TÉCNICO LISBOA

Dynamic Sensor Network for Air Quality Monitoring Using Sequential Decision-Making Under Uncertainty

João Ribeiro Dias

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pedro Manuel Urbano de Almeida Lima
Dr. Tiago Santos Veiga

Examination Committee

Chairperson: Prof. José Alberto Rodrigues Pereira Sardinha
Supervisor: Prof. Pedro Manuel Urbano de Almeida Lima
Member of the Committee: Prof. Manuel Fernando Cabido Peres Lopes

September 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

This project was intended to explore the topic of sensor deployment to monitor pollution and reduce the uncertainty of the measurements.

It was conducted in the Spring semester of 2022 with the Department of Computer Engineering at Instituto Superior Técnico.

The student supervision was under the responsibility of professor Tiago Veiga from the Norwegian University of Science and Technology (NTNU), aided by professor Pedro Lima from Instituto Superior Tècnico, University of Lisbon. Part of the project uses data and software from the Ph.D. student Abdulmajid Murad and from Mykhaylo Marfeychuk's Master's Thesis.

A special thanks to these two professors, who were available to support me with the initial part of the Master's Thesis during my exchange program, and to Abdulmajid Murad as well as Mykhaylo Marfeychuk that provided the needed tools to implement the pipeline of the project.

Lastly, I would like to thank my parents, girlfriend, and friends, that always supported me and never let me give up.

Abstract

Air pollution is undoubtedly one of the most concerning problems faced by humanity. Several studies prove that the effects of pollution are causing different pathologies and other problems. Part of the approaches to mitigate such problems consists of monitoring the regions that need more attention.

Pioneer work on sensor placement for pollution monitoring used static sensor placement. This thesis goes beyond that to provide a solution based on dynamic sensor placement, using a network of mobile sensors.

The thesis proposes to find a solution for the deployment of sensors in a certain region to obtain reliable data about the respective air quality with reduced uncertainty. To quantify the quality of these measurements, we have built on past work on Bayesian Neural Networks that provides forecasts of pollution levels and their uncertainty, learned from real sampled data. Our work has the multi-objective of maximizing the pollution coverage and minimizing the uncertainty of these measurements in a certain region. The city traffic SUMO simulator is used to extract data about pollution and traffic.

We describe a greedy algorithm for static sensor placement, used as a baseline, and other two algorithms based on mobile sensors: a reactive algorithm and a Reinforcement Learning (RL) algorithm with Proximal Policy Optimization (PPO) architecture. Besides proving the advantages of mobile sensors, we found that using RL we achieve better performance than a reactive algorithm with 1 and 2 sensors.

Keywords

Dynamic sensor deployment; Reinforcement Learning; Pollution monitoring; Decision-making under uncertainty;

Resumo

A poluição do ar é sem dúvida um dos problemas mais preocupantes enfrentados pela sociedade. Bastantes estudos provam que os efeitos desta poluição estão a causar diferentes patologias e outros problemas. Parte das abordagens de mitigação destes problemas consiste em monitorizar as regiões que carecem de mais atenção.

O trabalho pioneiro sobre colocação de sensores usou sensores estáticos. Esta tese propõe uma solução baseada numa instalação de sensores dinâmicos, usando uma rede de sensores móveis.

Esta tese propõe encontrar uma solução para a instalação de sensores numa certa região para obter informação fidedigna no que diz respeito à qualidade do ar, com a menor incerteza possível. Para quantificar a qualidade das medidas, fez-se uso de um trabalho que usando Redes Neuronais Bayesianas fornece previsões sobre os níveis de poluição e a sua incerteza, aprendidas com amostras de dados reais. Este projeto tem o multiobjectivo de maximizar a cobertura de poluição e minimizar a incerteza destas medições numa dada região. O simulador de tráfego de cidades SUMO é usado para extrair informação sobre a poluição e o tráfego.

Descrevemos um algoritmo ganancioso para posicionamento de sensores estáticos, usado como baseline, e os outros dois para sensores móveis: um algoritmo reativo e um de aprendizagem por reforço com uma arquitetura de Proximal Policy Optimization (PPO). Além de se provar as vantagens dos sensores móveis, percebeu-se que usando aprendizagem por reforço conseguimos um melhor desempenho que um algoritmo reativo com 1 e 2 sensores.

Palavras Chave

Instalação dinâmica de sensores; Aprendizagem por reforço; Monitorização de poluição; Tomada de decisão sob incerteza;

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
BNN	Bayesian Neural Network
CA2C	Compound Advantage Actor-Critic
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-network
DQN	Deep Q-network
DRL	Deep Reinforcement Learning
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GCC-MARL	Graph Convolutional Cooperative Multi-agent Reinforcement Learning
GCN	Graph Convolutional Networks
GPU	Graphics Processing Unit
IAC	Independent Actor-Critic
LIIR	Learning Individual Intrinsic Reward
MADDPG	Multi-agent Deep Deterministic Policy Gradient
MARL	Multi-agent Reinforcement Learning
MDP	Markov Decision Process
MLP	Multilayer Perceptron
MRFMR	Weighted Relative Frequency Of Obtaining The Maximal Reward
NN	Neural Network

NOx	Nitrogen Oxides
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NTNU	Norwegian University of Science and Technology
PM	Particulate Matter
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RL	Reinforcement Learning
SARSA	State Action Reward State Action
SUMO	Simulation of Urban MObility
TD	Temporal-Difference
TRPO	Trust Region Policy Optimization

1

Introduction

Contents

The first chapter presents the motivation and background that inspired the realization of this thesis. It is in this chapter that the goals and research questions will be defined. And finally a brief summary about the structure of the report.

1.1 Motivation

The concern about pollution is increasing year after year, and scientists all over the world work to keep our planet a good place for every species to live. It is known the effects that pollution can have in people for instance in the respiratory system [5] or circulatory system [6], [7]. To achieve the ideal goal of reducing pollution, it is crucial to monitor it at fine granularity and with the most detail possible.

Nowadays this monitoring is mainly accomplished by the deployment of sensors in a certain region. However, it is impossible to cover every piece of land. Therefore, we need solutions to approximate the measurements by forecasting pollution in areas without sensors. Also, since static sensors are not optimized to react to variations of these measurements and cannot change position, other solutions using mobile sensors bring new advantages, such as better coverage and decision based on real-time. However, such solutions are more complex and therefore need further study as we do in this thesis, using first a simpler reactive solution and then a solution with Reinforcement Learning (RL).

The problem of sensor placement, whether to measure pollution or other interesting city information has had numerous contributions that tried to solve it, but there is still a lot of work to do in this field. As it is not realistic to have sensors in every piece of land, pollution forecast surges to approximate the information we have about a certain region. It is therefore crucial to ensure that these forecasts are reliable, i.e. to have a way of reducing the uncertainty. There are some contributions regarding the quantification of uncertainty, such as [2] that tries to quantify the uncertainty in each time step for the pollution values forecasts of 24 hours, based on the last 24 hours data, such as traffic, pollution measurements, meteorology, and data from the municipality. Data such as pollution and traffic will come from Simulation of Urban MObility (SUMO) using tools from [1].

There are several studies in the literature that approach sensor deployment, but this thesis seems to be the first to consider this pipeline, using [2] model of uncertainty, and interacting with SUMO simulator. The sensors' deployment is multi-objective in the sense that we try to maximize the pollution coverage and minimize the uncertainty of the measures. Therefore, three algorithms are compared as possible solutions. A sequential order placement greedy algorithm uses static sensors. To deal with mobile sensors we used a Reactive algorithm and one inspired on Proximal Policy Optimization (PPO) architecture with RL, where the sensors can move and are supposed to give more coverage. These sensors are placed in a vehicle that circulates across the city in order to minimize the uncertainty of pollution values.

The final architecture will then be composed by this pipeline including the simulator and data extrac-

tion model, the uncertainty model, and our algorithmic solution that should figure out the best possible sensor placement or autonomous decision for movement in the case of mobile sensors.

1.2 Goals and Research Questions

In this section, multiple goals will be presented, immediately followed by the research questions that made their accomplishment possible.

Goal 1 *Investigate the state of the art in the field of sensor coverage*

Research question 1 *What techniques are being used for sensor placement if the sensors are static?*

Research question 2 *What are the best techniques to handle mobile sensors?*

Research question 3 *What are the pros and cons of static and mobile sensors for monitoring tasks?*

Goal 2 *Explore different algorithms for the placement of static sensors*

Research question 1 *Are there any algorithms in the literature that can provide an optimal or near-optimal solution?*

Research question 2 *Which of the algorithms can be good baselines for the mobile sensors algorithms?*

Goal 3 *Investigate the feasibility of Reinforcement Learning for mobile sensors*

Research question 1 *How competitive is Reinforcement Learning when compared to other algorithms?*

Research question 2 *What are the most adequate models for multi-agent setup within Reinforcement Learning?*

Research question 3 *What are the best models for continuous state space within Reinforcement Learning?*

1.3 Organization of the Document

The report is divided into different chapters to structure the information. The following chapter 2 is proposed to explain the main background concepts that are needed for the solution development. Chapter 3 presents a protocol used to search and evaluate relevant papers in the context of this project. It also has a section for the selection and quality assessment of these papers. After having the most pertinent papers there is another section for related work, where the contributions of these papers are explained in more detail.

The thesis then has a chapter to describe the methodology and experiments. It is divided into sections for different algorithms and the first one describes the pipeline of the proposal.

Finally, there is a section for experiments and a conclusion about the outcomes of the thesis.

