Project 12: Dispatching ambulances in the Netherlands with Deep Reinforcement Learning

Francesca Drummer
Computer Science-EEMCS
TU Delft, 5413990
F.K.drummer@student.tudelft.nl

Thomas Georgiou
Electrical Engineering-EEMCS
TU Delft, 5395186
T.Georgiou@student.tudelft.nl

Silvana van der Voort
Biomedical Engineering-3ME
TU Delft, 4398912
S.vandervoort@student.tudelft.nl

Athanasios Theocharis
Computer Science-EEMCS
TU Delft, 5621771
A.Theocharis-1@student.tudelft.nl

***Abstract*—In the Netherlands, the number of ambulances that arrive within the approximate target time has been decreasing. One possible reason for this is that the currently employed dispatching algorithm follows a greedy heuristic and thus is not optimal. This paper presents an improved ambulance dispatch algorithm using deep double Q-learning. The results show that the deep reinforcement learning model outperforms the greedy benchmark algorithm. Furthermore, we reflect on the future work needed and the societal and ethical impact of the implementation.**

***Index Terms*—Reinforcement learning, AI, Ambulance dispatch, Q-learning, Double Deep Q-learning Network**

I. Introduction

In situations of emergencies, an ambulance needs to be available at the accident location as soon as possible. The decision about which is the ideal ambulance to send out is referred to as dispatch problem. Solving the dispatch problem most efficiently is of high relevance since a few minutes can impact a patients' health [1]. For that reason, the norm in the Netherlands is that an ambulance has to reach the accident location within 15 minutes after the emergency call. However, several news articles [2] [3] claim that the arrival time has increased to an average of 16 minutes in the last year. In October 2021, the house of representatives of the Netherlands has discussed this issue due to its urgent matter. They stated the following that only in 6 of the 25 safety regions the maximum arrival time of 15 minutes (during emergencies) is respected within the approximate target value of 95% of the times. According to the other regions, the main reason for degrading arrival time percentage is due to COVID-19, e.g. thorough cleaning regulations. The plan of action is to optimize the deployment of ambulance care, think of optimizing the care coordination and process of triage [4].

Improving the ambulance dispatch system is one way of improving the ambulance care and hence improve the arrival time directly. It is important to know that in the Netherlands, ambulance are not located at the hospitals but rather at separate ambulance bases to have a better coverage of the area. There are several ways to decide which ambulance to dispatch. One can look at the ideal locations of ambulance bases related to the accidental locations, by applying the method Maximal Covering Location Problem (MCLP) as described by Yin and Mu [5]. The approach mentioned in the article is a static approach and therefore less flexible. Jaghtenberg, Bhulai and Van der Mei [6] described a different heuristic dynamic approach: maximum expected covering location problem formulation (MEXCLP), which shows a significant improvement compared to the static approach. They state that it will be more difficult to adjust the algorithm for a realistic approach. For example, changing accident probabilities or travel times during office hours are not implementable. The same authors also wrote an article on the Markov Decision Process (MDP), where they stated that this approach can reduce late arrivals by 13% [7]. Reinforcement learning (RL) learns how to behave in an MDP environment. This means that due to its way of learning it can be more dynamic and will enhance its compatibility to practice. Moreover, RL has been applied in several situations to optimize heuristic problems, meaning solving a problem when the classical methods are too slow. In [8], RL is used for dispatching ambulances, because the traditional alternatives are not flexible enough for the real world. In this project, RL needs to be combined with a Deep Neural Network (DNN) to generalize over the multiple safety regions in the Netherlands.

Section II elaborates on the assumptions of the project and the data used. Section III explains the methods of the project consisting of the greedy, RL, Q-learning, Double Deep Q-learning Network (DDQN) and its training. Section IV provides relevant results regarding the performance of the DDQN model compared to the greedy. In section V, the interpretation of results and future research will be discussed. Last, the conclusions regarding the problem are described in section VI.

II. Problem scope

In this paper, RL is applied to investigate how ambulances can be dispatched. In this section, the assumptions taken and

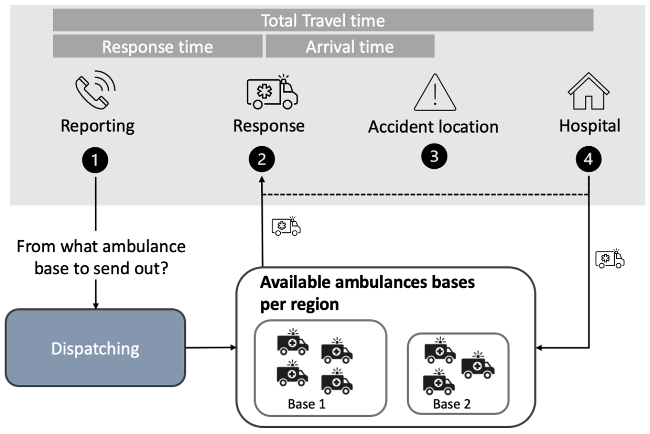


Fig. 1: Emergency response chain of events. Inspired from Figure 1 in [1]

the modeling data used are presented.

