_j, s_{j+1}, s_{j+2}, s_k)$, the same procedure is applied but with minimum travel time to location L. See Equation 3.2.

$$T((s_j, s_{j+1}), L) = \min_T \forall_j T(s_j, L) \tag{3.2}$$

Usage

These isochrones are evaluated for several starting locations (base stations) with the same overlap technique mentioned before. This is done for average car driving with time limits of 3, 5, 10, and 15 minutes. The result shows the estimated coverage of the base stations when driving from the base stations. This does, however, not consider traffic, but it does consider the road network and intersections. An example of one such isochrone is shown in Figure 3.3

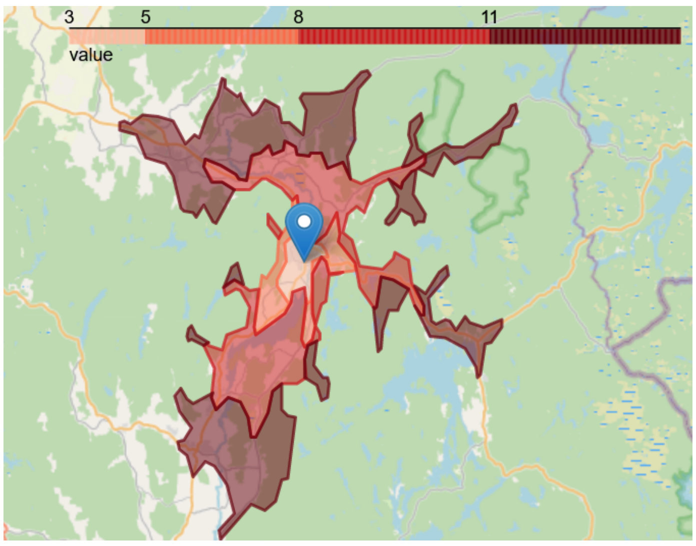

Figure 3.3: Shows an example of one isochrone from the base station in Bjørkelangen.

Chapter 4

Data Analysis

Since Hermansen [13], Schjølberg and Bekkevold [2] and Van De Weijer and Owren [12] all have written master theses using this dataset, data analysis provided here is aimed as supplemental analysis to provide valuable insights, and support the simulator's implementation (chapter 5).

4.1 Incident data analysis

This section provides a data analysis of the incident dataset. It is divided into sections exploring different aspects of the dataset.

4.1.1 Reponse time

The response time ($t_r = t_5 - t_1$, recall Figure 1.1) for a given incident i varies greatly depending on which priority $i_p \in \{A, H\}$ is assigned. As seen on Figure 4.1 (left), Acute incidents have a much lower response time on average than Urgent. Formally $(\overline{t_r}|i_p = A) \approx 12$ and $(\overline{t_r}|i_p = H) \approx 25$ minutes, which implies Equation 4.1.

$$(\overline{t_r}|i_p = A) < (\overline{t_r}|i_p = H) \tag{4.1}$$

The right plot on Figure 4.1 shows histograms of the response time for each priority $t_r|i_p$. This plot shows that the standard deviation of the response time for Acute incidents is much lower than for Urgent. (Equation 4.2). It might also look like there are fewer Acute incidents than Urgent incidents, but there are around 12% more Acute incidents than Urgent.

$$(\sigma(t_r|i_p = A)) < (\sigma(t_r|i_p = H)) \tag{4.2}$$

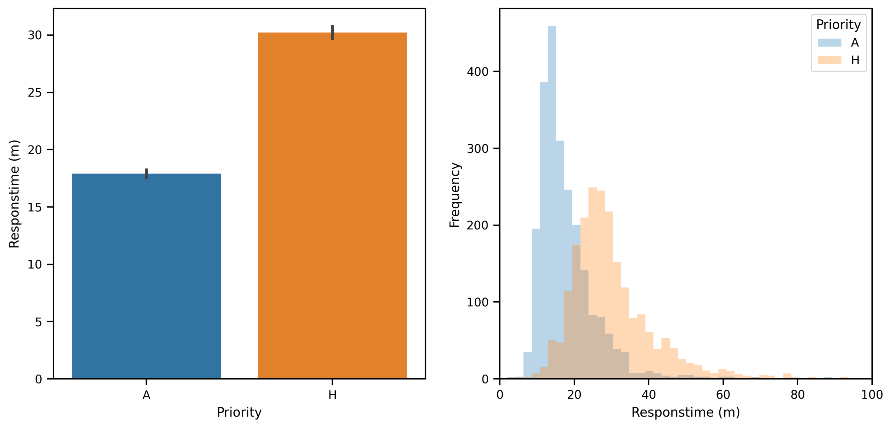

Figure 4.1: (Plot left) Average Responstime per priority, with marked 95% confidence interval. (Plot right) histogram plot of responstimes per incident.

Makeshift survival function

As discussed in section 2.1 there is a non-linear relationship between response time and survivability, which is described by a survival function (Figure 2.2). Such a survival function can be used to describe the relative importance of an incident (survivability) given response time t_r and incident priority i_p. In other words, these describe how much more important each incident priority is compared to each other as a function of t_r.

However OUH does not have such a survival function for Oslo and Akershus, therefore a survival function for each priority is improvised. The constructed survival functions is the inverse CDF of the response time t_r for each priority i_p, estimated from the dataset (Figure 4.2). See section A.3 for a mathematical explanation of this estimation.

It is worth noting that these do not represent any actual scientific accurate survival function. It is constructed based on low the reponse time is per priority, which is assumed to be some measure of the actual urgency of the incident. It estimates how well a given response time is for each priority compared to the data available.

Processing time

In Figure 1.1, some time is not spent on driving. Instead, this is time used by the dispatcher and ambulance crew to perform tasks. These different times are denoted t-d, i-d, and h-l on Figure 1.1, and are dependent on the priority of the incident. Histograms of these times per priority are shown in Figure 4.3, together with fitted probability distributions. The parameters for these distributions are given in Table 4.1.

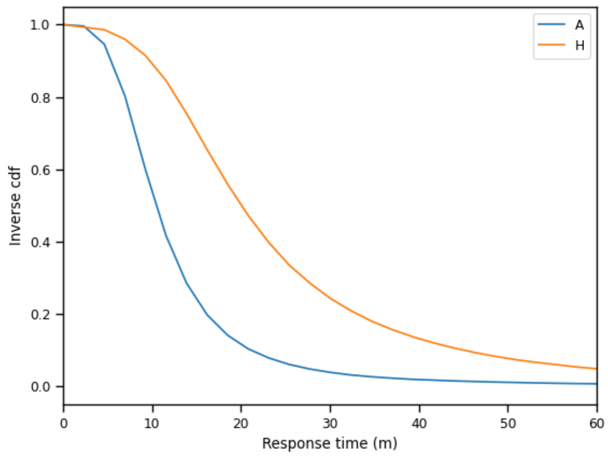

Figure 4.2: Shows the inverse CDF of the response time for each priority. In other words, the resulting makeshift survival function

After an incident is registered at the EMS, there takes some time before an ambulance starts driving towards the incident (t-d $= t_4 - t_1$, plot left in Figure 4.3). This time is spent by the dispatcher to asses the incident and by the ambulance crew to prepare for departure. On average, this time is considerably lower for Acute incidents than for Urgent (2.5 vs 6m).

Once the ambulance arrives at the incident location, the ambulance crew spends time treating the patient before driving to the hospital (i-d $= t_7 - t_5$, Plot middle in Figure 4.3). On average, this takes slightly longer for Acute incidents than for Urgent. This is most likely because the incident is more severe and takes longer to treat.

Once an ambulance has delivered a patient to the hospital, it also takes time before it is considered available again (h-l$= t_9 - t_8$, Plot right in Figure 4.3). The ambulance crew spends this time resetting equipment and preparing the ambulance for departure. On average, this takes about the same time per priority. However, there is a higher chance of this time being higher for Acute than for Urgent (20-40m range). This is probably because more equipment was used in the ambulance, which needs to be handled.

4.1.2 Incidents per location

Population

The higher the population that lives at a given location, the more likely it is that more incidents will occur there. This is explained by the regression line on Figure 4.4. However, this correlation is weak, with an R2-score of 0.16 and mutual

Priority	Type	Distribution	Parameters	Sum Squares
A	t-d	cauchy	loc=3.1, scale=1	0.005122
H	t-d	cauchy	loc=7.3 scale=3.5	0.004374
A	i-d	rayleigh	loc=0.9, scale=15.9	0.000661
H	i-d	chi2	loc=5.7, scale=0.3, scale=3.1	0.001041
A	h-l	chi2	loc=11.8, scale=-1.4, size=1.54	0.00119
H	h-l	lognorm	loc=0.3, scale=-4.6, size=18.2	0.000714

Table 4.1: Table of fitted distributions

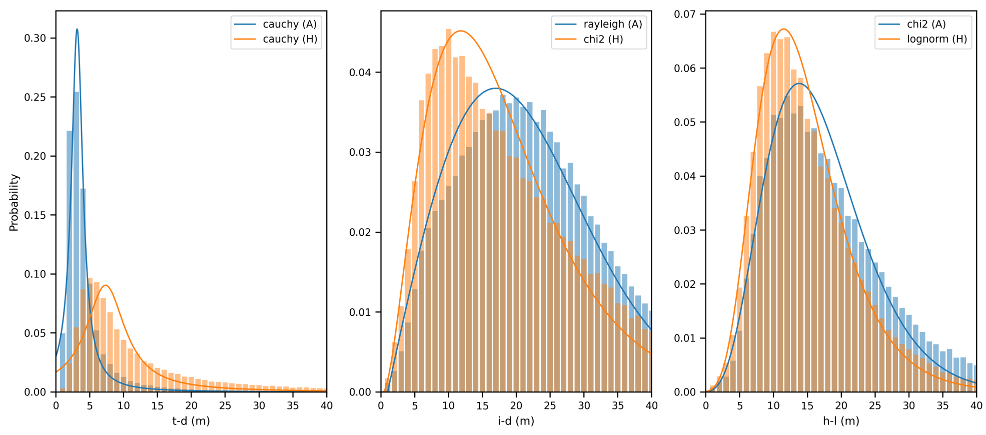

Figure 4.3: Shows a histograms of the different transition times with acute and urgent priority. (Plot Left) Time before starts driving to incident after a call is registered. (Plot Middle) Time before starts driving to hospital after arriving at the incident location. (Plot right) Time before ambulance is available after arriving at the hospital

information of 0.28, between the dependent and independent variables.

The main outliers of the linear regression model can be explained by how people travel. So locations that fit well to the regression line are where people live, while the main outliers are where people travel. This can be occasions like work, shopping or general commute, mainly in city centers. This fits well with the fact that according to OUH most incidents occur in homes. To illustrate this point, Figure 4.5 shows a map where bigger circles and more yellow color mean higher deviation from the regression line.

Another explanation can be that the outliers are locations where people outside Oslo and Akershus commonly travel to.

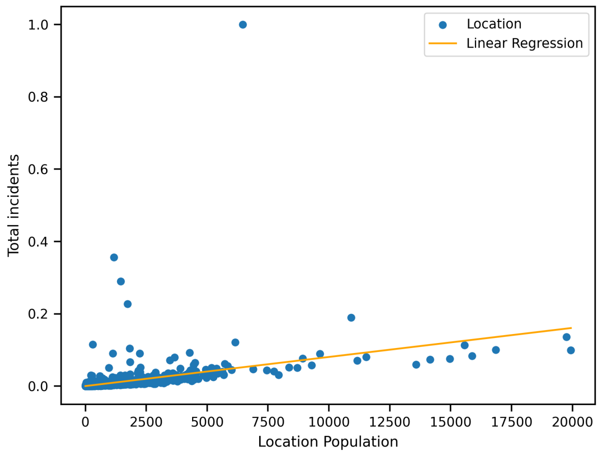

Figure 4.4: Showing a linear relationship between population and total amount of incidents

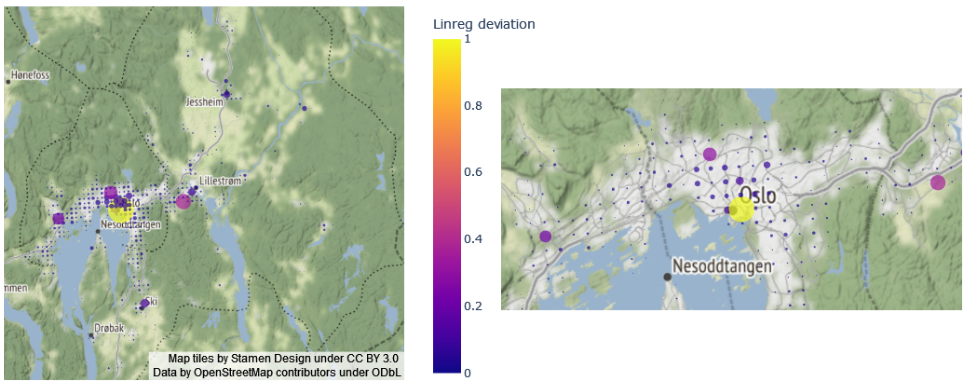

Figure 4.5: Map of deviation from population linear regression. Both a zoomed in and full perspective. Full map available online ht-tps://joneliasmoen.github.io/popLinreg.html

4.1.3 Incidents over time

Incident increase per week

Over the years, there has been a general increase in EMS demand. This can be due to an increase in population in Oslo and Akershus. Therefore, more accurate total population data for Oslo and Akershus is retrieved from Eurostat and interpolated for each week. This is compared to the total number of incidents.