2

Background

Contents

In this chapter, different fields of investigation and terminologies will be introduced to familiarize the reader with concepts used during the thesis.

Therefore, the simulator and the uncertainty quantification model used in the thesis will be briefly explained followed by notions of RL. Further, one will get knowledge about the fundamentals of Neural Network (NN) and how to use them in the context of Deep Reinforcement Learning (DRL).

2.1 SUMO Simulator

In this section, the Simulation of Urban Mobility (SUMO) in [8] will be briefly described, since this project uses the simulator as the environment for data collection.

2.1.1 Simulator

It is important to understand the main features of the simulator. SUMO consists of a free open-source project intended to support research on traffic-related studies in a fast, efficient and reliable way. It is inserted in the microscopic category of simulation tools since it models each vehicle independently.

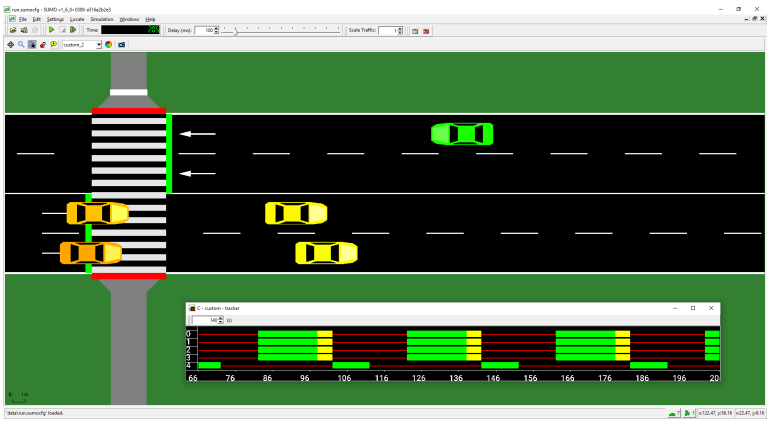


Figure 2.1: SUMO vehicles display (from https://www.eclipse.org/sumo/).

With this tool, we are able to construct scenarios with real-world information in a much faster way since it presents many applications for it. A simulation scenario is constituted by data from roads or footpaths, for instance, information from other infrastructures as lights and traffic demand. It also contemplates pedestrians and multiple vehicles such as cars, trams, taxis, bicycles, etc as seen in Figure 2.1. The simulator networks represent the streets and other paths with unidirectional edges that connect nodes. These edges can have some restriction details, such as direction or the type of vehicles permitted.

Their project has a wide range of tools and includes some applications that support the scenario construction, for example, a python tool called OSM Web Wizard that simplifies the creation of the

scenario with an interface and a map display as observed in Figure 2.2 and Figure 2.1. The tool that supports OSM Web Wizard with the display is SUMO-GUI, an application used for the visualization of the simulation.

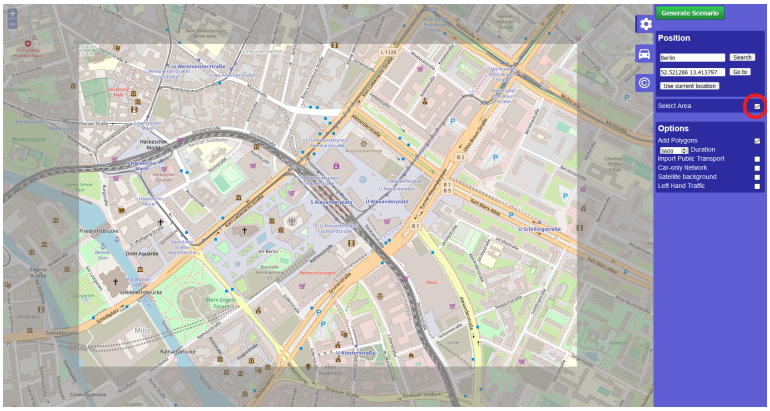


Figure 2.2: OSM Web Wizard (from `https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html`).

SUMO is also able to output some measurements from the simulations, being the most important for this thesis the traffic density and the emissions, which will be described in the next subsection.

2.1.2 Data output

The simulator is able to return the emissions of vehicles as well as a detailed track of the number of vehicles in every region of the simulation, which is the main reason why it is being considered for this thesis. The study carried out by [1] developed some models of data discretization that will integrate the pipeline of this thesis, because the author processed the data in a suitable way to be used in our thesis, which will be discussed in this subsection. The source code of the study was made available, and part of it will be integrated into the pipeline.

The main goal in [1] was to optimize the traffic in a city through the use of Reinforcement Learning. It is not important to go into much detail about the solution for their problem, because we will only use the modules that transform the data to be more suitable to use in our thesis.

The framework developed by [1] is divided into the modules of Figure 2.3. The map was also divided into a grid of cells for a certain region. Therefore data was collected and aggregated per cell. For this project, we are only interested in the Emissions Module that outputs the emissions for each cell. We also implemented a Traffic Module based on code from the last one to output the number of vehicles per cell. The data extracted by these modules will be used to train and test the Bayesian Neural Network (BNN) model and for our algorithms to make decisions based on it.

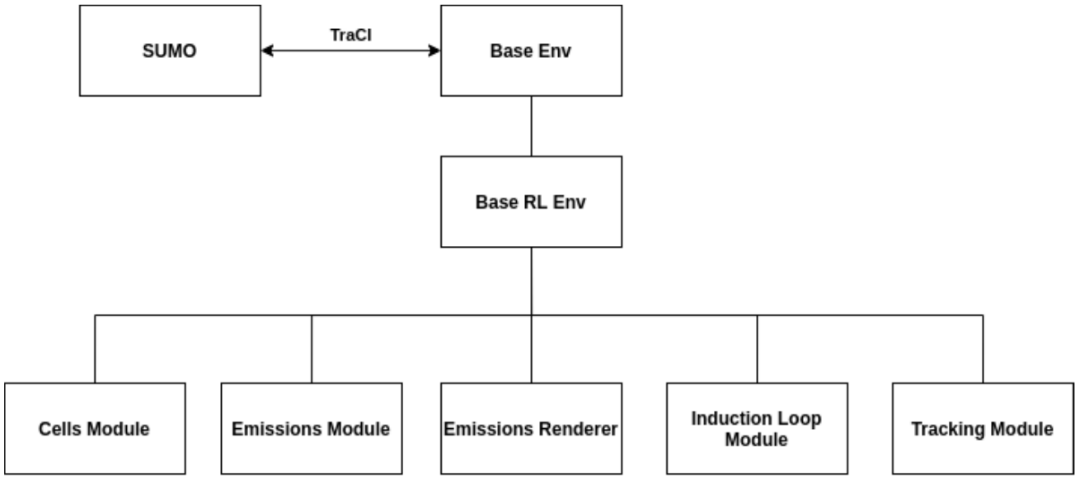


Figure 2.3: Modular structure of the framework to extract traffic and pollution data (from [1]).

There are different types of pollutant agents, but their study was focused on Nitrogen Oxides (NOx), because of the data availability. The downside about the data from the simulator is that it is not discretized for every cell, and therefore not completely suitable for the algorithms used by [1] as well as for this project. However, the author of the mentioned work already implemented this discretization. Also, the emissions that the simulator returned were transformed, since they were instantaneous. Then, Gaussian Blur was used to propagate the pollution to the neighboring cells, using the formula in (2.1) to apply a decay to each cell, being σ the dissipation and x and y the relative coordinates of the cell, leading to the assignment of more realistic values over time.

$$G(x,y) = \frac{1}{2\pi\sigma} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

2.2 Uncertainty in Air Quality Forecasting

It is possible to have an environment with pollution emissions for every cell of a grid. However, we can not forget that sometimes these values come from forecasts are not completely reliable. An approach with forecasts will lead to some uncertainty regarding those values. That is the main reason why this section will be focused on the work carried out by [2] that compares different methods for this quantification.

Their study proposed an improvement in state-of-art models for uncertainty quantification of air quality forecasts. The forecast models took as input data from air quality, traffic, meteorology, and reports about street cleaning. For the case of BNN, the model used in the thesis, it outputs a Gaussian, that is used to infer the forecast, being the mean value. They also described an upper bound and lower bound as being the sum or subtraction of the mean by the standard deviation multiplied by a z-score for 95%.

This quantification was made for more than one type of uncertainty. The epistemic uncertainty con-

sidered data-sparse regions, i.e. events that occur occasionally. The aleatoric uncertainty comes from observations noise, and finally the uncertainty regarding the model structure (e.g. number of layers and neurons) that is not perfect.

For the forecasts, they used the values of the mentioned data from 24 hours before to predict the future value of the air pollutant used, in this case, Particulate Matter (PM).

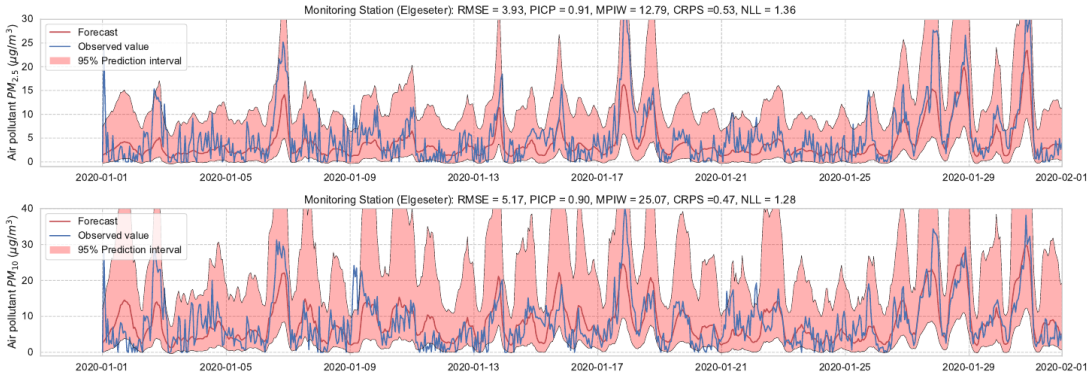


Figure 2.4: Forecast of air quality time-series with BNNs with uncertainty interval (from [2]).