A. Assumptions

Figure 1 illustrates the chain of events after an emergency call occurred. First, an accident occurs and is reported. We assume the following for accidents:

Assumption 1. *The occurrence of accidents is independent of previous accidents and asynchronous.*

Assumption 1 allows us to generate accidents a priori of determining the decision of the dispatching algorithm. More specifically, it assumes that each accident occurs with a given probability that is independent of previous events and also that only one accident occurs per minute. After an accident has been reported an ambulance is dispatched according to the algorithm. As there are 25 independent safety regions in the Netherlands we also divide the problem into regions in our model [9]. We assume that

Assumption 2. *The occurrence of accidents and ambulance dispatching is independent across regions.*

By assuming independence across regions we realistically model the emergency response in the Netherlands. At the same time, this allows for independent training of the algorithm across different regions. Additionally, our model implements that each safety region consists of several zip-codes and in some of these zip codes, ambulance bases are presented. Following, we assume that

Assumption 3. *Ambulances can only be dispatched when they are available at the base.*

Figure 1 shows that an ambulance is sent out from an available base per region. The ambulance does only become available again after returning to the base from the hospital. Thus, we do not take into account that an ambulance could also respond to an accident from the hospital or on the way towards the base. Note, that every 24 hours the environment is completely reset, meaning all ambulances become available again. This feature is required to allow training over multiple regions.

Lastly, we assume that

Assumption 4. *The total travel time is deterministic and known in advance.*

This is a simplifying assumption to improve the modelling ability and future research will focus on introducing stochastic travel times.

B. Modeling data

In the paper, the routes of an ambulance consist of three parts: (i) from ambulance base to the accident location (i.e. arrival time), (ii) from the accident location to the hospital, (iii) from the hospital back to the base. The data for the model consisted of:

- Distances between zip-codes per region in seconds
- Population per zip-code of all 25 regions.
- accidental rate per region per year.
- Zip-codes of ambulance bases and hospitals per region.
- Number of ambulances per base.

III. Methods

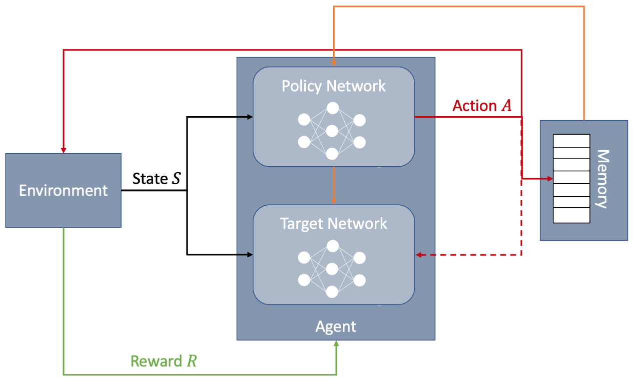
This section describes the dispatch algorithm implemented. First, a greedy dispatch strategy that serves as benchmark is explained. However, because the ambulance dispatch problem is a combinatorial optimization problem it is usually too complex to be solved optimally with conventional methods such as the greedy heuristic. Therefore, we developed a DDQN algorithm to solve the problem in a more efficient way. For both, the greedy heuristic and DDQN algorithm, the goal is to maximize the reward which corresponds to minimizing the arrival time (Figure 1). In particular, the reward is

$$\frac{10000}{arrival\ time}$$

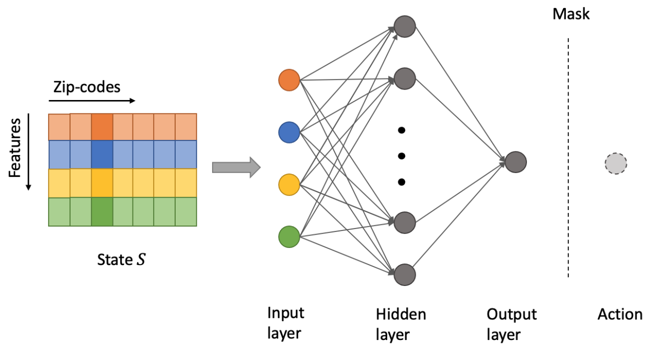
where arrival time is the travel time of an ambulance from the base to the accident location measured in seconds.This is an arbitrary reward function, causing rewards in the range of [10,100].