Figure 4.6 shows number of incidents per week through the dataset. It also shows interpolated increase in inhabitants for Oslo and Akershus. The correlation between these two are 0.76 Pearson and 0.79 mutual information.

The correlation depends on the aggregation of the number of incidents. For example, the correlation drops to 0.6 Pearson and 0.4 mutual for daily aggregation. While with monthly aggregation, it increases to 0.8 Pearson and drops to 0.5 mutual.

Population increases generally influence the increase in EMS demand.

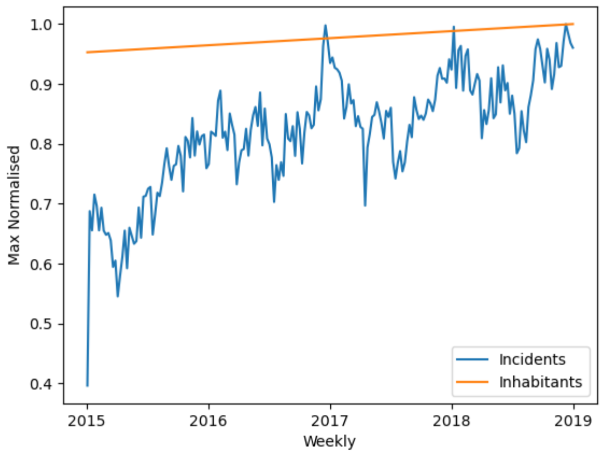

Figure 4.6: Increase in volume of incidents weekly. Compared to interpolated increase in inhabitants

Incident interval distribution

On average, the data shows that an incident occurs every 5.16 minute, with most incidents occurring within every minute. This is low due to the large area the dataset covered. The incident interval is the time between two incidents occur; in literature, the average of these values is called Mean time between failiure (MTBF). The maximum incident interval is around 2 hours or 124 minutes.

Figure 4.7 shows a histogram of the incident intervals, where each bin is one minute long. The X-axis is in minutes while the Y-axis is normalised counts. On top of this is a fitted exponential distrobution $f(x) = \lambda e^{-\lambda x}$ where $\lambda = 0.21$.

The Exponential distribution is known to fit a Poisson process since it is the inverse of the Poisson distribution. However, the λ parameter change over time, making it a non-homogeneous Poisson process. Therefore the λ value found is the average across the dataset ($\overline{\lambda(t)} = 0.21$). This is also further discussed below.The Exponential distribution is known to fit a Poisson process since it is the inverse of the Poisson distribution. However, the λ parameter change over time, making it a non-homogeneous Poisson process. Therefore the λ value found is the average

across the dataset ($\overline{\lambda(t)} = 0.21$). This is also further discussed below.

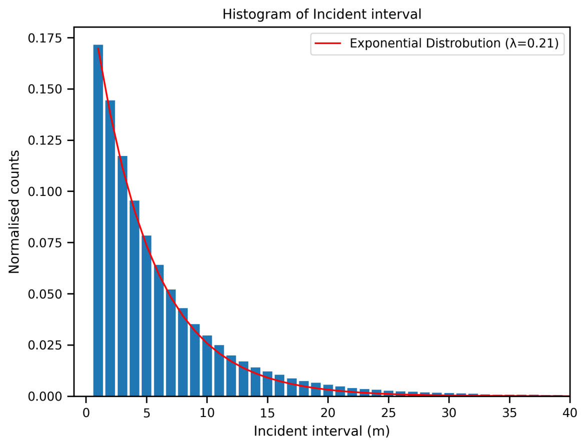

Figure 4.7: Histogram of incident interval per minute

Incident frequency hourly

On Figure 4.8 the X-axis is the number of incidents occurring within an hour, and the Y-axis is the number of hours with the given number of incidents. The Y-axis values are normalized to sum to one. On top of 4.8 are fitted Binomial, Poisson, Chi2, and Log-norm.

The Poisson distribution was first assumed since it is a discrete distribution that models variables with fixed interval events. However, the EMS demand is known to be a non-homogeneous Poisson process, which means that the interval change over time (not fixed) [13]. Because of this, the fixed Poisson and binomial distribution do not fit well; however, the chi2 and log-norm distributions fit better. The chi2 and log-norm distributions were applied since this distribution is essentially any multiplicative product combination of the values in Figure 4.7 such that the number of incidents per hour matches the X-axis. This assumption also fits well with the non-homogeneous Poisson process assumption.

The results show that on average 10.5 incidents occur every hour, as a consequence of non-homogeneous Poisson process.

4.2 Open Street Map (OSM) dataset

Recall the grid-cell graph H made in section 3.3.2, which forms the basis of this analysis.

Plotting a histogram of the degree of the nodes in the grid-cell graph gives an

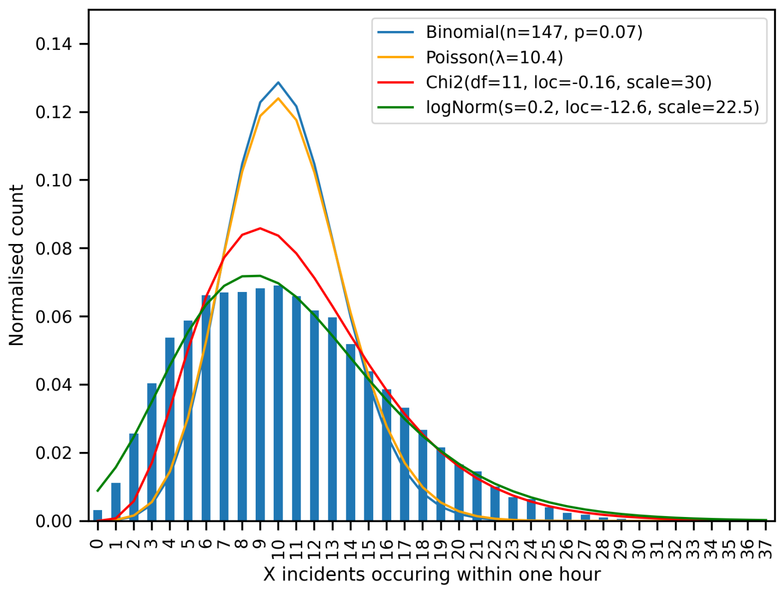

Figure 4.8: Number of incidents per hour, normalized by sum

idea of its connectivity. The degree of the nodes refers to how many other nodes a given node has edges to. This is shown on Figure 4.9, and is restricted by $0 \leq \deg(V) \leq 8$.

Most nodes have a degree of 1 or more, but two nodes have a degree of 7. 14% of the nodes have a degree of 0; these are not included in the graph but are shown here for illustration. A degree of 0 occurs with grid cells in the ocean or far into the wilderness.

Interestingly none of the nodes had a degree of 8 (fully connected). One could assume that this would be the case in the city center. However, this was not the case, most likely due to the spatial granularity (1x1km)

The degree can also be shown on a map for better interpretation (See Figure 4.10). The lighter the color of the cells means a higher degree and higher connectivity in the grid-cell graph.

4.3 Open route service

Figure 4.11 shows all the base stations' isochrones. This map is available online at https://joneliasmoen.github.io/, another version with time in seconds for 15, 30, 45, and 60 minutes is available at https://joneliasmoen.github.io/coverage.html.

Base stations farther from the city center have bigger isochrones due to the road

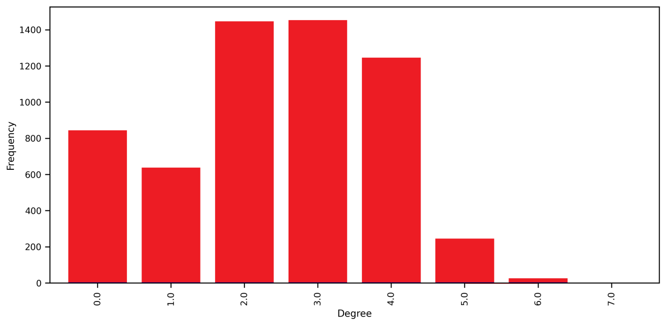

Figure 4.9: Hisogram of road network graph H degree.

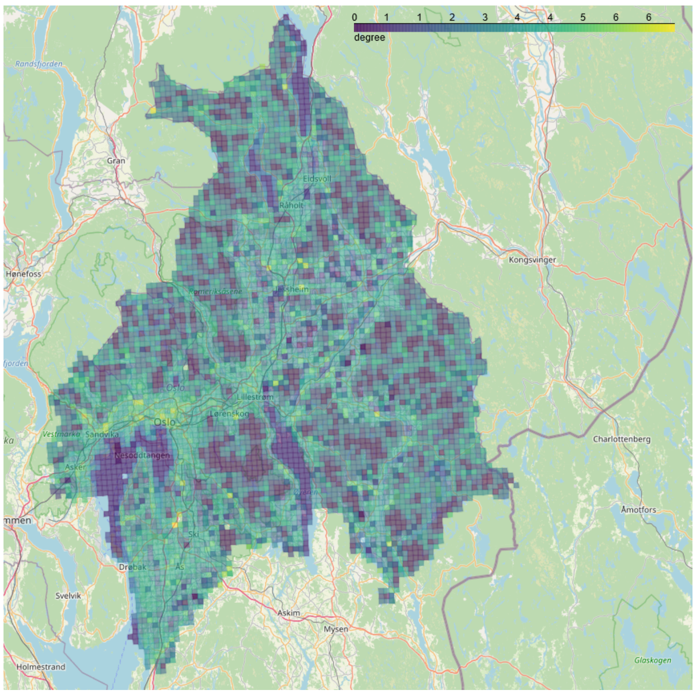

Figure 4.10: Road network graph degree. Degree referes to the number of grids a certain cell can travel to by car

network (Figure 4.12). In the city center, there are many intersections that slow down travel time. As a consequence, a high amount of base stations are located in the city center, where also many incidents occur.

It is worth mentioning that these isochrones do not represent the actual coverage of ambulances. It is instead an estimate of their coverage based on travel time by car under usual driving conditions. Therefore, ambulance coverage can be assumed to be a constant bigger and vary over time.

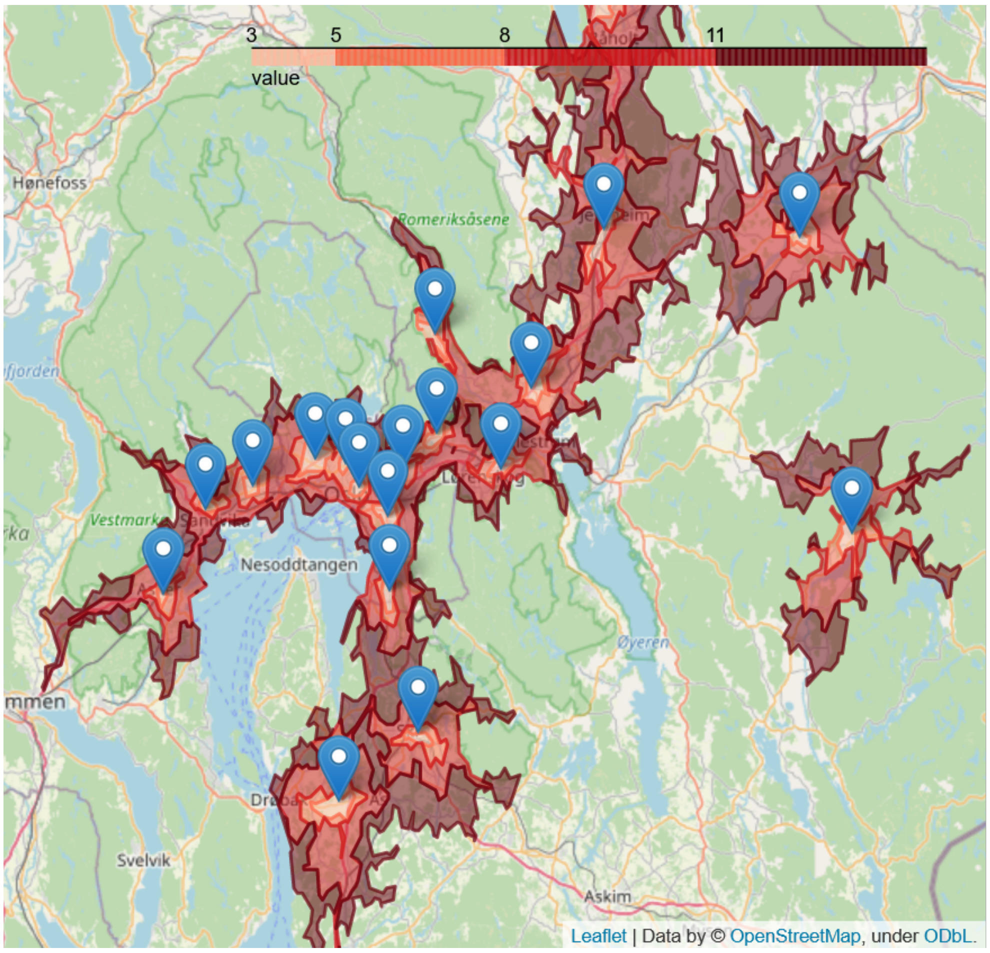

Figure 4.11: All isochrones from base stations in Oslo and Akerhus

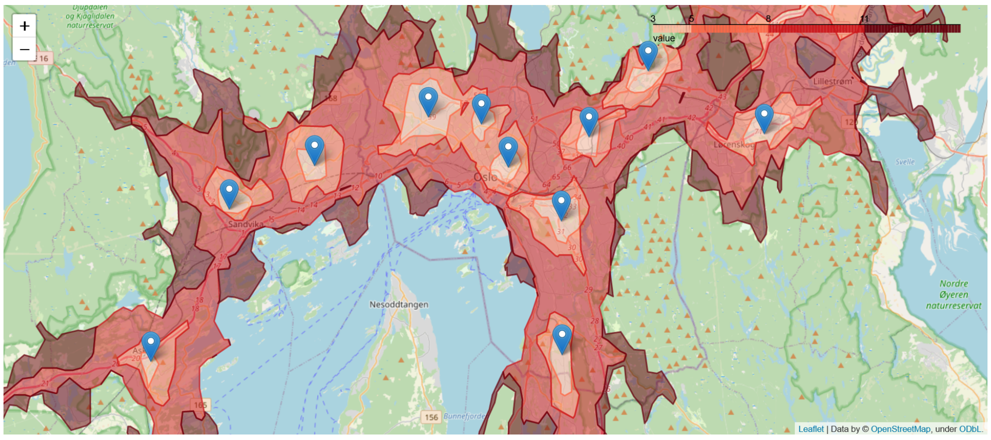
3
5
8
11
value
Lillestrøm
Sandvika
Oslo
Nesoddtangen
Nordre Øyeren naturreservat
Leaflet | Data by © OpenStreetMap, under ODbL.