The forecast illustrated by Figure 2.4 shows the results of using Bayesian Neural Networks (BNNs) trained with data from an entire year and predicting for one month. The prediction interval gives us the top and bottom bounds for PM-value.

The study compared the tradeoffs of using BNNs or other models such as Deep Ensemble, MC Dropout, or SWAG. It was concluded that we can get more reliable estimates about uncertainty with BNNs, however, the other approaches showed to be more practically convenient, for instance for large datasets.

This thesis will integrate the BNNs model into the pipeline since we need the most reliable estimates and the dataset is not large enough to need the other methods mentioned. The forecast model will have to be adapted to receive as input the traffic outputted by the aforementioned sources regarding the SUMO simulator. Then, we will be able to get the forecast of pollution and uncertainty of the values.

2.3 Markov Decision Processes

In [9] it is given a brief summary about Markov Decision Process (MDP), that will further be solved using RL algorithms. This model is represented by the tuple (S, A, P, R), where S is the set of states of the system, A the set of actions that can be taken in each state, P the transition probabilities matrix and R the set of rewards when a certain action is taken in a state. This model benefits from the fact that we can choose what action to take only with the current state, ignoring all the past information as represented in (2.2).

$$P(s_{t+1}|s_{0:t}, a_{0:t}) = P(s_{t+1}|s_t, a_t) \tag{2.2}$$

where s_t and a_t are the state and action respectively at time step t.

The mapping of each state to the possible actions is represented by a policy, and many algorithms try to maximize the expected rewards following this policy, being the optimal policy the one with the maximum accumulated discounted expected reward,

$$\mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... + \gamma^n r_{t+n}] \tag{2.3}$$

where r_t is the reward at each time step t and γ the discount at each time step.

As described in [10] the state value function v_π can be stated in (2.4) as the value of following the policy π when in state s. The called action value function q_π in (2.5) represents the return of choosing action a in state s following the policy π, being for both equations γ the discount rate and R the reward at time step t.

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \text{for all } s \in S \tag{2.4}$$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right], \text{for all } s \in S \text{ and } a \in A \tag{2.5}$$

2.4 Reinforcement Learning

In this section, RL will be addressed for Single-Agent and Multi-Agent mainly based on [10].

Starting by explaining the need for RL, we must consider an environment with one or more agents, where each agent interacts with the environment and has to take a specific action at each time step that may modify directly the same environment. This is often described through MDP described in the last section, combining states, actions, and rewards and where the probability of going to a certain state only depends on the current state, discarding information from the past. Then, RL solves an MDP without information about the transition probabilities and the reward known a priori to maximize the accumulated discounted expected reward (2.3).

This interaction with the environment makes use of both exploitation and exploration, being the first one a rule to choose actions based on the current policy and therefore with the information we already have. However, and since we can collect relevant information from states where the agent has not been before, arises the need for exploration that explores the unknown seeking different paths prioritizing future rewards.

2.4.1 Temporal-Difference Learning

There are methods that use Temporal-Difference (TD) to achieve the desired interaction with the environment and learn an optimal action-value function. TD learning contemplates a set of methods that are able to use bootstrap in their updates, i.e. instead of updating the value function at the end of an episode like Monte Carlo methodology, they use the estimates of every time step individually to update the value function as following

$$V(s_t) \leftarrow V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2.6}$$

where $V(s_t)$ is the estimate from state s_t at time step t, R_{t+1} the reward, α the learning rate and γ a discount factor. This gives TD learning methods the advantage of being online and incremental, not needing to wait for the end of the episode. Also, they tend to converge earlier than Monte Carlo methods.

It is also needed to differentiate on-policy from off-policy methods to account for exploitation and exploration.

On-policy TD Control

The first that will be explained is on-policy TD learning, where the action-value function Q is learnt directly from the current policy. The update is similar to the one in (2.6), and the update rule for State Action Reward State Action (SARSA), the most well-known on-policy method is presented by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, s_{t+1}) - Q(s_t, a_t)] \tag{2.7}$$

where $Q(s_t, a_t)$ is the estimate of how good is to take an action a_t from state s_t at time step t, α the learning rate, i.e. the relevance that each update will have, and γ a discount factor.

Off-policy TD Control

Unlike on-policy TD-learning, to achieve the optimal action-value function, instead of using the policy directly, the updates are independent of the policy and come from the action-value function Q. A greedy policy is used to get the reward estimation of choosing an action in a certain state. The following update rule shows a small but crucial difference from SARSA and comes from Q-learning, a very popular algorithm [9].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.8}$$

It is easy to understand that Q-learning takes more exploratory actions than SARSA, and is less

conservative. Then, it is important to measure this trade-off and choose the best option for each case.

Exploration vs Exploitation

As explained before, with TD methods, it is needed to trade off exploration and exploitation. Therefore, it is possible to find several algorithms that deal with this trade-off [11]. One of them is the simple ϵ-greedy that assign a probability for choosing the action having into account the current policy and another probability for choosing a random action. If it is considered its variation, decaying-ϵ-greedy, the probability of choosing a random value reduces over time, since the policy represents the environment better with more training.

2.5 Neural Networks

This section will cover some relevant concepts regarding neural networks, starting with the most basic and general models and going deeper into some more detailed. Most of these notions and figures are covered in [3].

2.5.1 Perceptron

Before going into detail about neural networks, it is meaningful to understand how the perceptron works. The simplest model receives an input $x_1, x_2, ..., x_n$ and outputs a value as shown in Figure 2.5.

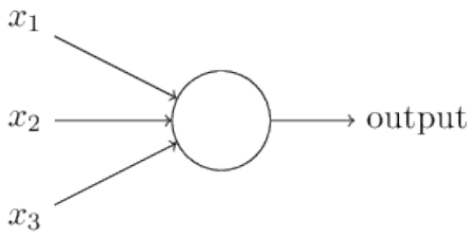


Figure 2.5: Perceptron (from [3]).

However, as not every feature of the input has the same importance, an improved model would receive as input $x_1, x_2, ..., x_n$ with respective weights $w_1, w_2, ..., w_n$ it outputs a value 0 if $\sum_{i=1}^{n} w_i x_i \leq threshold$ and outputs 1 if $\sum_{i=1}^{n} w_i x_i > threshold$. It is trivial to realize that such a solution is limited and not suitable for many of the decision problems we want to solve. Then, it is used to build more complete neural networks described below.

2.5.2 Multilayer Perceptron

As explained before, more capable models were needed. Multilayer Perceptron (MLP), is a neural network composed of multiple neurons. These models start with an input layer, followed by a number of hidden layers, and end with an output layer that can also contain one or several neurons as presented in Figure 2.6.

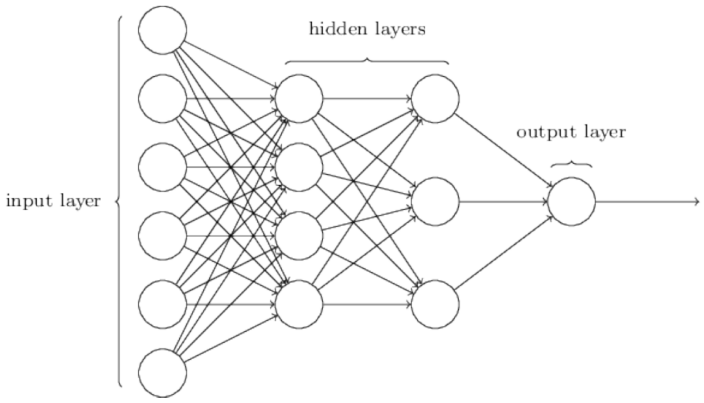


Figure 2.6: Multilayer Perceptron (from [3]).

These models are also called Feedforward Neural Networks because the information always goes forward until it reaches the output layer. But if it was just this, the neural network would never learn and therefore would never get better results. Then, it needs to update the weights of its layers according to the results. This step is called backpropagation and uses a loss function to calculate the error of the estimation. There are multiple loss functions, and one of the most commons is the mean square error, shown below.

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{2.9}$$

with N being the number of samples, $\hat{y}$ the prediction and y the real value.

The backpropagation step consists in computing the gradient regarding this error. The formulas used for this algorithm follow the chain rule derivation for each layer. A parameter called learning rate is also included to control the magnitude of each update. Although a bigger learning rate can fast the training process, it can also lead the model to a solution that is not optimal.

2.6 Deep Reinforcement Learning

Reinforcement Learning allows us to solve problems where an agent had to interact with the environment. However, it has to deal with the curse of dimensionality, and high-dimensional state spaces

became an obstacle. When Deep Reinforcement Learning appeared [12], it was used as a possible solution for these problems, since it used deep neural networks as the model to be trained and then deal with the complexity of the environment. [12] considered a new algorithm that used a deep convolutional neural network. It first appeared to solve an Atari game that was highly dimensional.

2.6.1 Policy Gradient Methods

Policy gradient methods [10] are described as algorithms that learn the policy without needing the update the value function. They are known to be effective for high dimensionalities, such as continuous action spaces, and have more convergence guarantees than TD-learning.