A. Greedy

The greedy heuristic is used as a benchmark for the RL dispatching algorithm. The greedy algorithm dispatches the ambulance closest to an accident. More precisely, the action selected in this algorithm is the one with the lowest arrival time, which corresponds to the highest reward. It is expected that the arrival time is low in the beginning. However, in the long run arrival times are expected to increase due to congestion of accidents and unavailability of nearby ambulances. When no ambulance is available, the accident will be added to a waiting list. An ambulance will be dispatched to the first accident location on the waiting list as soon as one becomes available.



(a) Illustration of Deep Reinforcement Learning (DRL); The environment sends a state representation S to the policy and target network. The policy network returns the optimal action chosen, stores it in the memory and updates the environment with it. Every n iterations the target network is updated and every k iterations, a mini-batch of size M, consisting of state S, action A and new state S' is retrieved from the memory to train the policy network.



(b) Illustration of DNN; the input is a feature x zip code matrix and the DNN consists of three fully connected layers. At the output layer masking is applied to return only the values of the possible bases.

Fig. 2: Overview of the DRL loop and the DNN architecture.

B. *Reinforcement Learning*

The goal of the RL model, is to overcome the limitations of the greedy algorithm and achieve low arrival time in the long run. An RL agent learns through trial and error by taking actions and experience rewards [10]. In our case, the agent is the instruction of the ambulance; the environment consists of zip-codes of the ambulance bases and hospitals and the sampling rate of accidents in region. More specifically, an RL implementation can be distinguished in five different blocks:

*1) **Environment**:* The environment represents the model of the world the agents act in and contains all observable and not observable information relevant to us. In our case the environment has information about the (i) accident sampling rate per zip-code and the (ii) distances and zip-codes of the ambulances and hospitals. The latter is necessary to evaluate the travel time of an ambulance to an accident location. As shown in Figure 2a the agent receives a state representation from the environment and the environment updates itself given action A. Since the action A defines which ambulance is dispatched, the environment updates the availability of ambulances accordingly. The environment saves the total travel time (TTT) of a dispatched ambulance. After the TTT passed the environment updates that the dispatched ambulance is back at its base. As illustrated in Figure 1 the TTT is the sum of the: (i) response time (1 min.), (ii) arrival time (dependent on travel distance to accident location), (iii) buffer time at the accident location e.i. put the patient in the ambulance (15 min.), (iv) travel time from accident to hospital and (v) travel time back to base.

*2) **Agent**:* The agent senses the environment though a state representation S, decides on an action A and receives a reward R (Figure 2a). After sensing the environment the agent returns an action A describing where the ambulance will go and then receives an reward R is the arrival time, which should be minimized.

*3) **Memory**:* Memory is used to store previously experienced states, actions and rewards. Its primary purpose is to provide replayable situations to train the neural network while removing training bias that stems from sequential states.

*4) **Learner**:* This code block is responsible to train the model, utilizing stored experiences in memory.

*5) **Runner**:* Its sole goal is to execute the implemented algorithm.

Nota that in this report we implement a version of multi-task RL as every state representation is different (in our case due to different sizes of regions).

C. *Q-learning*

Q-learning is a form of RL which uses Q-values to iteratively improve the agents behaviour. The Q-value $Q(s, a)$ represents the value of an action a in a specific state s and is updated for each state continuously. An advantage of Q-learning is that it uses off-policy which allows it to differentiate between learning and acting policy. However, Q-learning takes a long time to train since a Q-value is updated only once per action. Additionally, Q-learning requires a large amount of storage space for the rewards. Thus, basic Q-learning is ineffective to solve complex problems in large environments [10].