Figure 4.12: Isochrones in City center Oslo, Norway.

Chapter 5

Method

This chapter outlines the method for the simulation model, RL model, and its evaluation. The next chapter outlines each experiment where most of the methodology is explained here.

5.1 Simulator model

A simulator model was built based on knowledge gained from the incident dataset and data preparation of the OSM dataset. This simulator model was made to mimic a most general case of the EMS dispatch setup and counteracts the fact that ambulance location is lacking from the incident dataset.

Further, wholly relying on the incident dataset for incident generation can induce overfitting on the RL model. Therefore a combination of simulated incidents and incidents from the incident dataset is used for training and evaluation.

The incident dataset $\mathcal{D}$ has records $\mathcal{D} = (r_0, r_{\dots}, r_n)$. Where each record contains $(i_t, i_l, i_p, \text{t-d}, \text{i-d}, \text{h-l}, t_n)$ tuples. This record denotes that an incident i occurs with priority i_p at grid-cell i_l at time i_t. The dataset is sorted by increasing i_t. Furthermore, t-d, i-d, and h-l denote this incident's different processing times. Lastly, t_n is the time until the next incident occurs, which the RL model does not know.

The records from the dataset $\mathcal{D}$ are split into a time series train-evaluation split $\mathcal{D}_t, \mathcal{D}_e$. This is performed such that $\frac{|\mathcal{D}_e|}{3} \approx |\mathcal{D}_t|$, and $(\forall_{r_t \in \mathcal{D}_t} i_t \in r_t) < (\forall_{r_e \in \mathcal{D}_e} i_t \in r_e)$. Which essentially means that the size of the evaluation set is one third of the training set, and comes after the training set in time i_t. Which type of data used for training and evaluation is outlined in each experiment in the next chapter.

It is worth noting that the simulator does not update at fixed time intervals. This is different from a Discrete event time simulation (DES), which updates every time an event occurs. Instead the simulator updates every time an incident occurs, and uses the amount of time since the last incident occurred δt to update.

The simulator model considers incident priority, processing time, ambulance location/routing, synthetic/CAD data, incident queue, and day/night shifts for the allocation. It also keeps track of the state and action spaces for the RL model. Furthermore, the simulation methodology applies Numpy DateTime functionality, recursion, and the created grid-cell graph $H(V,E)$.

5.1.1 Synthetic Incident generation

The synthetic generator assumes incidents arrive with a global constant $\lambda = 0.21$ in the Exponential probability distribution (Equation 5.1). This is then used as a discrete PMF to draw $\delta t = X$, which is used to derive the location of the incident[1] i_l, and ambulance locations.

$$P(X = x) = \lambda e^{-\lambda x} \tag{5.1}$$

To get the location of the incident, a homogeneous Poisson process is assumed for all the locations. Each of these processes follows the Exponential distribution with parameters $\lambda_L = (\lambda_{l=0}, \lambda_{l=...}, \lambda_{l=n})$. Note that these parameters are used to derive the location of the incident once an incident has occurred (determined by the global $\lambda = 0.21$).

λ_L is derived from the dataset $\mathcal{D}$. This is done by calculating the average incident interval (Mean time between failiure (MTBF)) $\overline{t_n}$, for each location l (Equation 5.2[2]). Then Equation 5.3 is used to derive the λ_l parameter.

$$\overline{t_n}|l = \frac{\sum_{r\in\mathcal{D}|i_l=l} t_n}{\sum_{r\in\mathcal{D}|i_l=l} 1} \tag{5.2}$$

$$\lambda_l = \frac{1}{\overline{t_n}|l} \tag{5.3}$$

Based on these parameters, *genIncident* (Algorithm 2) is used to derive the location and priority i_p of the incident after δt has been drawn. Based on the homogeneous Poisson process assumption, the probability that an incident occurs at a given location P follows the CDF of the Exponential distribution (line 2). The probabilities P smooths out as more incidents occur.

This method gives a simple case of incident generation since the rate parameters are static throughout the simulation. Which again is based on an average scenario. In a real-world scenario, the rate parameters change over time, and the λ_l's covaries with each other. Because of this reason, the simulation model is also made to simulate the incidents from the incident dataset itself. But changing the rate parameters during simulation to mimic a more realistic scenario is also possible.

[1]Uses same notation as from the incident dataset. But is here generated

[2]i_l from the dataset records

Algorithm 2 *genIncident*, generate synthetic incident

```
Require: $\delta t$ time since last incident occured, $\lambda_L = (\lambda_{l=0}, \lambda_{l=\dots}, \lambda_{l=n})$ derived
    lambda for each grid-cell location, $T = (t_{l=0}, t_{l=1}, t_{l=\dots}, t_{l=n})$ time since in-
    cident has happened at each grid-cell location (initialized to zero).
Ensure: Generated location and priority of incident.
1: $T = T + \delta t$                          ▷ Update times since last incident occurred
2: $P = 1 - e^{\lambda_L T}$                   ▷ Update probabilities
3: $P = \frac{P}{\sum P}$                      ▷ Normalize
4: $i_l$ =Random($P$, 1)                       ▷ Draw one random location, from discrete PMF
5: $T_{l=i_l} = 0$                             ▷ Set time since last incident occurred at this location to zero
6: $i_p$ = Random(($A = 0.57, H = 0.43$), 1)   ▷ Draw incident priority
7: return $T, i_l, i_p$
```

5.1.2 Ambulance simulation

This section outlines how the ambulances are simulated by the simulation model. Keep in mind that the main flow of the simulation model is:

1. A state is provided to the RL model
2. The RL model performs an action
3. The simulation model updates based on the action performed.

Ambulance location

The simulation model keeps track of all ambulance locations, even ambulances en route to incidents, hospitals, or base stations. The state representation, however, only captures ambulances available to be dispatched, but the position of all ambulances is updated at each iteration. The ambulances move around in the grid-cell graph H.

Once an ambulance is dispatched (action performed), it automatically drives the shortest route to the incident, hospital, and base station, independent of the possible dispatch actions. It is assumed that ambulances are not patrolling around the city in anticipation of incidents. The shortest route is retrieved by a combination of caching and A* search on the grid-cell graph H.

Which ambulances are considered available to be dispatched is modifiable. A list of possible scenarios is outlined below. Due to a lack of information on the capacity of the ambulances, Scenario 1 and 2 is assumed.

Scenario 1. Ambulances **at base stations** are considered available.
Scenario 2. Ambulances **driving to base stations** are considered available.
Scenario 3. Ambulances **driving to incidents** are considered available.
Scenario 4. Ambulances **driving from incidents to hospitals** are considered available.

Assuming an ambulance has a capacity of more than one patient.

Scenario 5. **All ambulances** are considered available.

An ambulance's route starts at a grid location A and ends at a grid location B but goes through several other grid locations on the way. From the route $L = (A, l_{\dots}, B)$, the travel time between these cells are retrieved from the grid-cell graph $H(V, E)$. This denoted by $T = (E(L_i, L_{i+1}))_{i<|R|-1}$, and is used by Equation 5.4 to calculate the cumulative sum travel time. The last value of T_c is the total time needed to complete the route.

$$T_c = (\sum_{i=0}^{i=k} T_i)_{k \leq |T|} \tag{5.4}$$

Once an incident occurs, every ambulance is given the time since it was last updated (δt). From this, the accumulated time used on a route is kept track of ($\hat{t}$), which is updated by $\hat{t}_{\text{new}} = \hat{t}_{\text{old}} + \delta t$. The location of the ambulance L_c is then updated by Equation 5.5, which is also illustrated in Figure 5.1.

$$L_c = \min_L \lfloor T_c - \hat{t} \rfloor \tag{5.5}$$

In other words, the ambulance is at the grid cell with cumulative travel time most equal to but not greater than $\hat{t}$. This is the floor operation in Equation 5.5. This is because $\hat{t}$ needs to exceed T_c to be considered at that location since the cumulative sum travel time between locations is not necessarily linear.

Each ambulance takes on a state value. This represents whether the ambulance is at its base station, traveling to a incident, at the incident location, traveling to the hospital, at the hospital, or traveling to its base station.

If $\hat{t} > t_n$, then $\hat{t} - t_n$ is added to the next route. This is recursively performed on the routes (status change) for the ambulance until a route is not yet completed or at the base station. In this way, the simulator model does not need to be updated at regular intervals but only when incidents occur. section A.4 provides a detailed explanation of how the ambulances are updated.

Allocation / Reallocation

The base station of each ambulance is also changeable; this makes it possible to reallocate ambulances to a new base station during simulation. This thesis does not consider continuously changing the allocation during simulation (ambulance reallocation problem). However, the allocation is modified for day and night shifts during the simulation. The population-proportionate allocations derived from Schjølberg and Bekkevold [2] is assumed since it was shown to work best in a long-term perspective.

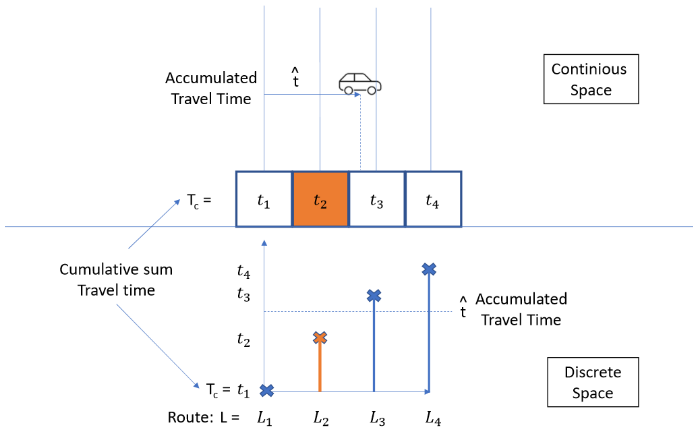

Figure 5.1: This shows how the location of an ambulance (located at the top) is derived, with the derived location marked in orange. The bottom graph (Discrete space) shows locations L on the X-axis and the cumulative sum travel time T_c on the Y-axis. The Y.-axis has marked $\hat{t}$. The top graph (continuous) shows the same but also with the cumulative sum travel time on the X-axis.

This shift reallocation is performed when training and evaluating the incident dataset. The day and night shifts have a different number of available ambulances since there are fewer incidents at night. The allocation is changed by keeping track of the current time of day in the dataset and switching the allocation at 08:00 and 20:00. This switch is done dynamically during simulation.

The switch from day to night allocation is the most complicated. Going through all the base stations, the number of ambulances to make unavailable is kept track of n_a. Let n_b be the number of ambulances at the base station. Then all the n_b ambulances are set to be unavailable. However, $|n_a - n_b|$ random ambulances are set to be unavailable once they return to their base station. Since it might not be the case that $n_a \geq n_b$, due to many ambulances being en route.

When switching to day time allocation all ambulances are re-set to be available.

Ambulance processing time

Between these status changes of the ambulances, there are assumed to be processing time. Recall the distribution found for these processing times in Figure 4.3.

When using the synthetic generator, the processing times for a given incident is drawn from these distribution. Which distribution to use depends on the priority of the incident i_p. When the incident dataset is used, these transition times are read from the dataset.

5.1.3 Incident Queue

When considering ambulance processing time, incidents can likely happen when ambulances are unavailable. Due to this, a *queue version* of the model is also built. In this version, incidents enter a queue if it occurs when there are no available ambulances. Then the model simulates forward in time until an ambulance is available.

Instead of simulating forward with a constant time step, the simulation uses a time step equal to the amount of time until a change in any of the statuses of the ambulances (t_a). In other words, it checks every ambulance and retrieves the time left until the current task is completed. From all these times, the minimum is used to simulate forward. A status change (task completion) means an ambulance has arrived at some location, which can be a base station, incident, or hospital.

During this waiting-for-ambulance simulation, an incident may occur. This is handled by looking forward and finding the time until the next incident occurs (t_p). While simulating forward, this value is subtracted by the time step t_a until it becomes $t_p \leq 0$. Once this happens, an incident with a timestamp $t - |t_p|$ is created and added to the incident queue Q. This process is repeated until an ambulance is available. *simNoAvail* (Algorithm 3) shows how this process works. section A.5 provides a more elaborate textual explanation.

The *timeUntilNextIncident* function returns t_n for the current record in the incident dataset. When using the synthetic generator, this is drawn from Equation 5.1. *newIncident* creates an incident based on the current record in the incident dataset and moves to the next record (Algorithm 2 for synthetic).