One of the most known policy gradient methods is REINFORCE that also tries to approximate the gradient, and it uses the following update

$$\theta_{t+1} \leftarrow \theta_t + \alpha\gamma^t R \nabla \ln \pi(a_t|s_t, \theta_t) \tag{2.10}$$

where s_t and a_t are the state and action taken at time step t, and R the total discounted reward.

2.6.2 Actor-Critic Models

As described before in subsection 2.4.1, TD-learning methods are very appealing since they use information based on experience incrementally at each step to update the policy, and then select the best actions according to it. Other algorithms that use policy gradients, described in 2.6.1 do not need the value function to select the actions.

Actor-critic [10] emerges to use both the advantages of action-value methods and policy-based ones. Then the actor is responsible for learning the policy and the critic responsible for learning the value function as depicted in Figure 2.7.

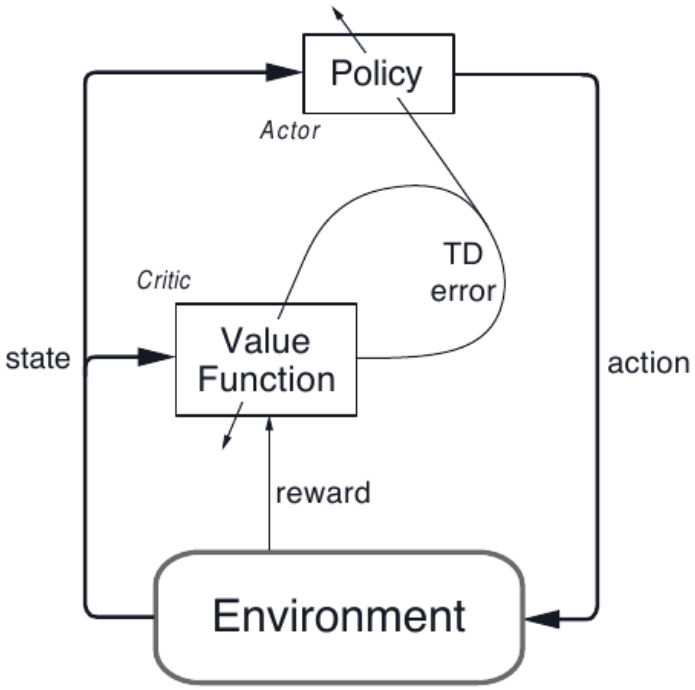


Figure 2.7: Actor-critic (from [4])

Therefore, we can use the state-value function as a baseline and apply the update in (2.11).

$$\theta_{t+1} \leftarrow \theta_t + \alpha(r_{t+1} + \gamma\hat{v}(s_{t+1}) - \hat{v}(s_t))\nabla \ln \pi(a_t|s_t, \theta_t) \tag{2.11}$$

where r_{t+1} is the one-step reward, $\hat{v}(s_t)$ is an estimation of the state-value function, γ the discount, α the learning rate, and s_t and a_t are the state and action taken at time step t. It is often described as (2.12).

$$\theta_{t+1} \leftarrow \theta_t + \alpha A(s_t, a_t)\nabla \ln \pi(a_t|s_t, \theta_t) \tag{2.12}$$

$$\begin{aligned} A(s_t, a_t) &= r_{t+1} + \gamma\hat{v}(s_{t+1}) - \hat{v}(s_t) \\ &= Q(s_t, a_t) - \hat{v}(s_t) \end{aligned} \tag{2.13}$$

where $A(s_t, a_t)$ is the advantage function and represents how valuable it is to choose a certain action a_t in the state s_t. The expression can be simplified to the state-action value $Q(s_t, a_t)$.

When training these models, some updates in the policy are so big that can lead to unexpected results in the performance. Some algorithms such as Trust Region Policy Optimization (TRPO) [13] limit the range of these policy updates. PPO [14] is an on-policy algorithm that uses the same principle as TRPO, however with a simpler algorithm that empirically achieves the same results. It uses clipped

surrogate objective (2.14), being $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ the probability ratio of an action according to the current policy, $\hat{A}_t$ the advantage function and ϵ a hyperparameter, used to clip the probability ratio between $1+\epsilon$ and $1-\epsilon$.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.14}$$

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta]|(s_t)] \tag{2.15}$$

The loss function (2.15) can also take into account the shared parameters between the policy and the value function with $L^{VF}(\theta)$ being the squared-error loss, an entropy bonus $S[\pi_\theta]|(s_t)$ to provide enough exploration and $c1$ and $c2$ coefficients.

With PPO we can also take advantage of parallel actors to collect data used afterwards to create the surrogate loss.

2.7 Multi-Agent Actor-Critic

Until now, for every notion of Reinforcement Learning, it was considered the case of single-agent architecture. However, sometimes we need to tackle the problem of having more than one agent interacting with the environment. A simple approach would be to continue considering only one agent and describe its state as the set of states of every action. Also, the actions state would also increase to consider an action for each agent. It is easy to understand the limitation of this process and the huge complexity that could come from this implementation.

To overcome this dimensionality issue, when the number of agents is too big, or if the overall complexity of the environment is not supported by a central system, it is needed to develop each agent with its own decision-making process. Baring in mind that there are cooperative and competitive multi-agents, only the first will be covered.

Thus, if every agent makes its own decisions, multiple algorithms have surged for multi-agent [15], and some concepts are common to most of them. One of these concepts is the notion of joint action that combines all the actions of each agent. The other is the reward, of taking a certain action, because now it is needed to consider the actions of the other agents.

An actor-critic architecture will be used in this thesis. However, although the architecture of a multi-agent actor-critic has its advantages, it also loses information. That is why our agents learn independently from each other in our algorithm. In order not to lose the information about the actions of other agents, every agent takes its actions sequentially taking into account the others.

3

Structured Literature Review

Contents

The objective of this chapter was to perform a literature review, structured to find the relevant papers for this project. These papers should answer the research questions of the section 1.2.

The protocol used for a Structured Literature Review is defined here. However, it is important to point out that this protocol was inspired by the work of [16].

Having that said, we will start by identifying the research and defining the databases used for it. A selection of primary studies will be done based only on the search options and not on the content of the study. The third step is to define the search queries and present the results of that researches. However, we will still have a lot of results, so it is crucial to filter them according to inclusion and quality criteria also detailed in the proper section.

The last section and the one with the most value for the thesis is the related work, where the contributions of each study will be explained.

3.1 Identification of Research

This section covers the challenging job of finding relevant papers regarding the research questions and the ones that would be useful for the solution development.

Therefore, a strategy should be implemented to define the list of computer science archives and search platforms, as well as the search string structure.

The first problem that is addressed is the list of search platforms chosen to find the papers. As most of them are already well-known, the choice was straightforward and based on these currently used platforms. However, it is important to explain, that Google Scholar was used to try to catch other relevant papers that were not in these databases. Then, the following sources are used:

- IEEE Xplore
- SpringerLink
- ACM digital library
- Google Scholar

3.2 Selection of Primary Studies

The search described in the last section will obviously output a huge amount of results. Yet, many of these papers could be discarded, since not all of them have the same relevance for the problem and the study should focus the effort on the most pertinent ones.

To achieve this goal, a protocol was defined to evaluate the relevance of the papers. Then, the selection process applied the following points to filter and distinguish among papers:

- If the same study is found twice keep only one without specific criteria, since all the used platforms are reliable.
- Keep the most recent version of the paper
- (Applied in a second search for finer granularity) Keep only the paper published in or after 2018
- Keep only the first 100 papers ordered by relevance (if the search contains more than 100 results)

It was decided to keep the first 100 papers if the search contains more than 100 results and discard the others because the platforms used to order the results by the relevance of the matching terms in the paper. Therefore, for a big output, the first 100 papers will be the most relevant ones.

3.3 Research Results

In this section the search queries used will be explained, giving a justification for each term. Then the results of the search will be presented and the relevant papers chosen before going to the quality assessment of them.

The search queries are as follows:

Search 1: ”sensor placement” AND optimization AND (pollution OR uncertainty)

The reasons that led to the choice of each search term was the following:

- **”sensor placement”:** it is a known term in the literature to define the research
- **”optimization”:** we want to find an optimal or at least near optimal solution
- **”pollution OR uncertainty”:** the two main topics of the research

After defining the query, the databases used got the following results:

Database	Results
IEEE Xplore	43
SpringerLink	100+
ACM Digital Library	42
Google Scholar	100+

Table 3.1: Total results for search query 1.

Search 2: ”sensor network” AND ”reinforcement learning” AND multi-agent AND cooperative

The reasons that led to the choice of each search term was the following:

- **”sensor network”:** straightforward and owes to the fact that we are interested in finding a network of sensors that monitor the pollution
- **”reinforcement learning”:** the chosen field to approach moving sensors that learn a policy
- **”multi-agent”:** to know how to work with multiple sensors
- **”cooperative”:** to differentiate between cooperative and competitive multi-agent models

After defining the query, the databases used got the following results:

Database	Results
IEEE Xplore	35
SpringerLink	100+
ACM Digital Library	33
Google Scholar	100+

Table 3.2: Total results for search query 2.