D. *Double Deep Q network*

Solving the ambulance dispatching problem in the Netherlands requires an effective learning algorithm in a large environment. Therefore, a DDQN [11] is used which combines Q-learning with a DNN. Additionally, in this case, the DDQN allows us to generalize the dispatch algorithm while training it for every region independently (Assumption 2). That means, the DDQN is flexible to adjust to a new problem setting and small differences in regions, e.g. adding an ambulance to a dispatch center or removing one in case of failure. Having one general network means that no retraining is required in case of small perturbations. DDQN combines two approaches; (i) experience replay and (ii) a target Q technique [12]. The experience replay eliminates the correlation between states, actions and rewards by storing the experiences in a buffer and selecting them randomly as learning data (enabled through the memory component in Figure 2a). Secondly, the target Q technique prepares a policy and target network separately as shown in Figure 2a. The target Q technique reduces correlations and overestimation of Q-values by learning the Q network based on the target value obtained by the target network. Together, the experience replay and target Q technique, stabilize the learning of the DDQN [13].

Figure 2b shows the chosen structure of the DDQN. The input is a `feature x number of zip-codes` in region matrix where the four features contain information about:

1) Boolean indicating if the zip code has an ambulance base
2) Number of ambulances available at zip-code
3) Total travel time (TTT)
4) Time during day

These features contain the information required to decide on the optimal ambulance to dispatch. In case a feature is not relevant the network will learn to ignore it by reducing the weight of the network edges. Masking is applied at the output layer to select the Q-values of the bases with available ambulances in the current region. The output of the DDQN defines at which zip-code the ambulance should be dispatched.

E. Training of DDQN

Algorithm 1 shows the high-level overview of the DDQN training. First, the environment, agent and memory are initialized. Then the network is trained for a specified number of episodes. The number of episodes needs to be high enough for the network to learn which features of the input have more influences of the output and which less. For each episode the DDQN is trained for a specific region. The region is chosen uniformly at random and changes between episodes. Each episode is one day long, where the day is represented by seconds. Note that the agent uses epsilon-greedy exploration. That means, it uses a declining probability of choosing actions in a random manner, rather than taking the neural network recommended action. This is used in order to explore as many actions as possible in the beginning and while the episodes pass and the epsilon is declining, we are taking the actions that our policy network produces (exploitation).

Algorithm 1 High-level view of DDQN training

```
procedure TRAINING LOOP DDQN
    Initialize environment
    Set up Deep RL agent (policy & target network, memory)
    for episode ∈ NR_EPISODES do
        Select random region
        for second ∈ TIME_DAY do
            Train DQN for region
        end for
    end for
end procedure
```

F. Self-attention mechanism

To further improve the performance of the dispatch algorithm we extend the DDQN with multi-head self-attention. Self-attention allows the agent to pay more attention to the the state representation. More specifically, we extended our DDQN implementation with a multi-head self-attention mechanisms. This structure is based on the transformer module proposed in [14]. The multiple heads allow to network to relate the input features of the state representation to each other in different ways.

IV. Results

This section reports the results from the greedy heuristic and our algorithm without self-attention. Figure 3 shows the difference of rewards between the two as the number of episodes increases. The difference is calculated as `model rewards - greedy rewards`. That means, our model obtains higher rewards when the difference curve (blue) is above the baseline (orange). Note that the difference curve is the reward difference of the model for all regions. Figure 3 shows that after 10.000 episodes our model outperforms the greedy heuristic most of the time in terms of higher rewards. The epsilon is reduced heavily near that point, so the exploration is over and the exploitation has started.

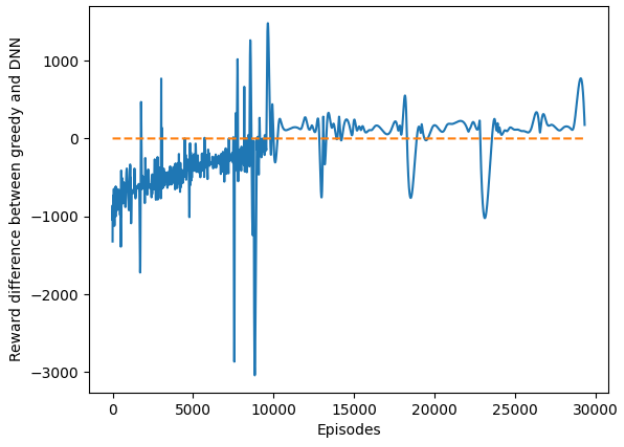


Fig. 3: Training progress of our DDQN algorithm compared to the greedy heuristic; shown by the difference curve of the rewards (blue), calculated as `model rewards - greedy rewards` in relation to the baseline (orange).

In Figure 4, the difference of the cumulative rewards of the DDQN and the greedy algorithm is plotted for a month of accidents. As expected, at the beginning the benchmark model outperforms the RL model. After approximately 13000 accidents they perform the same. Only after 2000 accidents an acute increase of the cumulative reward difference can be observed. Note that the difference between rewards increases step-wise and not continuously which means that our model and the greedy agent often choose the same actions. Only in specific situations these decisions differ which cause a step-wise increase of the cumulative reward.