Algorithm 3 *simNoAvail*, simulation function with no available ambulances

Require: Incident Queue Q, Simulation time t, Time until next incident t_p, Ambulances $a \in A$, where a has features; a_t time until route finished, a_b is busy (boolean)

Ensure: State with available ambulance. While considering incidents

```
 1: while $\forall_{a \in A} a_b$ do                          ▷ All ambulances busy
 2:     $t_a = \min((a_t)_{a \in A})$
 3:     $t = t + t_a$                                  ▷ Move simulation time forward
 4:     $t_p = t_p - t_a$
 5:     while $t_p \leq 0$ do                            ▷ An incident has happened
 6:         $i = newIncident(t + t_p)$          ▷ New incident with timestamp $t + t_p$
 7:         Q.insert(i)                               ▷ Add new incident to Q
 8:         $t_p = t_p + timeUntilNextIncident()$
 9:     end while
10:     $A = updateAmbulances(\delta t = t_a)$                ▷ This is section 5.1.2
11: end while
```

Once an ambulance is available, multiple incidents may be in the incident queue.

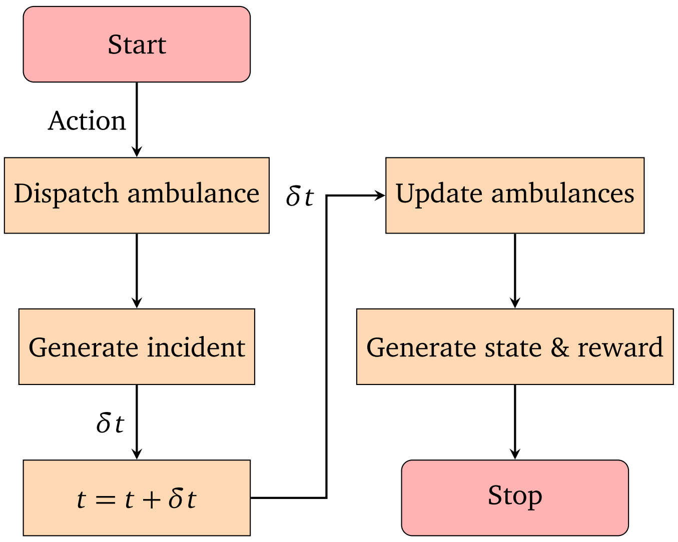

Figure 5.2: Shows the main flow of the nonQueue version of the simulator

Instead of considering a FIFO list of the incidents, all the incidents in the queue are a part of the action and state space provided. The RL model can decide which incident from Q and ambulance to dispatch rather than having to dispatch an ambulance to the first incident in Q (FIFO).

When there are multiple available ambulances and multiple incidents in the queue, the available ambulances reduce the queue down in size. With n available ambulances, $|Q|$ becomes $|Q| - n$.

How this process fits together with the simulation is shown in Figure 5.3. This differs from the regular flow of the simulator, which is shown in Figure 5.2

5.1.4 Small world setup

The model was set up to reduce to a small set of locations. In this way, a small subset of Oslo and Akershus can be simulated instead of its entirety. The subset of locations is defined by a circle on a map. A center and radius distance is defined, and locations within the circle are kept in the model. By default, a 5km radius from Oslo Central Station is assumed.

5.1.5 RL setup

Below state, actions, and rewards for the reinforcement learning algorithm are outlined. Figure 5.4 shows how the RL model interacts with the simulator, together with input and output.

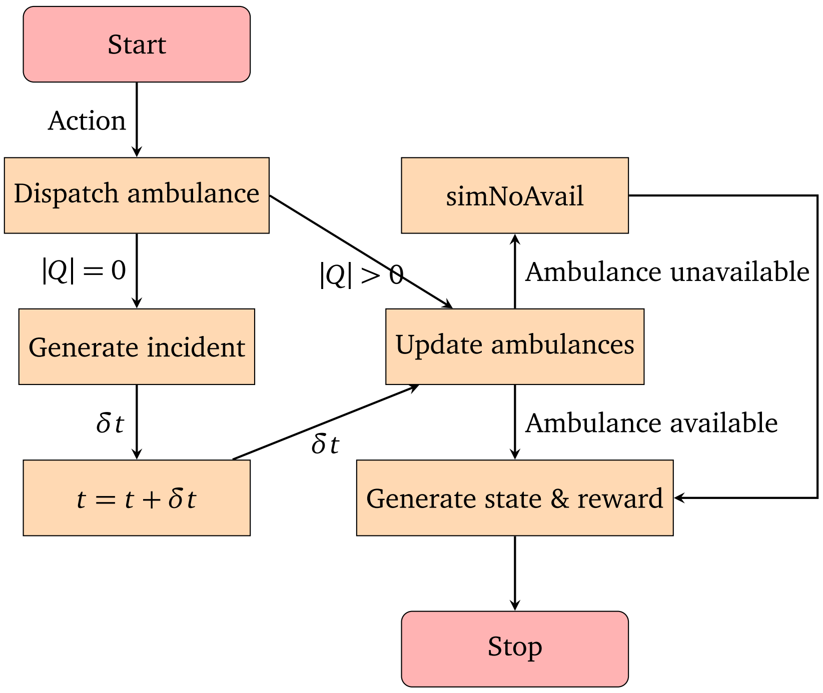

Figure 5.3: Shows how the queue version of the simulation model handles unavailable ambulances.

State

The state provided by the simulator is based on the location of the ambulances and incidents. Since the reinforcement learning model used supports multiple state inputs, the model provides one list of the ambulance locations A and one list for the incident locations I.

These lists are equal in length to the number of locations N simulated in the model ($|A| = |I| = N$). The ambulance list A is defined by $A = (f_{ak})_{k \leq N}$, where the value of f_{ak} is the number of available ambulances located at location k. Similarly, The incident list I is defined by $I = (f_{ik})_{k \leq N}$, where the value of f_{ik} is the number of incidents located at location k. This makes $\sum I$ the current number of incidents and $\sum A$ the number of available ambulances. When multiple ambulances are located at a base station location k, then $f_{ak} > 1$. Further, there can be more than one incident in the incident list if the queue model is used ($\sum I > 1$).

Action

In the non-queue model, an action is a choice between available ambulances. In other words, the action space equals the number of simulated ambulances, but only available ambulances can be chosen in any given state.

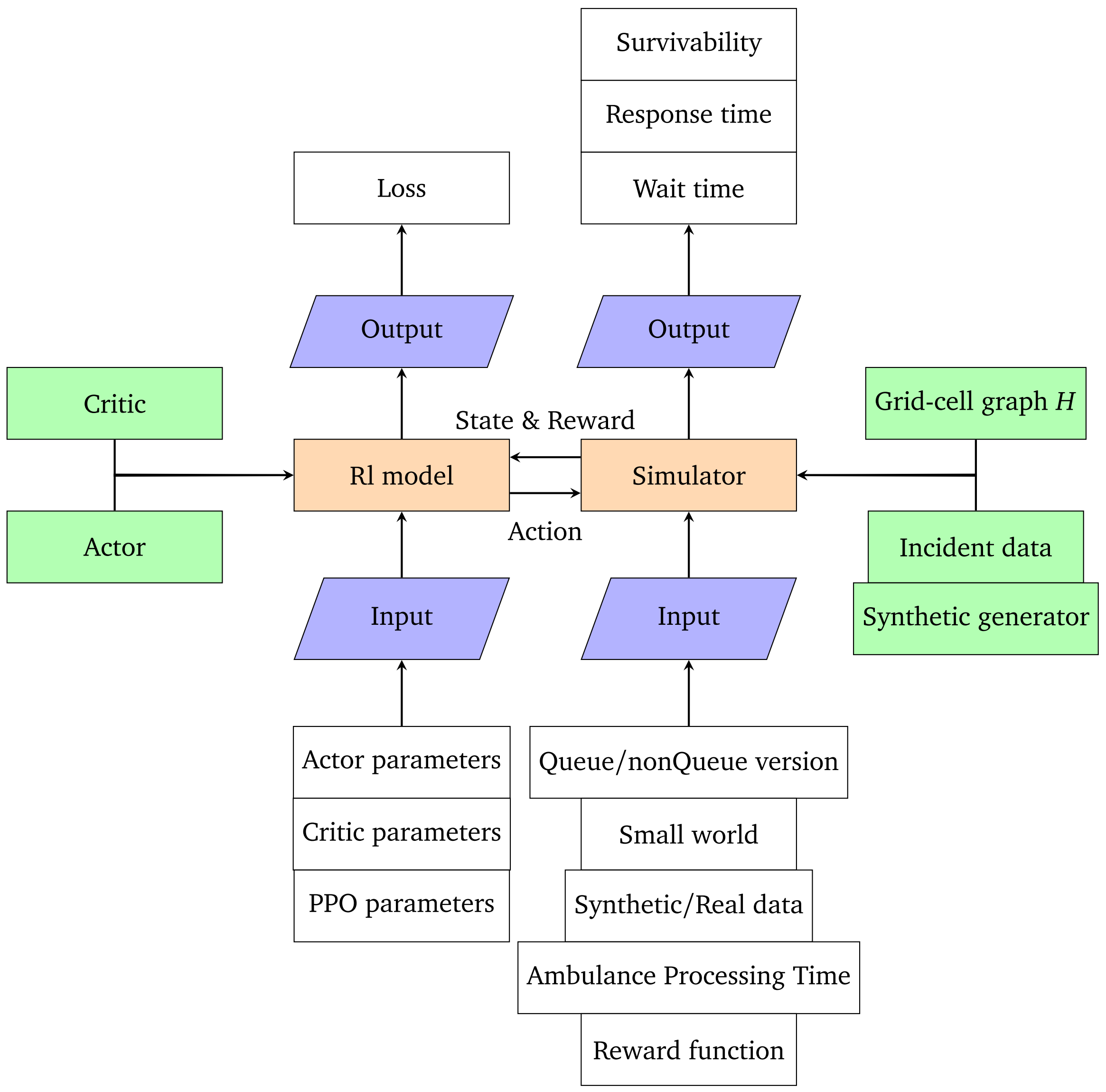

Figure 5.4: This shows how the RL model interacts with the simulator, together with its input/output. Lastly elements in green shows the components the RL and simulator comprises of.

In the queue model the action space equals the Cartesian product $A \times I$. In other words, both an available ambulance and an incident are chosen.

Reward

Inspired by Liu *et al.* [40], the reward is defined as the negative sum of the waiting time of the incidents (including those in Q). In other words, when an incident occurs, the waiting time increases until it is attended by an ambulance. In a given state s, the reward $R(s)$ is defined as the negative sum of these waiting times $W_t(i)$ (Equation 5.6).

$$R(s) = -\sum_i w_t(i) \tag{5.6}$$

Compared to using response time as a reward, this reward function is more intuitive for the model. When using response time as a reward, the effect an action has on the reward might come several states later in the simulation. When using waiting time, the reward monotonically decreases over the states after an action is performed.

The simulation model may simulate past the state where the wait times can be gathered. In other words, it is possible that an ambulance is dispatched to an incident, and the ambulance arrives at the hospital in the next state. Hence there is no wait time in that state for that incident. In this case, the response time for that incident is used instead of its wait time.

This reward function does not consider incident priority (inspired by Bandara *et al.* [21]). To achieve this, the makeshift survival function (Figure 4.2) and Equation 5.6 is combined into Equation 5.7. i_p is the priority of incident i, and f is the makeshift survival function.

$$R(s) = \prod_i f(w_t(i), i_p) \tag{5.7}$$

5.2 Reinforcement learning model

Since there are various available ambulances to dispatch in each state of the environment, a maskable PPO RL model is used. This means the model can only choose a feasible action in each state.

The PPO model maintains two MLP networks (As seen in Figure 2.4). This is both the value network (Critic) and the policy network (Actor). See section 2.5 for a description of how the PPO model work.

5.2.1 Hyperparameter search

There are many different hyperparameters to tune in the PPO model. These are hyperparameters such as the learning rate, discount factor, replay buffer size, batch size, epochs, clip range, and network architecture. Once specified in the experiments in the next chapter, these are tuned by Optuna[3].

[3]https://optuna.org/

5.2.2 Environment

The PPO model interacts with ten different instances of the simulator model in parallel. Each of these environments has different random number generator seeds, which affects the synthetic incident generator (Algorithm 2), and which ambulances are set unavailable (section 5.1.2). This does not affect anything else in the simulator. Using more than one simulator instance promotes more extensive exploration by the PPO model.

5.2.3 Evaluation

The trained reinforcement learning model (RL) is compared to three other dispatching agents. The first agent dispatches the ambulance with the lowest Haversine distance (Haversine), while the second agent according to the lowest Euclidean distance (Euclidean). Finally, the last agent dispatches a random available ambulance (Random). These policies are shown in the results in the next chapter, differentiated by the name in parentheses.

The Haversine distance might fit better for this application than the Euclidean distance. While the Euclidean distance calculates the straight line distance, the Haversine calculates the distance on a spherical object (such as the earth). Using the Euclidean distance to measure distance on the earth would be like using a ruler on a globe to measure the distance. While the Haversine, would be like using a soft tape measure. The difference is that the ruler (Euclidean) assumes the route is straight, while the soft tape (Haversine) assumes it can be curved. This distance difference is small for short distances (Such as a diameter of 10km in this case) but deviates more as the distance becomes larger (up to 100km). The Haversine distance is more accurate than the Euclidean since it considers the earth's curvature. Which might be clear in the experiments.

It is a possibility that the RL model dispatches the same ambulance as the Haversine policy. Hence the fraction of such actions performed is kept track of (Haversine fraction). This shows how different the trained policy is from the Haversine policy.