Some researches were made in Google Scholar to compare some models and to find a reason for their implementation. Some of them did not mean the same search query protocol as the last two queries and instead used more than one query covering the terms of the topic in research. These searches are presented in the following list:

- **Search:** ”centralized critic” AND ”decentralized critics”
- Multiple searches considering reinforcement learning algorithms
- Multiple searches considering actor-critic models
- Multiple searches considering multi-agents

3.4 Quality Assessment

Now that we have a set of relevant papers for the problem, more detailed filters will be applied to exclude some studies that despite being relevant, are more relevant than others. First, it was defined some inclusion criteria to differentiate between primary and secondary studies and quality criteria to define the importance that should be given to each of them.

3.4.1 Inclusion Criteria

As this search was already based on some filtering options defined in the last sections, to chose the relevant papers among all the results obtained in the search it was used the following considerations about the papers:

- Title
- Abstract
- Conclusion and discussion

To filter the best studies it was defined some inclusion criteria. It is important to explain that the criteria vary from research question. The criteria used for this division is the following:

Criteria for research for static sensors The study covers static sensor placement with a solution at least near optimal

Criteria for research for mobile sensors The study covers mobile sensors with a solution at least near optimal

Then, after applying this filtering we will consider fewer studies as primary for our project, and they are presented on the following tables.

Database	Results
IEEE Xplore	5
SpringerLink	7
ACM Digital Library	2
Google Scholar	1

Table 3.3: Total results after filtering for search query 1.

Database	Results
IEEE Xplore	6
SpringerLink	0
ACM Digital Library	1
Google Scholar	0

Table 3.4: Total results after filtering for search query 2.

3.4.2 Quality Criteria

The last quality assessment of the papers is more detailed and is based on a protocol with 7 criteria questions to which it was given punctuations: 1 (completely covered), 0.5 (partially covered), and 0 (not covered). This protocol makes it able to know the importance that should be given to each study.

Criteria 1 The study covers the research objective

Criteria 2 The study presents solutions directly related to air pollution or forecast uncertainty

Criteria 3 There is evidence that supports the choice of the given algorithm(s)

Criteria 4 The solution is reproducible

Criteria 5 The solution was properly experimented against good performance assessments

Criteria 6 There are strong analysis and discussion comparing the study to other solutions

Criteria 7 There is a direct solution or a solution that with improvements can be applied to the project

Now that both the protocol and quality questions are defined, each of the studies will be given the respective punctuation as shown in the following tables.

Study	C1	C2	C3	C4	C5	C6	C7	Total/7
[17]	0.5	1	0.5	1	0.5	0.5	0.5	4.5
[18]	0.5	1	0.5	1	0.5	0.5	0.5	4.5
[19]	0.5	0.5	1	1	0.5	0.5	0	4
[20]	0	1	0	0.5	0	0	0	1.5
[21]	0.5	0.5	1	1	1	0.5	0.5	5
[22]	0.5	1	1	1	0.5	1	0	5
[23]	1	NA	NA	NA	NA	NA	NA	NA
[24]	0.5	0	1	1	0.5	1	0.5	4.5
[25]	1	0	1	1	1	0.5	1	5.5
[26]	0.5	0	1	0.5	1	1	0.5	4.5
[27]	0.5	0.5	1	1	1	0.5	0	4.5
[28]	0.5	0.5	1	1	1	0.5	0	4.5
[29]	1	0	1	1	1	1	1	6
[30]	1	0	1	1	1	0	0.5	4.5
[31]	1	0	1	1	1	0.5	1	5.5

Table 3.5: Punctuation for papers of search query 1.

Study	C1	C2	C3	C4	C5	C6	C7	Total/7
[32]	0.5	0	1	1	1	0.5	0.5	4.5
[33]	0	0.5	1	1	1	1	1	6
[15]	1	NA	NA	NA	NA	NA	NA	NA
[34]	1	0	1	1	1	1	0.5	5.5
[35]	1	0	1	1	1	1	0.5	5.5
[36]	0.5	0	1	1	1	1	0.5	5
[37]	0.5	0	0.5	0	0	0	0	1

Table 3.6: Punctuation for papers of search query 2.

Some other relevant papers were found without following the search queries protocol and happen that some of them actually provided good insights for mobile sensors. The following table shows how we classify such papers.

Study	C1	C2	C3	C4	C5	C6	C7	Total/7
[38]	1	NA	NA	NA	NA	NA	NA	NA
[39]	1	0	1	1	1	1	0.5	5.5
[40]	0.5	0	1	1	1	0.5	1	5.5
[41]	0.5	0	1	1	1	0.5	0.5	4.5
[42]	0.5	0	1	1	1	1	1	5.5
[43]	1	NA	NA	NA	NA	NA	NA	NA
[44]	1	0	1	1	1	1	1	6
[45]	1	0	1	1	1	1	1	6

Table 3.7: Punctuation for some relevant papers relate to mobile sensors that did not follow the search query.

Apart from the studies selected to be classified in the above tables, other studies were taken into consideration for being recommended by the supervisor or people in the community. Others were in references or were also gathered in an early stage of the project when the protocol was not defined and therefore the rules can not be applied to them. All in all, they were discussed before or will be discussed in the related work.

3.5 Related Work

This section will describe in more detail the conclusions and discussions about each paper and how they can provide some knowledge required to build the solution.

3.5.1 Static sensors

Sensor placement has been studied for a long time, and it is visible by the number of papers that we were able to find. Some related to water pollution (e.g. [19]), others about structural health issues [29], and some also covered the problem of air pollution (e.g. [17]). Nonetheless, many of them contributed with good insights into the problem aforementioned.

A solution for air pollution is in the interest of the population, and there have been performed case studies in places where this implementation would be important. For example in the city of Cambridge [17], Hong Kong [18], the city of Modena [19], and China [20]. This project will also test the solution in Trondheim, a city in Norway.

In [17] it is referred that static sensors can be expensive, which also introduces the topic of low-cost mobile sensors that will be explored later. Their first greedy algorithm chooses sensor by sensor the one that provides the highest reward, with the second one considering also a general cost constraint. Once more in [18], it was also proposed two greedy algorithms (one of them also considered the cost of maintenance, construction and operations such as sampling or calibration) for the fixed-location sensors that would choose the location sensor by sensor according to the reward, but using weights. Other so-

lutions for these types of greedy algorithms also considered a division into batches [21], i.e. maximizing the gain for smaller subsets of sensors sequentially placed. [28] proposed a hierarchical algorithm not for pollution, but leak detection. However they used the same sequential selection of sensors but having into account the variation of joint entropy when a sensor fails. Another recent paper [30] approaches indoor crowdsensing, crucial in times with COVID-19 optimizing the location of sensors with sequential placement using signal strength Gaussian distributions.

The study done by [46] models the data with Gaussian Processes and explores how to place sensors in order to maximize mutual information, which is proved to have advantages when compared to other design criteria. Further, they also considered node failures and uncertainty. Although the problem is NP-hard, mutual information benefits from submodularity and their greedy algorithm shows to achieve a near-optimal solution with an approximation guarantee from the optimal solution. Taking advantage of submodularity they can also bound the difference between their results and the optimal solution, and search for tighter bounds. To make the process more efficient they use a lazy algorithm and local kernels.

A survey [23] compared greedy algorithms already presented with others for sensor placement, for instance the mixed-integer programming and a Genetic Algorithm (GA). In the paper, they also refer that deterministic algorithms can complement GA to improve the search for optimal solution and have better scalability performance. It can be also noticed that mobile sensors are referred to as future and more optimized solutions, as we will talk about later. [22] used GA to detect a sudden release of a chemical in a ventilation system, having into consideration the time needed for the detection, the exposure, and probability of events not detected. [24] used GA for structural health monitoring. As it is known, Evolutionary Algorithm (EA), being the most common GA do not guarantee a theoretically optimal solution, but in practice, they find almost optimal solutions, which was the case in the experiments of [25] and explained in [23].

A recent study on structural health monitoring tried to improve the problem of slow convergence of GA using neural networks that would update the set of variables used in GA [29]. They achieved the optimal locations for the problem with much fewer generations.

[19] have into consideration not only the sensing but also the uncertainty and noise that can come from the measurements, trying to minimize the leakage detection error and the leakage localization error in water distributed networks. As it was a multi-objective problem, Non-dominated Sorting Genetic Algorithm II (NSGA-II) was used. NSGA-II consists of a fast algorithm for finding the optimal solution in multi-objective domains. As described in [47] NSGA-II uses domination to rank the population according to different fronts, i.e. an individual dominates another if it is fittest in at least one variable of the multi-objective, and the other variables have equal or better values. Then, after eliminating the worst fronts, crowding distance is then used to delete the elements of the worst of the remained fronts and maintain

the size of the population. [31] also adopted NSGA-II for vibration detection to minimize the number of sensors placed and maximize the amount of information collected. In the study, they used elitism to preserve the best candidates of the population to be present in the next generation.

3.5.2 Mobile sensors

It was noticed in the literature that some would refer to mobile sensors as a solution to have a better spatial-temporal coverage of some region that is being monitored. That was the main reason why the focus of the this section was on this type of approach.

When the space states are huge, for instance when it is continuous, studies done by [12] in Atari have proved that the use of Deep Reinforcement Learning (DRL) can provide approximations that alleviate the curse of dimensionality.