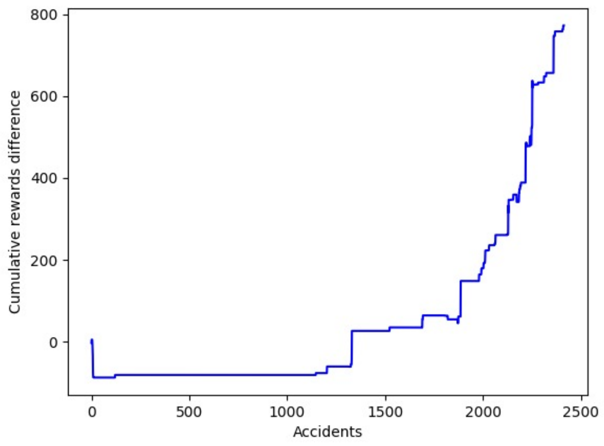


Fig. 4: Cumulative difference in rewards of the model actions and the greedy algorithm actions, showcasing the performance of our trained model over the greedy algorithm throughout a month of accidents.

V. Discussion

A. Observation and Interpretation

The results show that the DDQN algorithm outperforms the greedy heuristic over the long run. However, the step-wise increase in cumulative reward from Figure 4 also shows that the DDQN algorithm and greedy heuristic often make the same decisions. From this we can conclude that the greedy implementation works better than expected for this particular dataset. The good benchmark performance of the greedy agent was not expected for such a complex combinatorial problem and can have multiple reasons. First, we speculate that the location of the bases and allocation of the ambulances was already spread optimally across the Netherlands. Besides that, it can also be that the dataset does not take into account extreme cases or ambulance repair which simplifies the problem. Nonetheless, the performance of the DDQN is superior than the greedy in the long run as it saves more time, which is an important aspect when it comes to ambulances and emergencies. Also with its possibility to adapt it to the real world situation, it is more beneficial as an implementation tool especially in the long run.

In addition to the DDQN algorithm we also ran the extension with self-attention. The results with self-attention were more unstable than without. For that reason only the results with self-attention were included We believe that the instability of the self-attention training arises from the fact that our environment is not complex enough. Additionally, it might be that the features are not representative enough for which it is not possible to learn relations between the features.

B. Future work

During this project, we have made multiple modelling assumptions. Here we will discuss some of the limitations that the assumption impose:

(i) In real life, the amount of time an ambulance is at the accident can vary from five minutes to several hours in real life depending on the severity of the accident. Thus, the travel time of an ambulance is neither deterministic nor known in advance (Assumption 4). Therefore, we suggest to replace the constant of fifteen minutes with a variable. Ideally, this variable is drawn from a Gaussian distribution with mean at fifteen minutes and can vary between one minute and one hour.

(ii) The accidental sampling distribution currently applied is uniform. However, there are times and days were the need for ambulances are higher than normal, for example on New Years Eve, the amount of ambulances needed increases after the fireworks are set off. This has not been taken into account yet. By choosing a more representative sampling distribution for the accident occurrence this could be adjusted.

(iii) We dispatch only one ambulance at a time even though big emergencies (e.g. plane crash or collapsed buildings) would require multiple ambulances to be dispatched. This would not only influence the availability of the ambulances, but also create more problems for the dispatch of ambulances to the other accidents.

(iv) In our model, ambulances only become available after returning to their original base (Assumption 3). However, in theory, ambulances are available after delivering patients to the hospital. Additionally, when there are big events the ambulances could also be reallocated towards bases closer to these events which would reduce the arrival time.

It is important to note that the assumptions made, do not decrease the quality of the model. Even though assumption (i) would cause an ambulance to be available sooner, we took the most undesirable case where it would always go to the hospital. And assumption (ii) was left out as it would be case dependent and there was not enough data for this. The other two assumptions are less important, as assumptions (iii) and (iv) do not occur often and were therefore left out. Moreover, our research focuses on minimizing the arrival time by utilizing RL. Since both the greedy and the DDQN algorithm are based on the same assumptions our comparison is not influenced by any of these limitations.

For future research, the aforementioned assumptions and extreme cases could be taken into account (in its existing order) to improve the model, before implementing it into the ambulance dispatch system. For example, the dispatch algorithm could use a general adversarial network (GAN) to enable the reallocation of ambulances which we assumed as not possible (Assumption 3). In that case, the DDQN would consist of two networks: one deciding where the ambulance should be dispatched and another to decide on which location it should be reallocated to. Besides that, we suggest to train the network with self-attention over more episodes to achieve a more meaningful difference to the greedy heuristic.