Chapter 6

Experiments and results

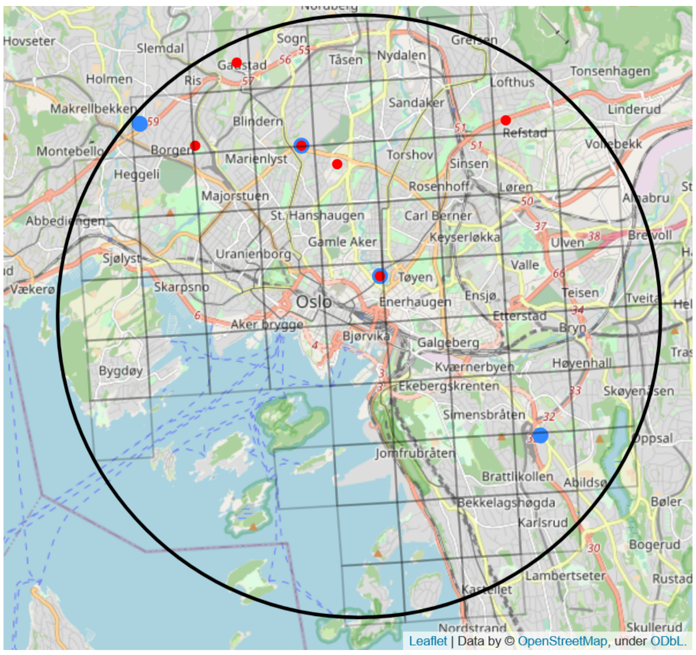

Figure 6.1: Shows considered small world (5km radius from Oslo Central station). The big black circle show the area considered, while the small black squares the grid-cells. Base stations are marked with blue circles, while hospitals in red.

This chapter outlines the experiments, their setup, results, and discussion. It is worth noting that the travel time is perfectly accurate according to real-world ambulances. However, it provides an estimate according to following the speed limit of the roads.

When referencing RL model, this is the same as referencing the PPO model in the previous chapter.

6.1 Experiment 1

In the first experiment, the model is trained and evaluated on synthetic data (subsection 5.1.1) in the area of Oslo city center (subsection 5.1.4, Figure 6.1). Ambulance processing times (section 5.1.2) are not considered for this experiment, and incidents are considered without priority.

The policy and value MLP network of the PPO model was setup with a architecture of 3 layers of 100 nodes with the Relu activation function (section 5.2). The learning rate was set to 0.001. Both the value and policy network shares a feature extractor, which in this case is just the concatenated states.

The wait time reward (Equation 5.6) is used, while the action space is a choice between available ambulance. The action space is masked, this makes it so that the RL model can't dispatch ambulances which are unavailable, and speeds up the training process.

Since the problem has an infinite time horizon (no explicit end state), the model collects a batch and trains on it after 1028 steps in the dispatching environment. The model is trained on 6 million incidents and achieves reasonable metrics during training.

6.1.1 Results and discussion

The result is compared to different agents described in subsection 5.2.3. The different agents are under the **Model** column in Table 6.1. As seen from Table 6.1, both the mean response time and mean reward (wait time) are better for the RL model than for any other agents. Remember that a larger mean reward is better since it is the negative of the wait time (Equation 5.6).

For better visualization, the zero response time and wait time values are ignored in the histograms and CDF plots (Figure 6.2), which may make the result deviate from the table result (Table 6.1). Zero response time means an incident happens at the same location as an available ambulance since this experiment does not consider ambulance processing time.

Both the Haversine and the Euclidean distance are partially random since there can be multiple ambulances having the lowest distance to the incident. For example, this can happen when a base station with multiple ambulances is the closest to an incident. A random ambulance among these is chosen.

The RL model has probably learned to dispatch the ambulance with the lowest road network travel time. Judging from the results from the Haversine Fraction (Table 6.1), the lowest road network travel time corresponds to the Haversine about 31% of the time.

The performance of the Euclidean agent changed wildly during the runs of the experiment. Sometimes it performs better than the random policy, while other

Model	Mean Response Time	Mean Reward	Haversine Fraction	Iterations
RL	12.99	-10.93	0.31	50001
Haversine	14.25	-16.63	1.00	50001
Euclidean	14.42	-16.56	0.80	50001
Random	15.11	-19.20	0.22	50001

Table 6.1: Shows results from **experiment 1**. Keep in mind that Mean reward (Wait time) should be as high as possible. Notice how the Haversine agent has slightly lower response time than euclidean.

times not. This is because the random policy can choose the ambulance with the shortest road network travel time, while the Euclidean agent sometimes will pick an ambulance with a high road network travel time. Because of this, the results depend on the number of incidents where the lowest Euclidean distance ambulance corresponds to the lowest road network travel time ambulance. The Haversine agent consistently performs better than the Euclidean agent, but has slightly worse **Mean reward** (-16.63 vs -16.56).

Figure 6.2 shows histograms and CDF for the different policies (See figure caption). The results indicate that the RL policy performs better than the other policies.

All the policies have roughly the same histogram distribution for low wait and response time (0-15m); however, the RL model has a lower tail for higher response time (15-40m). This means that a lower fraction of incidents has a higher wait and response time for RL than the other policies, which is also reflected in the CDF.

The **Wait time histogram** has a broader shape than the **Response time histogram**. This is because the wait time for the same incident can be counted into the wait time reward multiple times. For example, this happens when an ambulance spends several states before arriving at the incident location. Hence the wait time for the incident is counted in all those states. This means that how broad the wait time histogram is depends on how often incidents occur (a state is generated).

In consideration, the action space of the RL model is not optimal since it is rigid for the number of ambulances. There is little to no mapping between the ambulance location space provided and the action space. Choosing an ambulance corresponds to choosing a base station to dispatch an ambulance. Whether the PPO model can consider and dispatch ambulances en route to its base station is debatable (Scenario 2) since the action space does not indicate the ambulance location (or status).

6.2 Experiment 1.2

In the next experiment, the same assumptions are kept (as in Ex1, section 6.1), but the same model is evaluated on the test set of the incident dataset $\mathcal{D}_e$. This

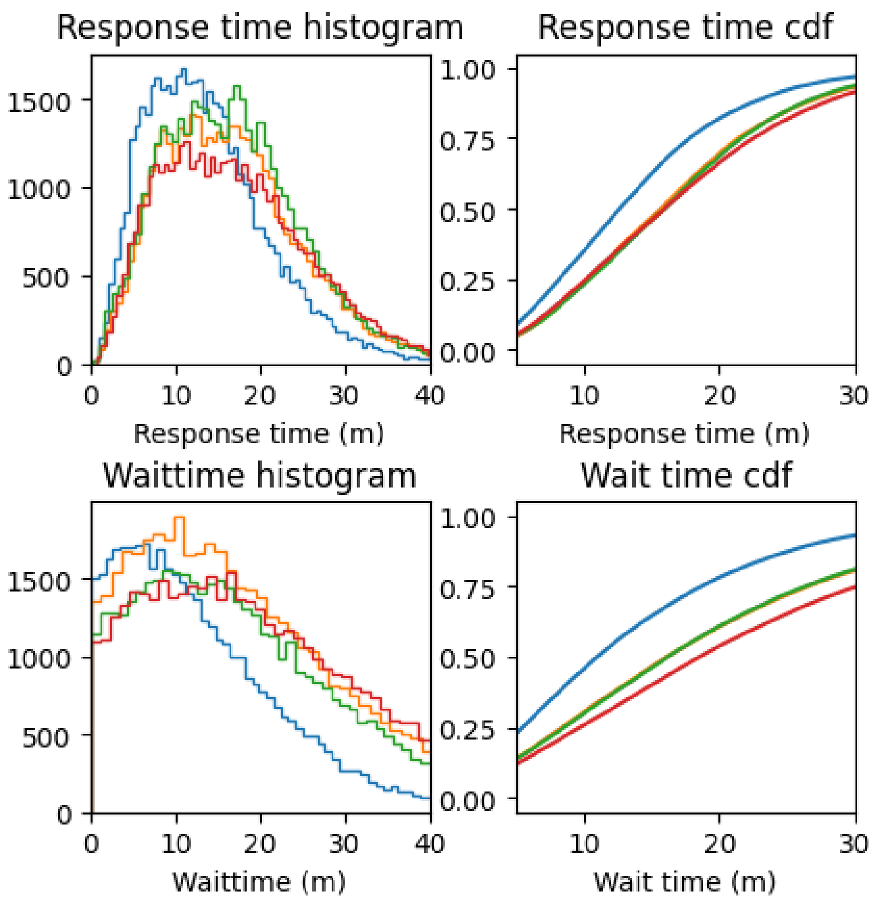

Figure 6.2: Shows results for the different policies evaluated (**ex 1**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time and waittime

will unveil whether the RL model works in a realistic scenario and performs better than the heuristics under fixed incident locations.

The main difference between the synthetic dataset and the test set is that the global λ and λ_l change over time. The ambulance allocation also changes depending on day and night shifts (section 5.1.2).

6.2.1 Results and discussion

The results are shown the same way as in the previous experiment (as in Ex1, section 6.1). Table 6.2 shows the same columns as previously, the same with Figure 6.3.

The **Mean reward** (wait time) for all policies is much lower for this experiment than in the previous one. This is most likely because incidents happening at night are also considered, where the time between incidents is much longer. This means there are few states where an ambulance is en route to an incident (wait time for that incident is reflected in the reward).

Comparing the **Mean response time** for the RL model and the Haversine policy, the difference is not that large (16.78 vs. 17.29m in Table 6.2). This differs from the previous experiment, where the difference was more significant (12.99 vs. 14.25m in Table 6.1). This somewhat contradicts the fact that the Haversine fraction of the RL model is higher in Table 6.1 than in Table 6.2 (0.31 vs. 0.23). This can either be an artifact that the $\mathcal{D}_e$ has a much smaller sample size than in the previous experiment (29k vs. 50k iterations), or that the adaptability from the synthetic data to $\mathcal{D}_e$ is sub-optimal.

If the last case is valid, then the actions performed by the RL, which was not equal to a Haversine action in the previous experiment, have a higher impact on the response time than in this experiment.

Overall the RL agent outperforms the other agents in both experiments. The difference between Euclidean and Haversine is slim, but generally, the Haversine performs best. Based on the CDF's from Figure 6.3 and Figure 6.2, these policies appear highly overlapping.

Model	Mean Response Time	Mean Reward	Haversine Fraction	Iterations
RL	16.78	-0.67	0.23	29052
Haversine	17.29	-1.69	1.00	29052
Euclidean	17.36	-1.69	0.84	29052
Random	18.18	-2.05	0.22	29052

Table 6.2: Shows results from **experiment 1.2**. Keep in mind that Mean reward (Wait time) should be as high as possible.

6.3 Experiment 2

Based on knowledge from the previous experiments, the following experiment takes the complexity a bit further.

The RL model is trained on $\mathcal{D}_t$ and evaluated on $\mathcal{D}_e$. The training on $\mathcal{D}_t$ is performed until reasonable metrics have been achieved. Meaning that the records in the dataset are looped through until the model is stopped manually. Once the end of the dataset is reached, the simulation and RL model reset and start from the beginning of the dataset.

The ambulance processing time (section 5.1.2) read from $\mathcal{D}_e$ is considered. This will make the response time higher than in the previous experiments.

The model uses the survivability reward (Equation 5.7). This also means that

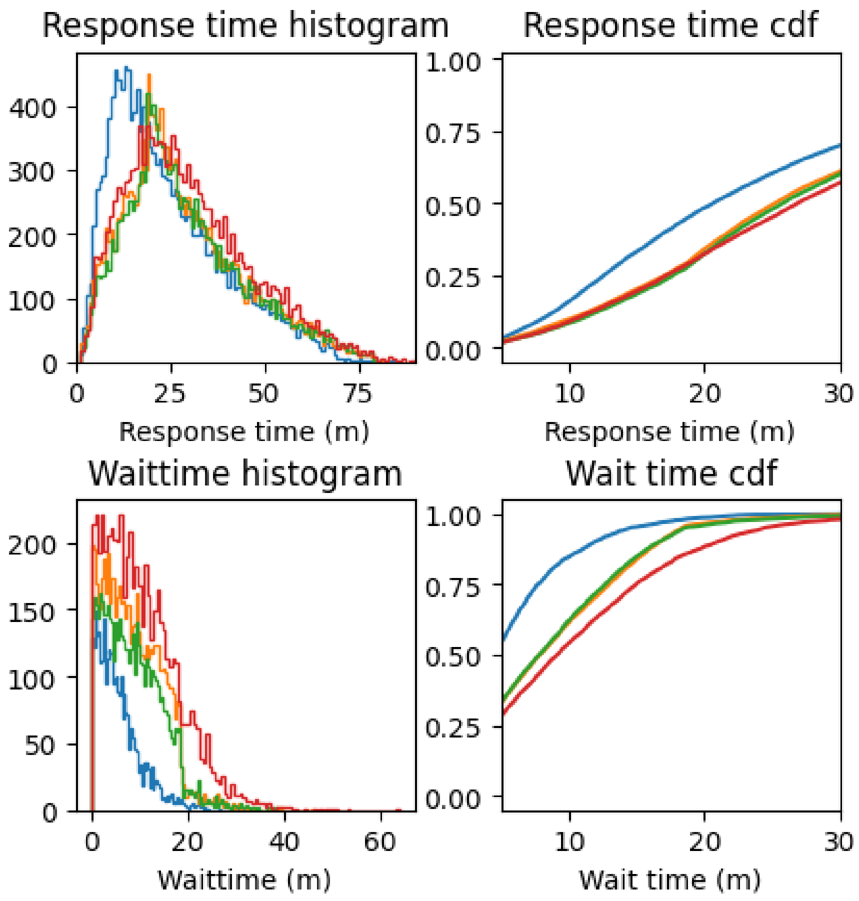

Figure 6.3: Shows results for the different policies evaluated (**ex 1.2**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time and waittime

incident priority i_p is considered for this experiment.