The number of agents, for the problem vehicles with sensors, could be dealt with single-agent architecture and centralized decision-making if we were considering only one or two agents. However, multiple vehicles could be used to have finer granularity measurements and therefore a better model. As for the matter, when all the agents are homogeneous it is easier to scale the system with a multi-agent architecture [15]. One of the most promising studies is related to Markov decision processes and uses reinforcement learning. This owes to the fact that with Multi-agent Reinforcement Learning (MARL), it is possible to achieve speedup improvements, especially if there is decentralized learning of the policy. We can achieve significant improvements dealing with the curse of dimensionality and if one agent fails, the others can substitute the work that was being performed [38]. One approach to deal with MARL was proposed by [32] with the use of a Weighted Relative Frequency Of Obtaining The Maximal Reward (MRFMR), an algorithm with the main contribution of balancing exploration and exploitation in MARL. It was proved to converge faster. Each agent needs to observe the rewards of the other agents, however, it does not need to observe the actions, which is the second contribution of the paper. The article mentions also other algorithms in the category of independent learners that have been overcome by this algorithm. Not mentioned in this article, a paper with a learning automata-based MARL was found [35], and it also outperformed most of the algorithms mentioned in the last article.

During the research, and having now into account the monitoring task, a lot of the papers found were about crowdsensing that can be associated with the monitoring of variables such as pollution for example. Used for crowdsensing with mobile sensors, a solution for networks of vehicles was explored in [39]. They used deep reinforcement learning with a Deep Q-network (DQN) approach to select the best vehicles to achieve the maximum coverage in space and time. Their environment was divided into regions of interest, which can be also seen as cells in a grid. In this study, the agent was responsible for all the vehicles and selected the ones that would provide the best coverage. [40] was conducted based on task allocation using a Double Deep Q-network (DDQN) solution, where the mobile sensors would

be requested to do some tasks having into account the time limit for performing such tasks. The agent would be assigned a reward for each assignment. and each assignment would be an action. This can be interesting for monitoring tasks since the actions would not be based only on the next road segment or the neighbor cell. However, this simplicity can also have the cost of less detailed decision-making, and here is the tradeoff.

[41] proposes an asynchronous solution for crowdsensing since they use only one agent and the training is computationally expensive. Their algorithm is IMPALA based and uses multiple actors each with its Central Processing Unit (CPU) and one learner Graphics Processing Unit (GPU), achieving better speed than the normal CPU implementation.

Some explored the monitoring problem having into account that vehicles have limited energy capacities. [42] uses Convolutional Neural Network (CNN) to extract the features, but it chooses to use Deep Deterministic Policy Gradient (DDPG), which they refer to overcome DQN that is limited to small action spaces.

Other frameworks, such as [34] also consider limitations such as a possible restrictive communication between the mobile sensors using a Multi-agent Deep Deterministic Policy Gradient (MADDPG) to enable each agent to achieve dynamic coverage without losing network connectivity. The algorithm also uses centralizing training and decentralized execution.

The synchronous advantage actor-critic is also known as Advantage Actor-Critic (A2C), and then, [36] used a Compound Advantage Actor-Critic (CA2C) for sensing tasks with the objective of minimizing the age of information, by learning the optimal trajectories of unmanned aerial vehicles. It was proved to outperform DDPG.

The work done by [33] collected data regarding the taxis of Shenzhen in China and used a Graph Convolutional Cooperative Multi-agent Reinforcement Learning (GCC-MARL) inspired on Graph Convolutional Networks (GCN) to change the route of these taxis or other for-hire vehicles without having a negative impact on the orders of the clients. There is also a filtration on the noises regarding noise from the long-term reward. The decision of using a multi-agent model owes to the huge number of vehicles used. Instead of considering a grid of cells with locations with more granularity, as will be decided further in the architecture design, they considered segments of the roads and that was the main reason for the graph. In our case, it would make more sense to use a grid instead of a graph. They define the environment as partially observable, using the Markov decision model, in this case, a multi-agent reinforcement learning model with actor-critic. There is a centralized training and decentralized execution, i.e. a central critic and an actor for each agent that takes action independently of the other agents. Their algorithm shows to surpass other types of algorithms such as Independent Actor-Critic (IAC) [48] with decentralized critic, Central-V that does not consider the influence of every agent in the global reward [49], and Learning Individual Intrinsic Reward (LIIR) that considers that every agent has an intrinsic reward [50].

Other modern studies in the literature used PPO or variations of it to learn policies for monitoring tasks and routing vehicles. [44] used PPO to move robots in order to monitor changes in the environment. The communication was done by sharing parameters between agents through the use of Graph Attention methods. As they were interested in a persistent coverage setup, a discount was applied for each area without a sensor at a specific time step. In the context of the aforementioned water monitoring, [45] made use of PPO to cover information in water reservoirs while minimizing the uncertainty of measurements and avoiding obstacles.

A recent study carried out by [43] discussed the fact that the current literature often uses a central critic and decentralized actors, i.e. centralized training and decentralized execution for its popularity in the community. However, they demystify the pros and cons of this framework and the trade-off between stable value functions compared to policies. They analyze both the variance and the expected gradient and tested for many domains, reaching the conclusion that decentralized critic tends to have better performance than a centralized one for some domains. One of the reasons that lead to the use of centralized critic was the fact that it would be discarded after training. However, on one hand, decentralized proved to retain more information, leading to more policy robustness due to less variance in its updates. On the other hand, decentralized critics would create a higher bias and instability in the Q-values updates during the training. Then this tradeoff should be taken into account.

4

Methodology

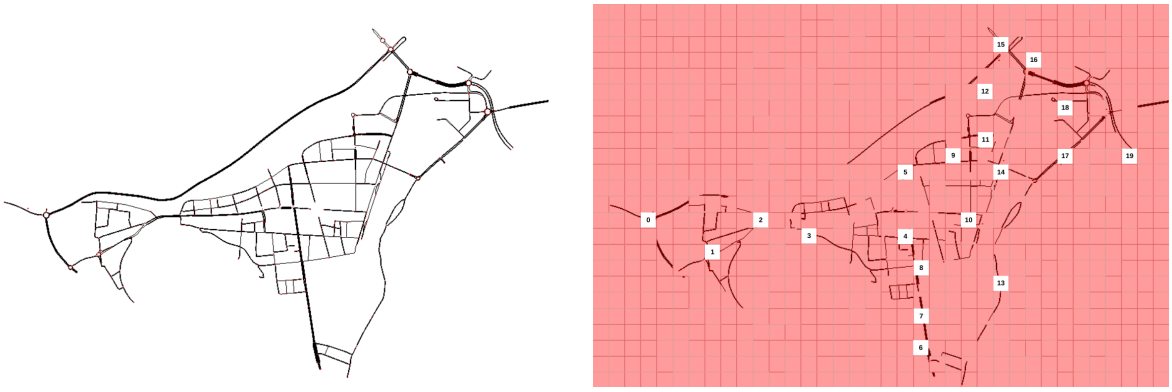
Contents

This chapter will present different architectures for the solution. The algorithms chosen enable us to address both the differences in performance between static and mobile sensors and the differences between an algorithm with a reactive behavior based only on the last state with an algorithm with RL that takes into consideration the history of observations and makes sequential decisions to maximize the cumulative reward of these decisions. Then, by comparing the results of these algorithms in the next chapters we expect that mobile sensors can outperform by a great margin static sensors in the task of covering the areas that we give the most relevance. We also want to discuss in which situations we take the most advantage of using a reactive agent or RL. The experiments include the importance that we give to the pollution measurements and the uncertainty of these measurements. We will start by explaining the environment and how to simplify it in order to apply the algorithms to it. To better understand the architecture of the pipeline some notions will be given about the environment and interaction with it. Then, all the algorithms will be covered. The first one will approach static sensors, with a greedy algorithm, and the last two will work for mobile sensors using a reactive algorithm and a PPO architecture with reinforcement learning.

4.1 Environment Design & Interaction

The first note to make is about the programming language that will be used and Python [51] was chosen to implement all the algorithms since it is a high-level language that is able to provide all the tools and frameworks needed. The main library used for the Reinforcement Learning algorithm was RLlib [52], since it had already a lot of features implemented in this area. The problem of sensor placement has been considered for numerous studies and for the majority of them it was needed to make some simplifications on the environment to reduce the computational complexity.

We can enumerate different simplifications, such as encompassing different points of a certain region in cells with a bigger size. For that the value of a certain variable would have to be simplified as well, being the mean to represent the entire region.



(a) Original map **(b)** Simplification

Figure 4.1: Simplification of the map using a grid.

A grid is defined to represent the map of the region we want to study and where we perform the sensor deployment. The grid is composed of cells as shown in Figure 4.1(b), each one with a value corresponding to the amount of the total emissions of every vehicle in that region, the number of vehicles moving, the forecast of the pollution and the uncertainty about the forecast of the pollution or an uncertainty associated to the last pollution measurement, and a sensor if it is the case. To reduce the complexity and runtime of the experiments we only used 20 cells according to their location in regions of interest.

Both the pollution and the traffic measurements were taken every second for each cell. We considered a time step length of 10 minutes and the values that we use for each time step are the averages of the pollution and the traffic measurements during the 10 minutes. It is common to use induction loops to get the number of vehicles that pass on a certain road during a certain period. However, the simulator enabled us to measure the number of vehicles that were moving and contributing to the pollution of the entire cell each second. In this way, we could get more precise values of the traffic density. We then used an average, because if, for instance, a car is moving in the current second it would still be in movement in the next second, so the sum would not be appropriate.

Since the values present different magnitudes, they are normalized following the simple linear normalization (4.1).

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

where x' is the new value after normalization of x and the $min(x)$ and $max(x)$ represent the minimum and maximum value that each variable can have, and these bounds are fixed.

This grid has a different number of cells and each of them has different sizes according to the experiments that are performed. It is important to understand that this type of discretization leads to the loss

of information. However, the finer the granularity, the more complex would be the computations. So, it is discretized for simplification.