C. *Societal and Ethical impact*

A common problem of DNN's is their black box behaviour. That means, DNN's can be viewed in terms of input and output, without knowing the internal workings. In our case, we know the state representation (e.i. where an accident occurs and where ambulances are available) and the output (which ambulance is sent out) but we do not get any information about the reasoning behind this output. The limited insight into the networks working reduces the interpretability of DNN's and hence, the trust of users into the final decision. Thus, relying on an algorithm of which we do not understand the workings should be critically evaluated especially when the network is involved in life threatening decision.

We suggest that this critical evaluation can be done during the test phase of the network. The goal of the test phase would be to evaluate if the network dispatches the optimal ambulance. Due to the computational complexity of the problem the optimal ambulance to dispatch is unknown. Hence, it is impossible to check if the network returns the optimal solution. However, during the networks reward can be compared to the rewards of current approaches. Since several institutions want a more effective method to reduce the arrival time of an ambulance, we believe that deciding for the overall best performing algorithm could be valid option. Especially, since a patients life is on the line and the 95% norm is currently not met.

VI. Conclusion

For this project, a DDQN algorithm to solve the ambulance dispatch problem was developed. The results show that the DDQN algorithm returns higher rewards over the long run. It is expected that the DDQN is going to return a higher reward when the environment incorporates more complexity of the real world such as changes of accident distribution during big tragedies and travel time changes related to rush hours. Overall, the DDQN algorithm can help to improve the waiting time of the patient and therefore be a useful tool to increase the response value percentage of the ambulances to arrive on time.

Online content

The data and code are available at: https://github.com/athatheo/ambulance-dispatch-DDQN.

References

[1] Carvalho, Ana Captivo, M. Marques, Inês. (2019). Integrating the ambulance dispatching and relocation problems to maximize system's preparedness. European Journal of Operational Research. https://doi.org/10.1016/j.ejor.2019.11.056

[2] RTV Drenthe, 'Ambulances in Drenthe vaker te laat', September 2021 https://www.rtvdrenthe.nl/nieuws/173462/Ambulances-in-Drenthe-vaker-te-laat

[3] De Koning, Adrianne, 'Ambulances zijn vaker te laat tijdens tweede golf', https://www.pzc.nl/rotterdam/ambulances-zijn-vaker-te-laat-tijdens-tweede-golf a6e961cf/?referrer=https%3A%2F%2F

[4] de Jonge, Hugo, 'beantwoording kamervragen: Ambulances Noorden komen vaker te laat', October 2021. Rijksoverheid

[5] Yin, P., Mu, L. (2012). Modular capacitated maximal covering location problem for the optimal siting of emergency vehicles. Applied Geography, 34, 247–254. https://doi.org/10.1016/j.apgeog.2011.11.013

[6] Jagtenberg, C., Bhulai, S., van der Mei, R. (2015). An efficient heuristic for real-time ambulance redeployment. Operations Research for Health Care, 4, 27–35. https://doi.org/10.1016/j.orhc.2015.01.001

[7] Jagtenberg, C. J., Bhulai, S., van der Mei, R. D. (2017). Markov Decision Processes in Practice. In R. J. Boucherie N. M. van Dijk (Eds.), Optimal Ambulance Dispatching (pp. 269–291). Springer Publishing. https://doi.org/10.1007/978-3-319-47766-4_9

[8] Liu, K., Li, X., Zou, C. C., Huang, H., Fu, Y. (2020, November). Ambulance Dispatch via Deep Reinforcement Learning. Proceedings of the 28th International Conference on Advances in Geographic Information Systems. https://doi.org/10.1145/3397536.3422204

[9] Rijksoverheid, Veiligheidsregio's en crisisbeheersing https://www.rijksoverheid.nl/onderwerpen/veiligheidsregios-en-crisisbeheersing/veiligheidsregios

[10] Russell, S. J., Norvig, P. (1995). Artificial intelligence: A modern approach. Englewood Cliffs, N.J: Prentice Hall

[11] van Hasselt, H., Guez, A., Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1). arXiv:1509.06461.

[12] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). https://doi.org/10.1038/

[13] Jang, Beakcheol Kim, Myeonghwi Harerimana, Gaspard Kim, Jong. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2941229

[14] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need. arXiv:1706.03762