The queue version of the simulation model is used. This also means that the action space is the Cartesian product of the state lists A and I (section 5.1.5). The state representation is similar to the previous experiments (section 5.1.5); however, an incident survivability list $I_s = (f_{sk})_{k \leq N}$ is also considered. This list is similar to I, but the value f_{sk} is equal to Equation 5.7 for incidents at location k.

The same small world setup as in the first experiment is assumed (section 6.1).

Lastly Optuna is used for hyperparameter search (subsection 5.2.1). In the parameters found, the learning rate was set extremely low $1.9x10^{-5}$ and the replay buffer is large with 11k steps in the environment. The MLP networks has 2 layers with a size of 300 and 400 nodes respectively.

6.3.1 Results and discussion

Even though the model is not directly trained on response time, Figure 6.5 shows how the mean response time (Y-axis) reduces during training of the model. Mean response time is calculated by the latest 1000 response time per training step (11k steps in the simulator). Not every step in the simulator produces response time. The model was trained on $\approx$ 70 million incidents, and ran over the course of four days. As seen from the figure the response time is reduced by $\approx$ 5m.

Figure 6.4 has a similar setup than in the previous experiments. However the inverse CDF is shown for survivability, since it should be as high as possible.

The Haversine and Euclidean agents have no consideration of the priority of the incidents. It simply dispatches the ambulance to the incident with lowest distance.

Unexpectedly in Table 6.3, the Random agent performs better than the Haversine and the Euclidean agent. Even though the **Mean response time** is lower, the response time CDF and histogram on Figure 6.4 the Haversine agent performs better. Judging from the histogram, the Haversine agent has slightly more lower reponse times (<25m) than the Random and Euclidean agent. The high **Mean response time** and low **Reward** for these agents is probably because they have no consideration for the priority or for the incident queue.

The RL model seems to have not overfitted on $\mathcal{D}_t$. There is, however, significant doubt that this RL model considers the future incident distribution. Since the state space provided provides little information about the history of the incident distribution. If this was the case, the synthetic data generator should be used in combination with $\mathcal{D}_t$ to reduce overfitting RL model assumed future incident distribution.

Model	Mean Response Time	Mean Reward	Haversine Fraction	Iterations
RL	23.31	0.57	0.19	29052
Haversine	28.88	0.42	1.00	29052
Euclidean	29.13	0.41	0.74	29052
Random	26.42	0.46	0.23	29052

Table 6.3: Shows results from **experiment 2**. Keep in mind that Mean reward (Survivability) should be as high as possible.

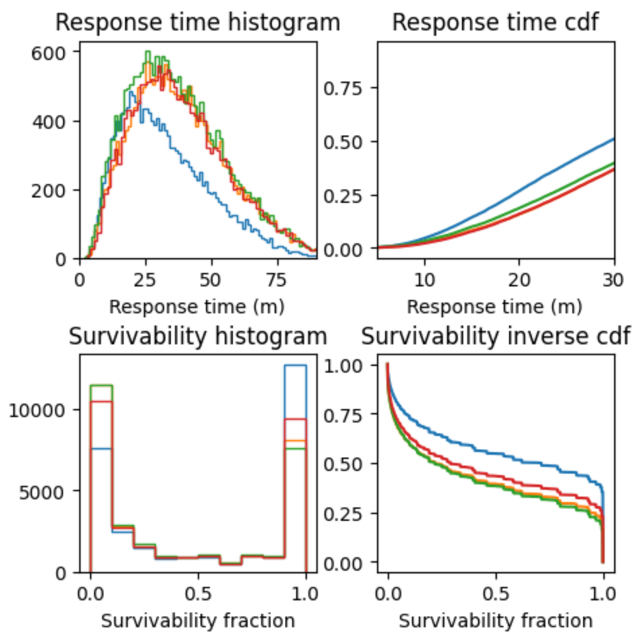

Figure 6.4: Shows results for the different policies evaluated (**ex 2**): RL (Blue), Haversine (Green), Euclidean (Orange) and Random (Red). Shows both histogram and CDF for response time. Inverse CDF is shown for survivability

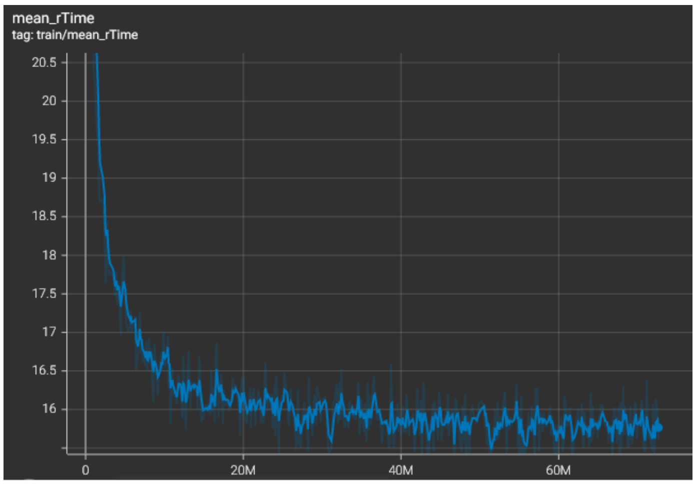

Figure 6.5: Shows the reduction in mean response time (Y-axis) during training of the RL model in **experiment 2**. X-axis shows the total number of incidents trained on. (Screenshot from TensorBoard).

Chapter 7

Conclusion and Future work

This chapter concludes this thesis and provides a description of possible future work.

7.1 Conclusion

This thesis has explored the usage of RL and PPO to the ambulance dispatching problem as a potential decision support tool for the EMS. The main contribution is that this implementation considers incident priority, incident queue, day/night shifts, usage of PPO, an extensive data analysis, and reproducible OSM travel time estimation. Furthermore, a simulation model written in Python is implemented, which has vast Machine Learning support. Finally, a simple method for synthetic incident data generation is also provided.

A literature overview is outlined with a table of the most recent literature on the usage of RL to the ambulance dispatching problem. Finally, a discussion of these implementations and recent trends are outlined.

This thesis has also shown how OSM can be preprocessed to be used for scientific purposes. Extensive work went into the preprocessing to make the data tractable.

Furthermore, extensive work went into building a simulator for the ambulance dispatching problem. This simulator considers many aspects of the ambulance dispatching problem, such as incident priority, incident queue, ambulance shifts, small world setup, Synthetic/Real incident data, and ambulance processing time. This simulator uses a time step equal to the incident interval time, which can speed up the simulation. Furthermore, it is implemented in Python, making it more suitable for Machine Learning. Extensive data analysis is performed with external links for created maps (section A.6). This gives many valuable insights into the problem of ambulance dispatching and the concept of coverage. Overall, the PPO model consistently outperforms the other heuristic agents evaluated throughout the experiments. Generally, the Haversine distance mostly outperformed the Euclidean distance when considering **Mean response time**; however, this difference is extremely slim (9.8s saved on avg). This shows that RL and road network travel time have great potential in reducing response time and increasing survivability. However, more work is needed for RL to be implemented as a decision support tool for the EMS (Future work).

The RL model did not overfit on the training dataset, however, a more complex model might do so. Therefore, a combination of synthetic and real data is recommended during training.

7.2 Future work

Potential future solutions and extensions to this implementation are outlined here. This thesis has provided a good foundation for future theses on the ambulance dispatching problem.

7.2.1 Reinforcement learning (RL)

This section outlines how the different aspects of the RL model can be improved.

Curse of dimensionality

The state and action space is vast in the ambulance dispatching problem. There are several possibilities to reduce the dimensionality. Currently, the size of the MLP networks scale poorly to the number of locations. The networks need two or more layers of > 100 nodes for 69 locations.

The feasible action space is dynamic. Since each state has varying available ambulances to dispatch (action space), this is a challenge for RL since the model has to choose a feasible action in every state, which can be very sparse compared to the action space as a whole. This was solved by action masking in this implementation, but it might not be the best approach.

This problem can be solved by making the RL model choose a specific action among a list of possible actions. These actions can then include estimated lost coverage, travel time, and work imbalance. Such a model might also work better with changing dynamics such as traffic congestion. Liu *et al.* [41] used such a setup for the ride-hailing problem.

Generally, the inclusion of heuristics in the state space of the RL model can help with the large dimensionality and intractability. This can also be combined with MCTS as utilized in Mukhopadhyay *et al.* [7].

Using sophisticated deep learning techniques to reduce the state space dimensionality is also possible—for example, convolutional neural networks, autoencoders, and embeddings.

Allocation / Reallocation

In this implementation, the allocation follows the population-proportionate allocation from Schjølberg and Bekkevold [2].

By making the RL model select both an ambulance, incident, and base station (Cartesian product), the ambulance reallocation problem can also be incorporated. In this case, the ambulance selected is reassigned to the base station selected by the RL model. However, this needs to also consider which ambulances to set as unavailable for the switch to the night shift.

The allocation implementation by Schjølberg and Bekkevold [2] could also somehow be used to optimize the allocation in real-time such that the allocation is dynamically changing during simulation. There also exist many other ambulance reallocation models which can be explored. For example, Nasrollahzadeh *et al.* [6] provides an excellent way to incorporate ambulance reallocation and dispatching.

Coverage

As of now, the coverage of the ambulances is not considered. By using the *egoGraph* method in the NetworkX package on the grid-cell graph H, the coverage of the ambulances can be estimated (or OpenRouteService). This can then be incorporated into the state and reward for the RL model.

POMDP consideration

As discussed in section 2.8.1, the ambulance dispatching problem might be a partially observable Markov decision process.

To counter this fact, it is possible to include information about the historic incident distribution into the state space. For example, maintaining a list containing the Poisson rate's Exponential moving average (EMA) per location. This can also be used in combination with the synthetic incident data generation technique (subsection 5.1.1) to make an expected incident probability distribution. Combining this with the coverage estimation can provide a reasonable estimate of the preparedness of the ambulances.

Another option can be to use the previous work on forecasting by Hermansen [13] and Van De Weijer and Owren [12] in combination with the RL model.

Another option is to use Recurrent PPO, which is shown to be a good baseline for POMDP[50]. However, this also needs to consider the sparse feasible action space of the ambulance dispatching problem.

RL model

A disadvantage of using PPO is that it is hugely sample inefficient. This is common among online RL techniques where the replay buffer is disregarded after it is used. Some of the models trained in this thesis were trained over several days and on several million incidents.

Offline techniques might be more suitable for this problem. Using Experience Replay as in Liu *et al.* [40] can be enough to assure stable learning.

7.2.2 Simulation

The simulation model implemented works for a most basic ambulance dispatching scenario. However, extensions can be made to the simulation model to simulate more complex ambulance dispatching scenarios.

For example, not all incidents need to be transported to the hospital. Incidents can be simulated to take on a value that defines the underlying urgency. This can then be used as a probability that the incident needs to travel to the hospital once an ambulance arrives. The processing time distributions found in section 4.1.1 can inform a possible probability distribution of this probability.

Using these underlying urgency values can also inform the possibility of an ambulance attending other incidents while traveling toward the hospital.

As discussed in chapter 1, making it possible for the RL model to reassign ambulances that are en route to an incident can be considered. This can possibly reduce response time and increase survivability.

Since the EMS is bound to dispatch some ambulance once an incident occurs, it might be unrealistic not to dispatch an ambulance. As discussed by Nasrollahzadeh *et al.* [6], the best ambulance to dispatch might be unavailable but very soon be available. The simulator can be adapted to work continuously so that this ambulance can also be dispatched. In other words, dispatching two ambulances to one incident (one immediately and one later).

Furthermore, the consideration of more realistic travel time estimation can be explored. In this implementation, the travel time is based on the speed limits; but it is possible to train a LSTM model based on data from Mapbox. This can then form a basis to estimate travel times of different EMS vehicles and incident priority.

Synthetic incident generator

The λ_l used for the synthetic incident data generation is static. However, in a realistic scenario, these covary with each other over time.

It is possible to build a more sophisticated generator. This can be done by simulating the covariance matrix of the λ_l. Other techniques like Gaussian mixture models, Variational autoencoder (VAE), Generalized Linear Model (GLM)[7] and Wasserstein generative adversarial neural network (WGAN) can also be used. On the other side simple technique is to sample incidents randomly from the incident dataset.

7.2.3 Data analysis

Since the incident dataset has the ID of the ambulance dispatched, it is possible to quantify work imbalance. This can be done by looking at the amount of time since the dispatched ambulance was last utilized in the dataset.

Furthermore, this might unveil unconsidered aspects such as lunch breaks and assumed ambulance starting position. It can also be used to see if work imbalance directly impacts the performance metrics of the EMS system.

All of this can be used to find good ways to quantify the work imbalance, which can be incorporated into the reward of the RL model.

7.2.4 Other optimization techniques

Ant Colony Optimization (ACO) which has been shown to work for Dynamic Vehicle Routing Problem (DVRP)[51] and Mutli Depot Vehicle Routing Problem (MDVRP)[52] problems. As discussed in subsection 1.3.1, these problems have a close relationship to the ambulance dispatching problem. Such an implementation can also be combined with the heuristics found by Bandara *et al.* [21].

ACO might be more suitable for the ambulance dispatching problem since it can be more adaptable than RL for changing traffic conditions.

Such an implementation can also consider routing the ambulances through the roads to maximize coverage along its route. While at the same time balancing response time and incident priority.