Figure 4.2 represents every component of the thesis pipeline and the flows between them. The sensors can be static or mobile, so the diagram in Figure 4.2 represents them as a black box. It is however important to have into consideration that the mobile sensors are placed into vehicles and in that case, the roads were discretized and also considered as a black box. Instead of reading segments used for instance in [33], the cells will represent both the departure and arrival points.

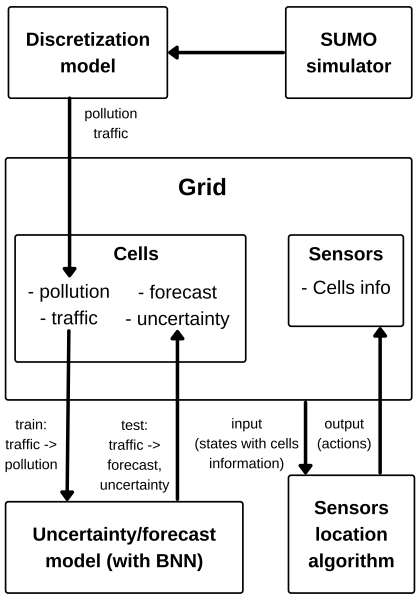


Figure 4.2: Environment scheme .

It is important to explain that our pipeline works with both the SUMO simulator, the discretization of data models from [1] and the forecasts and uncertainty provided by the BNN model [2]. Although the pipeline incorporates different elements to work as a whole, each one of these elements, such as the simulator, the uncertainty model, and the algorithm can be substituted for different options if required, since they are independent of each other.

As the interaction with the SUMO simulator can take a long period of time, to speed up the process of training and testing, we only used the simulator to extract the data. Then, the extracted data is used for both the training and testing of the BNN model as well as the algorithms mentioned.

The data from the simulator is discretized for a limited number of cells. The pollution and traffic are used to train the BNN model, being the traffic the input and pollution the target. This model will then evaluate different traffic data to forecast the value of pollution. As the output comes as a Gaussian, the forecast is the mean and the uncertainty is the standard deviation multiplied by a z-score for 95%. As a disclaimer for this rule, as the model was already providing an upper bound and lower bound associated

with this standard deviation, the real uncertainty used in the experiments is the difference between these two bounds, which is just twice the value mentioned. The algorithms can use the provided values to make their decision of placement and movement.

To simulate a reduction in the uncertainty according to the presence of a sensor in a certain cell, we considered that when a sensor is in some cell, the pollution value used is the real pollution of the cell since the sensor can measure the pollution directly and we consider that the uncertainty regarding this value is zero. When the sensor leaves the cell, the uncertainty regarding the last measurement of pollution follows the formula (4.2), chosen only based on practical trials and suitable for this specific case.

$$u_{t+1} = u_t * 3 + 0.02 \tag{4.2}$$

where u_t is the uncertainty at time step t. The uncertainty increases according to (4.2) until reaching the uncertainty associated with the forecast, the time when we start to consider the forecast uncertainty, as well as the forecast. Therefore, the pollution value that we use is based on the current uncertainty. We use the last pollution measurement of a sensor in that cell if the uncertainty is lesser than the uncertainty of the forecast, and we use the forecast of pollution if the uncertainty associated with the last measurement becomes greater than the uncertainty associated with the forecast.

The state of each time step includes the previously mentioned variables for every cell, such as the real pollution, the forecast of the pollution (needed for the cells where the sensor is not present), the uncertainty associated with the pollution (forecast or last measurement), and the traffic. The sensors can observe the variables described in every cell, but not the pollution real-time value in the cells where they are not present. The actions are only needed for the mobile sensors and contain the movements to every cell including staying in the same cell. We assumed that one time step (10 minutes) was enough for a sensor to move to a cell and take the real pollution measurement.

All the algorithms will follow the same reward for each time step and it is described by the formula in (4.3). The parameters δ_i help us change the relevance that is given to each of the other variables and test different combinations in the chapter 5. The variable $\overline{p_t}$ the value of the average real pollution over 10 minutes (600 time steps of the simulator) that will be covered by every sensor in its cell at each time step t. The average uncertainty $\overline{u_t}$, as explained before is associated with the measured value or the forecast and is considered to be the uncertainty that we would have in that cell if the sensor was not present, i.e. the reduction of uncertainty that we are getting for being in that cell. The last variable c corresponds to the cost of moving the vehicle of the sensor multiplied by the Manhattan distance $d(s_{t-1}, s_t)$ between the last cell s_{t-1} and the current cell s_t, given by the formula $|x_1 - x_2| + |y_1 - y_2|$.

$$R(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t) \tag{4.3}$$

This reward function will be revisited for each algorithm in the next section since in the case of static sensors, there is no cost of movement, and in the reactive algorithm, we use both the reward function and another similar function to make the decision.

4.2 Static Sensors

This section presents an algorithm that serve as a comparison to the mobile sensors. It is a simple greedy algorithm that would make the static placement decision for each sensor separately and sequentially.

4.2.1 Greedy Algorithm

To have a good benchmark for our sensor placement algorithms we started with a simple greedy algorithm for static sensor placement. It works in a very straightforward way and the idea is that each sensor is placed sequentially and only depends on the sensors placed so far. This first approach chooses the best location for the sensors according to the gain of the cell during a certain interval. The gain is represented in (4.4).

$$Gain = \delta_1 \cdot \overline{p} + \delta_2 \cdot \overline{u} \tag{4.4}$$

where δ_i is the importance given to each factor and will be decided empirically. As we want to give more importance to the cells with the most pollution and the most uncertainty, the sensors should be placed there. So we try to maximize the average measured pollution $\overline{p}$, and the average uncertainty $\overline{u}$. Then the algorithm runs through all the sensors and places each of them one by one as described in Algorithm 4.1.

Algorithm 4.1: Greedy Algorithm for static sensors

```
begin
    Initialize S as the set of sensors;
    Monitor the values of the grid G without sensors;
    foreach s ∈ S do
        Get the cell with the most Gain not yet selected;
        G(best) ← s;
```

4.3 Mobile sensors

The previous section covered the placement of sensors if they were static. This approach can have certain limitations, mainly related to the area that they could cover. Other low-cost sensors capable of moving could be used to cover a bigger area. Therefore, we present first a reactive algorithm that tries

to overcome this problem followed by a RL algorithm that could be applied for these sensors to learn how to interact with the environment.

Reinforcement learning is one of the main fields to approach these types of problems aiming for long-term results and where an agent learns how to act by exploring the environment and developing its model. It was chosen due to its properties of online interaction with the environment, its simplicity, and its popularity, which is very useful since there is a lot of information about its algorithms and variations.

4.3.1 Reactive Algorithm

The first algorithm to deal with mobile sensors is simple and makes its decisions on every time step based only on information from the last experience, having also into account the costs of moving between cells. As shown in 4.2, the algorithm will get a gain associated with the cells that are currently being covered by sensors at each time step. This is used to compare results with the other algorithms, and should not be confused with the RL reward.

The algorithm chooses the next action, i.e. the next cell to move to, by analyzing the current state. As we have information about the pollution on the cells covered by sensors, the forecast on the other cells, the uncertainty of every cell, and the cost of moving between cells is known a priori by each sensor, we created a simple metric, named potential gain, to evaluate the gain that we would get if we moved to other cells, using the cost of moving to that cell in the next time step. It calculates a potential gain so that the sensors can move to new cells and get a better gain in the next time step. The algorithm will achieve a performance as good as the similarity between two consecutive time steps, since the gain would also be similar in the next time step, being better if the cost of moving to that cell in the next time step was not too big.

Algorithm 4.2: Reactive Algorithm

```
begin
    Initialize M as the set of mobile sensors;
    foreach t ∈ timesteps do
        foreach m ∈ M do
            Calculate the potential gain for all the cells;
            Get a gain for the current cell with the sensor;
            Move the sensor to the cell with the best potential gain not yet selected;
```

The gain is presented in (4.5), where $\overline{p}$ is the average real pollution over 10 minutes, $\overline{u}$ the uncertainty of the value of pollution being considered if the sensor was not in that cell, and c is the cost at a certain time step multiplied by the Manhattan distance $d(s_{t-1}, s_t)$ between the last cell and the current cell. The potential gain (4.6) is similar, with a change in the pollution parameter, since we cannot have the real value, due to the lacking of sensors in that region. Therefore, we use the forecast f. This is possible since the only variable needed to calculate the forecast is the traffic and at the end of the time step, we

already have access to that information. Also, the distance is between the current cell and cells it can move to in the future.

$$Gain(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t) \tag{4.5}$$

$$Gain_{potential}(s_t, a_t) = \delta_1 \cdot \overline{f}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_t, s_{t+1}) \tag{4.6}$$

4.3.2 Reinforcement Learning algorithm

Environment description

The environment can be described as an MDP where we can define the tuple (S, A, P, R), where S is the set of states, A the set of actions, P the transition probabilities, and R the reward function that maps the state-action value to a real number.

Then, it is needed to describe our problem according to that tuple:

- **States:** each state s_t represents a set of cells each with the real pollution value from the simulator in the cells with sensors, the forecast of the pollution in the remaining cells, the number of vehicles, the uncertainty from the BNN model and from past visited cells, as explained before, and the position of the vehicles at a certain time step t.
- **Actions:** the agent can choose an action a_t to move to each cell of the grid or stay according to the policy.
- **Reward:** the reward function of each agent that maps state and the actions a_t of every sensor values to a real number, is given by the expression $R(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t)$, where δ_i are parameters determined empirically, $\overline{p_t}$ the average real pollution over 10 minutes (600 time steps of the simulator) that will be covered by every sensor in its cell, $\overline{u_t}$ the average uncertainty for the cell if the sensor was not present. As explained before, when a sensor moves to a cell its uncertainty decreases to zero and increases following the equation (4.2) until reaching the forecast uncertainty (being equal to the forecast uncertainty until a sensor passes the cell again), and c the cost of moving the vehicle of the sensor and $d(s_{t-1}, s_t)$ the Manhattan distance between the last cell and the current cell.