7.2.5 Limitations

There is, however, a significant limitation on the availability of data. This puts a limit on how realistic the simulation can be. For example, there is little indication of how often an ambulance is reassigned to another incident, carries multiple patients to the hospital, or the starting location of the ambulances.

Bibliography

[1] V. Schmid, 'Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming,' *Eur. J. Oper. Res.*, vol. 219, no. 3, pp. 611–621, Jun. 2012, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2011.10.043.

[2] M. E. Schjølberg and N. I. P. Bekkevold, 'Simulation and Optimization of Emergency Medical Services in Oslo and Akershus,' M.S. thesis, NTNU, 2022. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3019906.

[3] N. Torres, L. Trujillo, Y. Maldonado and C. Vera, 'Correction of the travel time estimation for ambulances of the red cross Tijuana using machine learning,' *Comput. Biol. Med.*, vol. 137, p. 104 798, Oct. 2021, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2021.104798.

[4] A. Mukhopadhyay, G. Pettet, S. Vazirizade, D. Lu, S. E. Said, A. Jaimes, H. Baroud, Y. Vorobeychik, M. Kochenderfer and A. Dubey, 'A Review of Incident Prediction, Resource Allocation, and Dispatch Models for Emergency Management,' *arXiv*, Jun. 2020. DOI: 10.48550/arXiv.2006.04200. eprint: 2006.04200.

[5] Wouter Steenbeek, 'Geographic distance and Road distance are highly correlated (in the four largest municipalities of the Netherlands),' *Wouter Steenbeek*, Sep. 2020. [Online]. Available: https://www.woutersteenbeek.nl/post/geometric-road-distance-g4.

[6] A. A. Nasrollahzadeh, A. Khademi and M. E. Mayorga, 'Real-Time Ambulance Dispatching and Relocation,' *Manufacturing & Service Operations Management*, Apr. 2018. [Online]. Available: https://pubsonline.informs.org/doi/epdf/10.1287/msom.2017.0649.

[7] A. Mukhopadhyay, G. Pettet, C. Samal, A. Dubey and Y. Vorobeychik, 'An online decision-theoretic pipeline for responder dispatch,' in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, ACM, Apr. 2019. DOI: 10.1145/3302509.3311055. [Online]. Available: https://doi.org/10.1145%5C%2F3302509.3311055.

[8] A. H. Wong and T. J. Kwon, 'Advances in Regression Kriging-Based Methods for Estimating Statewide Winter Weather Collisions: An Empirical Investigation,' *Future Transportation*, vol. 1, no. 3, pp. 570–589, Oct. 2021, ISSN: 2673-7590. DOI: 10.3390/futuretransp1030030.

[9] *AMK - Tid fra AMK varsles til ambulansebil er på hendelsessted*, [Online; accessed 27. Jan. 2023], Jan. 2023. [Online]. Available: https://www.helsedirektoratet.no/statistikk/kvalitetsindikatorer/akuttmedisinske-tjenester-utenfor-sykehus/tid-fra-amk-varsles-til-ambulanse-er-pa-hendelsessted.

[10] M. Samdal, K. Thorsen, O. Græsli, M. Sandberg and M. Rehn, 'Dispatch accuracy of physician-staffed emergency medical services in trauma care in south-east Norway: a retrospective observational study,' *Scand. J. Trauma Resusc. Emerg. Med.*, vol. 29, no. 1, pp. 1–12, Dec. 2021, ISSN: 1757-7241. DOI: 10.1186/s13049-021-00982-3.

[11] K. Bohm and L. Kurland, 'The accuracy of medical dispatch - a systematic review,' *Scand. J. Trauma Resusc. Emerg. Med.*, vol. 26, no. 1, pp. 1–10, Dec. 2018, ISSN: 1757-7241. DOI: 10.1186/s13049-018-0528-8.

[12] E. Van De Weijer and O. A. Owren, 'Forecasting Ambulance Demand in Oslo and Akershus,' M.S. thesis, NTNU, 2022. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3030248.

[13] A. H. Hermansen, 'Machine Learning for Spatio-Temporal Forecasting of Ambulance Demand: A Norwegian Case Study,' M.S. thesis, NTNU, 2021. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2833870?locale-attribute=no.

[14] A. H. Hermansen and O. J. Mengshoel, 'Forecasting Ambulance Demand using Machine Learning: A Case Study from Oslo, Norway,' in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Dec. 2021, pp. 01–10. DOI: 10.1109/SSCI50451.2021.9659837.

[15] P. Abreu, D. Santos and A. Barbosa-Povoa, 'Data-driven forecasting for operational planning of emergency medical services,' *Socioecon. Plann. Sci.*, vol. 86, p. 101 492, Apr. 2023, ISSN: 0038-0121. DOI: 10.1016/j.seps.2022.101492.

[16] L. A. McLay and M. E. Mayorga, 'A model for optimally dispatching ambulances to emergency calls with classification errors in patient priorities,' *IIE Trans.*, vol. 45, no. 1, pp. 1–24, Jan. 2013, ISSN: 0740-817X. DOI: 10.1080/0740817X.2012.665200.

[17] A. Mukhopadhyay, Y. Vorobeychik, A. Dubey and G. Biswas, 'Prioritized Allocation of Emergency Responders based on a Continuous-Time Incident Prediction Model,' in *AAMAS '17: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, May 2017, pp. 168–177. DOI: 10.5555/3091125.3091154.

[18] C. J. Jagtenberg, S. Bhulai and R. D. van der Mei, 'Dynamic ambulance dispatching: is the closest-idle policy always optimal?' *Health Care Manag. Sci.*, vol. 20, no. 4, pp. 517–531, Dec. 2017, ISSN: 1386-9620. DOI: 10.1007/s10729-016-9368-0. eprint: 27206518.

[19] S. K. Keneally, M. J. Robbins and B. J. Lunday, 'A markov decision process model for the optimal dispatch of military medical evacuation assets,' *Health Care Manag. Sci.*, vol. 19, no. 2, pp. 111–129, Jun. 2016, ISSN: 1572-9389. DOI: 10.1007/s10729-014-9297-8.

[20] J. C. P. Roa, J. W. Escobar and C. A. M. Moreno, 'An online real-time matheuristic algorithm for dispatch and relocation of ambulances,' *International Journal of Industrial Engineering Computations*, vol. 11, pp. 443–468, 2020.

[21] D. Bandara, M. E. Mayorga and L. A. McLay, 'Optimal dispatching strategies for emergency vehicles to increase patient survivability,' *Int. J. of Operational Research*, vol. 15, no. 2, pp. 195–214, Aug. 2012, ISSN: 1745-7645. DOI: 10.1504/IJOR.2012.048867.

[22] T. Van Barneveld, R. Van Der Mei and S. Bhulai, 'Compliance Tables for an EMS system with Two Types of Medical Response Units,' *Computers & Operations Research*, vol. 80, pp. 68–81, Nov. 2016, ISSN: 0305-0548. DOI: 10.1016/j.cor.2016.11.013.

[23] V. Bélanger, A. Ruiz and P. Soriano, 'Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles,' *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 1–23, Jan. 2019, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.02.055.

[24] J. Holler, R. Vuorio, Z. Qin, X. Tang, Y. Jiao, T. Jin, S. Singh, C. Wang and J. Ye, 'Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem,' *arXiv*, Nov. 2019. DOI: 10.48550/arXiv.1911.11260. eprint: 1911.11260.

[25] D. Neira, J. W. Escobar and S. I. McClean, 'Ambulances Deployment Problems: Categorization, Evolution and Dynamic Problems Review,' *International Journal of Geo-Information*, vol. 11, no. 2, pp. 1–37, Feb. 2022, ISSN: 2220-9964. DOI: 10.3390/ijgi11020109.

[26] R. J. McCormack and G. Coates, 'A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival.,' *Elsevier*, May 2015. [Online]. Available: https://dro.dur.ac.uk/15555.

[27] E. Erkut, A. Ingolfsson and G. Erdogan, 'Ambulance Deployment for Maximum Survival,' *ResearchGate*, Oct. 2010. [Online]. Available: https://www.researchgate.net/publication/255579100_Ambulance_Deployment_for_Maximum_Survival.

[28] H. Setzler, C. Saydam and S. Park, 'EMS call volume predictions: A comparative study,' *Comput. Oper. Res.*, vol. 36, no. 6, pp. 1843–1851, Jun. 2009, ISSN: 0305-0548. DOI: 10.1016/j.cor.2008.05.010.

[29] Z. Zhou, 'Predicting Ambulance Demand,' Ph.D. dissertation, Aug. 2015. [Online]. Available: https://ecommons.cornell.edu/handle/1813/41181.

[30] M. A. Mahmood, J. E. Thornes, F. D. Pope, P. A. Fisher and S. Vardoulakis, 'Impact of Air Temperature on London Ambulance Call-Out Incidents and Response Times,' *Climate*, vol. 5, no. 3, p. 61, Aug. 2017, ISSN: 2225-1154. DOI: 10.3390/cli5030061.

[31] F. Mannering, 'Temporal instability and the analysis of highway accident data,' *Analytic Methods in Accident Research*, vol. 17, pp. 1–13, Mar. 2018, ISSN: 2213-6657. DOI: 10.1016/j.amar.2017.10.002.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction (Adaptive Computation and Machine Learning series)*. Bradford Books, Nov. 2018, ISBN: 978-0-26203924-6.

[33] S. Bhatt, 'Reinforcement Learning 101 - Towards Data Science,' *Medium*, Apr. 2019, ISSN: 2450-1292. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292.

[34] *6.6 Actor-Critic Methods*, [Online; accessed 18. May 2023], Dec. 2018. [Online]. Available: http://www.incompleteideas.net/book/ebook/node66.html.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, 'Proximal Policy Optimization Algorithms,' *arXiv*, Jul. 2017. DOI: 10.48550/arXiv.1707.06347. eprint: 1707.06347.

[36] L. A. McLay and M. E. Mayorga, 'A Dispatching Model for Server-to-Customer Systems That Balances Efficiency and Equity,' *Manufacturing & Service Operations Management*, Dec. 2012. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/msom.1120.0411.

[37] L. A. Albert, 'A mixed-integer programming model for identifying intuitive ambulance dispatching policies,' *J. Oper. Res. Soc.*, pp. 1–12, Nov. 2022, ISSN: 0160-5682. DOI: 10.1080/01605682.2022.2139646.

[38] X. Li and C. Saydam, 'Balancing ambulance crew workloads via a tiered dispatch policy,' *Pesqui. Oper.*, vol. 36, pp. 399–419, Sep. 2016, ISSN: 0101-7438. DOI: 10.1590/0101-7438.2016.036.03.0399.

[39] G. M. Carter, J. M. Chaiken and E. Ignall, 'Response Areas for Two Emergency Units,' *Oper. Res.*, Jun. 1972. [Online]. Available: https://pubsonline.informs.org/doi/epdf/10.1287/opre.20.3.571.

[40] K. Liu, X. Li, C. C. Zou, H. Huang and Y. Fu, 'Ambulance Dispatch via Deep Reinforcement Learning,' *ResearchGate*, pp. 123–126, Nov. 2020. DOI: 10.1145/3397536.3422204.

[41] Y. Liu, F. Wu, C. Lyu, S. Li, J. Ye and X. Qu, 'Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform,' *Transportation Research Part E: Logistics and Transportation Review*, vol. 161, p. 102 694, May 2022, ISSN: 1366-5545. DOI: `10.1016/j.tre.2022.102694`.

[42] Z. Qin, H. Zhu and J. Ye, 'Reinforcement Learning for Ridesharing: An Extended Survey,' *arXiv*, May 2021. DOI: `10.1016/j.trc.2022.103852`. eprint: `2105.01099`.

[43] O. Elfahim, E. M. B. Laoula, M. Youssfi, O. Barakat and M. Mestari, 'Deep Reinforcement Learning Approach for Emergency Response Management,' in *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, IEEE, pp. 18–20. DOI: `10.1109/ISCV54655.2022.9806108`.

[44] C. Hua and T. Zaman, 'Optimal Dispatch in Emergency Service System via Reinforcement Learning,' in *AI and Analytics for Public Health*, Cham, Switzerland: Springer, Jan. 2022, pp. 75–87. DOI: `10.1007/978-3-030-75166-1_3`.

[45] J. J. Cochran, L. A. Cox Jr., P. Keskinocak, J. P. Kharoufeh and J. C. Smith, *Wiley Encyclopedia of Operations Research and Management Science*. Jun. 2010, ISBN: 978-0-47040063-0. DOI: `10.1002/9780470400531`.

[46] T. Morimura, K. Ota, K. Abe and P. Zhang, 'Policy Gradient Algorithms with Monte-Carlo Tree Search for Non-Markov Decision Processes,' *arXiv*, Jun. 2022. DOI: `10.48550/arXiv.2206.01011`. eprint: `2206.01011`.

[47] K. Jordahl, J. V. den Bossche, M. Fleischmann, J. Wasserman, J. McBride, J. Gerard, J. Tratner, M. Perry, A. G. Badaracco, C. Farmer, G. A. Hjelle, A. D. Snow, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, N. Eubank, maxalbert, A. Bilogur, S. Rey, C. Ren, D. Arribas-Bel, L. Wasser, L. J. Wolf, M. Journois, J. Wilson, A. Greenhall, C. Holdgraf, Filipe and F. Leblanc, *Geopandas/geopandas: V0.8.1*, version v0.8.1, Jul. 2020. DOI: `10.5281/zenodo.3946761`. [Online]. Available: `https://doi.org/10.5281/zenodo.3946761`.

[48] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, 'Stable-baselines3: Reliable reinforcement learning implementations,' *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: `http://jmlr.org/papers/v22/20-1364.html`.

[49] *Figure 1 The Oslo road network (NVDB)*. [Online; accessed 29. Apr. 2023], Apr. 2023. [Online]. Available: `https://www.researchgate.net/figure/The-Oslo-road-network-NVDB_fig1_321262463`.

[50] T. Ni, B. Eysenbach and R. Salakhutdinov, 'Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs,' *arXiv*, Oct. 2021. DOI: `10.48550/arXiv.2110.05038`. eprint: `2110.05038`.

[51] H. Xu, P. Pu and F. Duan, ‘Dynamic Vehicle Routing Problems with Enhanced Ant Colony Optimization,’ *Discrete Dyn. Nat. Soc.*, vol. 2018, Feb. 2018, ISSN: 1026-0226. DOI: 10.1155/2018/1295485.

[52] P. Stodola and J. Mazal, ‘Ant colony optimization algorithm for Multi-Depot Vehicle Routing Problem with Time Windows,’ *OPT-i 2014 - 1st International Conference on Engineering and Applied Sciences Optimization, Proceedings*, pp. 184–192, Jan. 2014. [Online]. Available: https://www.researchgate.net/publication/286115453_Ant_colony_optimization_algorithm_for_Multi-Depot_Vehicle_Routing_Problem_with_Time_Windows.

Appendix A

Additional Material

A.1 Make intersection graph (textual)

Given below is a stepwise textual explanation of the *makeIntersectionGraph* algorithm.

1. Remove self referring edges.
2. Detect intersection nodes by having a degree bigger than 2.
3. Make a temporary copy of the graph, and remove these nodes from the graph.
4. Find all connected components in the graph.
5. Label edges by their connected component sub graph.
6. Collapse the nodes and edges to contain 2 nodes and 1 edge in each subgraph. Sum up their features.
7. Add back and re-connect the intersection nodes removed in step 2.
8. Find connected components.
9. If there exist more than one component, search original graph for edges which connects these components. Add these edges to the graph.
10. Remove sub-components which has no edge in original graph which connects them to the graph.
11. The result is a graph without sub-components which represents the entire road network in Oslo and Akershus.

A.2 Make grid-cell graph (textual)

Using this grid cell division, edges in the subgraphs which connects two subgraphs together are sought for. If two subgraphs have a such an edge which connect them together, an edge is made between these grid cells in the resulting grid-road-network-graph H. The edge represents that it is possible to travel by car between those two grid cells.

A.3 Makeshift survival function estimation

Given the event S that a patient survives, the survival function $s(x,p)$ is estimated (Equation A.1). Where x is the response time, p is the priority of the incident, and $P(S)$ is the probability of survival. The probability of survival $P(S)$ is unknown and assumed to equal one.

$P(x \wedge p)$ is estimated by CDF of the dataset available (Equation A.2). Where $\mathcal{D}$ is the dataset with records $\mathcal{D} = (r_0, r_{...}, r_n)$. Where each record contains (t_r, i_p) tuples. Equation A.3 and Equation A.4 is used to signal that a condition is true given a record from the dataset, t_r and i_p are the response time and incident priority of record r.

Note that we do not include the probability of the incident being $i_p = p$ since an incident with priority p already happened. Furthermore, we are calculating the probability of the response time being higher than x since the response time should be as low as possible (mimicking a survival function). This probability then decreases as x gets higher.

$$s(x,p) = P(S|x,p) = P(S)P(x \wedge p) \approx P(x|p) \tag{A.1}$$

$$P(x|p) \approx P(t_r > x | i_p = p) = \frac{\sum_{r \in \mathcal{D}} c(r,p,x)}{\sum_{r \in \mathcal{D}} c_2(r,p,x)} \tag{A.2}$$

$$c(r,p,x) = \begin{cases} 1 & \text{If } t_r > x \wedge i_p = p, \\ 0 & \text{Otherwise.} \end{cases} \tag{A.3}$$

$$c_2(r,p,x) = \begin{cases} 1 & \text{If } i_p = p, \\ 0 & \text{Otherwise.} \end{cases} \tag{A.4}$$

In an optimal scenario, this estimation should be conditioned on the position of the dispatched ambulance and incident. Since two incidents with the same actual urgency will have different response times depending on the position of the closest ambulance to each incident. It is assumed that the dataset is large enough to validate this estimation.

A.4 Breakdown of updating ambulances

This section breaks down how the ambulances are updated and dispatched, referred to as "Update ambulances" and "Dispatch ambulance" in Figure 5.2 and Figure 5.3.

Furthermore, the ambulance location (section 5.1.2) is implemented as a *path* class used in the algorithms. This also handles the ambulance processing time (section 5.1.2), which is added to the time needed to complete the route. This class has a *getPos* function which returns the current position of the ambulance and if the route is finished. Furthermore, it stores t_e, which is $\hat{t} - t_n$ as explained in section 5.1.2.

Each ambulance takes on a status value s. This represents whether the ambulance is at its base station $s = 0$, traveling to an incident $s = 1$, at the incident location $s = 2$, traveling to the hospital $s = 3$, at the hospital $s = 4$, or traveling to its base station $s = 5$.

All the ambulances in the simulator are stored in a list $a \in A$. The function *simulator:UpdateAmbulances* (Algorithm 4), goes through each ambulance a and calls its *ambulance:updateLocation* (Algorithm 6). *simulator:UpdateAmbulances* then calculates the reward (Equation 5.6) based on information gathered from *ambulance:updateLocation*.

"Dispatch ambulance" in Figure 5.2 and Figure 5.3 calls *ambulance:assignIncident*(Algorithm 5). Which assigns the ambulance to the given incident, calculates the shortest route, and updates the status of the ambulance. *ambulance:updateLocation* deals with the status change of the ambulance, its path (route), and response time. The variable *os* is false when the switch to night allocation has determined that this ambulance should be off duty once it has arrived at its base station (section 5.1.2). This then sets the status $s = -1$, so the ambulance is not updated anymore (See line 2 in *simulator:UpdateAmbulances*). All the $s = -1$ ambulances are set to $s = 1$ once the switch to the day shift happens. Recursion is also applied until a route is not finished or has arrived at its base station.

Both *ambulance:assignIncident* and *ambulance:updateLocation* use the *shortestPath* function. This function checks the cache for the given target locations; if it is there, it is returned. If it is not there, then A* search is applied. *multiSourceDjikstra* works the same way but has a separate cache (with l as key). *multiSourceDjikstra* returns the shortest path from multiple locations (hospitals) to the given location (incident location).

A.5 Incident queue (textual)

Elaboration of the *simNoAvail* algorithm. Also see Figure A.1

1. Check how much time until each ambulance changes its current status. Assign its minimum value to t_a.
2. Simulate t_a time forward ($t = t + t_a$)
3. Subtract $t_p = t_p - t_a$
4. while $t_p \leq 0$

Algorithm 4 *simulator:UpdateAmbulances*

Require: Class variables: Ambulances $a \in A$ with feature a_s ambulance status and function *updateLocation*, grid-cell graph H, responeTime list T. Function arguments: δt time since last update, t current simulation time.
Ensure: Updated ambulances and calculated reward

```
 1: $r = 0$                                                   ▷ Reward
 2: for $a \in A | a_s \geq 0$ do                    ▷ Update only onShift ambulances
 3:     status, avail, rTime = $a.updateLocation(H, \delta t, t)$
 4:     $w_t = -1$                     ▷ Wait time, reward contribution for this ambulance
 5:     if rTime $\neq$ -1 then                 ▷ Ambulance have collected response time
 6:         $T$.insert(rTime)
 7:         $a.r_t = -1$                            ▷ Reset response time of ambulance
 8:         $w_t$ =rTime                         ▷ Set response time as a part of reward
 9:     else if status = 1 then          ▷ Ambulance is currently travelling to incident
10:         $w_t = a.ia.getWaitTime(t)$
11:     end if
12:     if $w_t \neq -1$ then
13:         $r = r - w_t$                                  ▷ Add contribution to reward
14:     end if
15: end for
16: return $r$
```

Algorithm 5 *ambulance:assignIncident*

Require: Class variables: l ambulance location, b ambulance is busy (boolean), s ambulance status, os ambulance is onshift (boolean), ia incident assigned, p current path instance. Function arguments: H grid-cell graph, t current simulation time, i incident
Ensure: Assigns incident i to this ambulance

```
1: if $(\neg b \vee s \in \{0, 5\}) \wedge os$ then
2:     $b = 0$                                                 ▷ Set to is busy
3:     $s = 1$                              ▷ Set status to 1 (travelling to incident)
4:     $ia = i$                             ▷ Assign this incident to this ambulance
5:     spath = $shortestPath(H, l, i_l)$  ▷ find shortest path from this ambulance to
   the incident. A*/Cache
6:     $p$ = new $path$(spath, $t$)
7: end if
```

Algorithm 6 *ambulance:updateLocation*

Require: Class variables: l ambulance location, ba ambulance base station, s ambulance status, b ambulance busy (boolean) p current path instance (implements), ia current incident assigned, h_l hospital gridlocations, t_e extra time, t_r response time, os ambulance is onShift (boolean). Function arguments: H grid-cell graph, δt time since last update, t current simulation time.

Ensure: Updated location and status of this ambulance.

```
1:  if $s > 0$ then
2:      $l$,done = $p$.getPos($\delta t$, $t_e$)
3:      $t_e = 0$
4:      if done then                                          ▷ Is finished with route
5:          $t_e = p.t_e$           ▷ p.$t_e$ is left over time after completing route(path)
6:          $b = 1$                                           ▷ Assume is available
7:          $s = s + 1$                                    ▷ Increase status variable
8:          if $s > 5$ then                                ▷ Arrived at base station
9:              if $os$ then
10:                 $s = 0$                            ▷ Set at base station if onShift
11:             else          ▷ Is False when this ambulance should be offduty as
    determined by day/night shift allocation switch
12:                 $s = -1$                                        ▷ Mark as offduty
13:             end if
14:         else if $s = 2$ then                         ▷ Arrived at incident location
15:             $ia$.update($\delta t$)                      ▷ Updates reponse/wait time
16:             $t_r = ia$.responseTime                       ▷ Fetches response time
17:             $ia$ = None
18:             spath = multiSourceDjikstra($H$, $h_l$, $l$)     ▷ Finds closest hospital.
    A*/Cache
19:             spath.reverse()          ▷ Make path from ambulance to hospital
20:             $p$ = new path(spath, $t$)
21:             $b = 0$                                               ▷ Is now busy
22:             $s = 3$                                      ▷ Travelling to hospital
23:         else if $s = 4$ then                              ▷ Finished at hospital
24:             $s = 5$
25:             spath = shortestPath($H$, $l$, $ba$)        ▷ find shortest path from this
    ambulance to its base station. A*/Cache
26:             $p$ = new path(spath, $t$)
27:             $b = 1$                                              ▷ Is available
28:         end if
29:         return updateLocation($H$, $\delta t$, $t$)        ▷ Apply recursion until a path
    (route) is not finished
30:     else
31:         $b = 0$                             ▷ Is unavailable. Currently on route
32:     end if
33: end if
34: return $s$,$b$,$t_r$
```

4.1 create incident with timestamp $t - |t_p| = t + t_p$, and add it to the queue Q.
4.2 check how much time until next incident happens. Assign it to value t_i
4.3 $t_p = t_p + t_i$

5. Update ambulances.
6. Repeat until an ambulance is available.

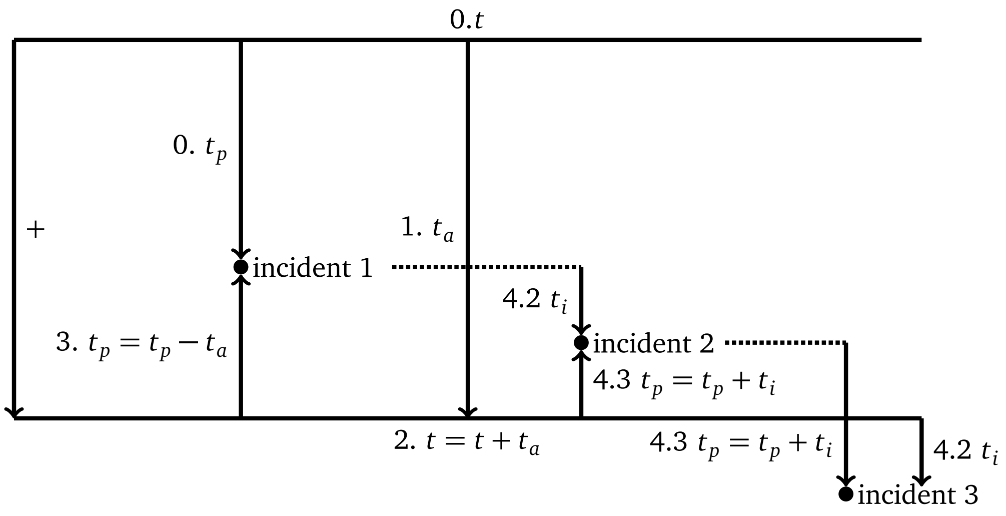

Figure A.1: Shows visually how incidents during simulation while waiting for ambulance. Upwards arrows are negative, while downwards positive. Numbered according to step-wise list.

A.6 All externally created links

1. https://github.com/JonEliasMoen/Amb-public
2. https://joneliasmoen.github.io/
3. https://joneliasmoen.github.io/coverage.html
4. https://joneliasmoen.github.io/popLinreg.html

NTNU

Norwegian University of
Science and Technology