Our reward function will contain information about pollution and uncertainty since the environment is very informative and these are the main measurements of our sensors. These measurements are normalized between 0 and 1 so we can easily vary the relevance of these parameters since they have

the same scale. Also, we added a cost to once more test if RL can make decisions that avoid high-cost movements of the sensors.

Implementation

For this implementation, the algorithm followed an actor-critic-based structure. Other algorithms such as variations of A2C or Asynchronous Advantage Actor-Critic (A3C) were thought to be a good fit, but we opted for PPO due to its simplicity, and its updates control mechanism.

By looking at the pseudocode in 4.3 we can see that each sensor (we will refer to a sensor as a vehicle carrying the sensor for simplicity) will have its own actor and critic. It will be a neural network model that receives as input the information described in the environment description normalized with $\frac{x_i - min(x)}{max(x) - min(x)}$. The model will output the best action according to the policy.

Algorithm 4.3: Independent Learning PPO algorithm

```
begin
    Initialize M as the set of mobile sensors;
    foreach i ∈ iterations do
        foreach t ∈ timesteps do
            foreach m ∈ M do
                Pass state s_t of m as input to its actor and critic;
                Use an action mask to avoid collisions;
                a_t(s_t) ← action output from the actor model;
                v(s_t) ← value output from the critic model;
        Calculate the advantage estimates;
        Update all of the actors;
        Update all of the critics;
```

In this implementation, each agent will have its own actor-critic structure and learn independently from the other agents. However, the choice of the actions will be made sequentially. Therefore, when choosing an action, each agent will have an additional information external to the policy about the decisions of the agents that precede it on the priority list to avoid collisions between agents.

Both the actor and the critic have the same neural network structure, to reduce the running time of our training. It was composed by 2 hidden layers of 256 neurons using a hyperbolic tangent as their activation function. Then we choose the greatest value from an output vector of 20 neurons to get the action and the second output to obtain the value of the actor and critic respectively. The architecture mentioned is represented in the Figure 4.3.

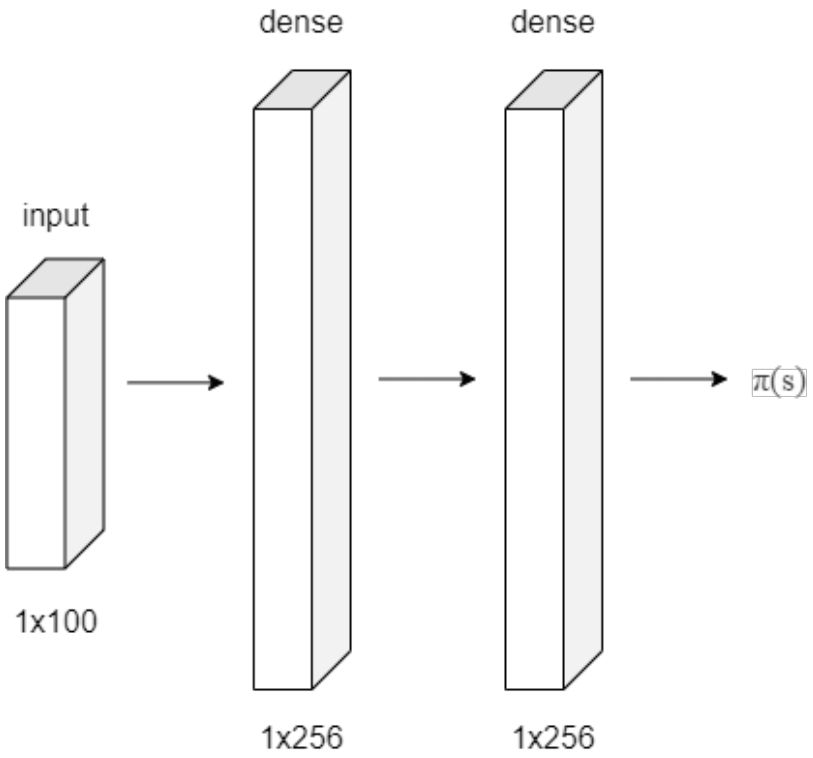


Figure 4.3: Architecture of the model.

Layer	Activation Shape	Activation Size	# Parameters
Input Layer	(100, 1)	100	0
Dense Layer 1	(256, 1)	256	25856
Dense Layer 2	(256, 1)	256	65792
Output (actor)	(20, 1)	20	5140
Output (critic)	(1, 1)	1	257

Total parameters: 97045
Trainable parameters: 97045
Non-trainable parameters: 0

Table 4.1: Parameters of the Neural Network

5

Experiments & Evaluation

Contents

This section will cover the experiments used to evaluate the algorithms described in the last sections. Python was the main programming language and Matplotlib was used as the preferable framework due to the number of tools that it has to simplify the visualization of information. To show the results, graphs were used to compare parameter variations within the same algorithm or to compare the different algorithms.

The experiments were performed in the city of Trondheim, Norway, where we have monitored the performance of each algorithm over 28 days divided into timesteps of 10 minutes. The data collected consisted of both emissions and traffic as shown in the Figures 5.1 and 5.2. During the experiments, we provided 61 days of data to train the BNN and 56 days to test. The test sample was then used to train the algorithms of the sensors with 28 days and 28 days to test.

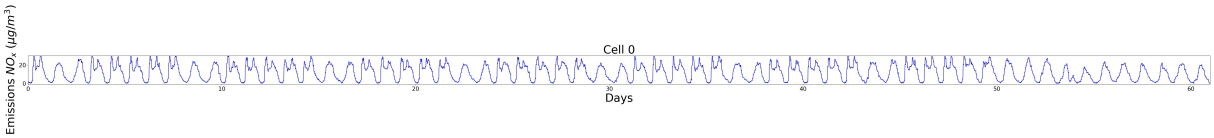


Figure 5.1: Values of NO_x emissions during 61 days for one cell.

Figure 5.2: Values of traffic during 61 days for one cell.

5.1 Data Analysis

To understand if the data provided by the simulator, in this case, the pollution and the traffic, was realistic and sufficient for our experiments, we plotted a graph with a single day mean values of data collected over 61 days (Figures 5.3 and 5.4), where we can observe an increase in the pollution and traffic during the middle of the day, correspondent to the period of time when people are working and being active. The lowest values are in the morning and at night for cell 0, with a minimum for emissions of 0.82 $\mu g/m^3$ and maximum of 23.07 $\mu g/m^3$, and a minimum for traffic of 0.04 and maximum of 1.43.

Recall that the measurements of traffic refer to the average number of vehicles during the length of a time step (10 minutes) that were moving and contributing to the pollution every second as explained in section 4.1. Therefore, these values should not be confused with the total number of vehicles in that cell during the 10 minutes, which would be much higher than the presented ones.

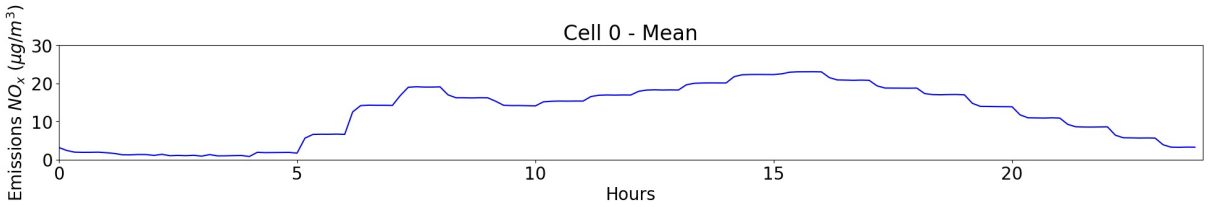


Figure 5.3: Mean values of NO_x emissions of a single day from data collected over 61 days for one cell.

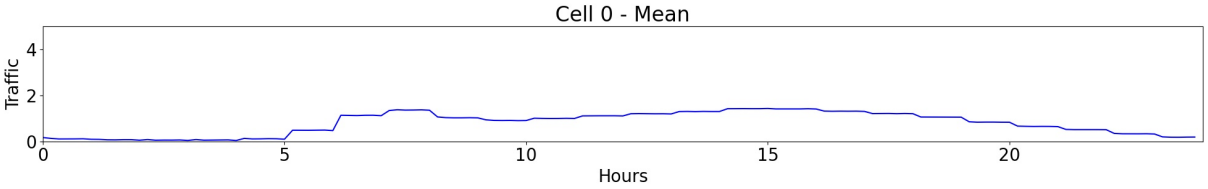


Figure 5.4: Mean values of traffic of a single day from data collected over 61 days for one cell.

Apart from the variation during the day, it is also important to understand if the values are not the same every day. Therefore we plotted the standard deviations for each hour of the day corresponding to the same 61 days. The Figures 5.5 and 5.6 provide us good insights that the data is not constant and have temporal variation. For cell 0 there was a minimum for emissions of 0.63 $\mu g/m^3$ and maximum of 12.71 $\mu g/m^3$, and a minimum for traffic of 0.04 and maximum of 0.97.

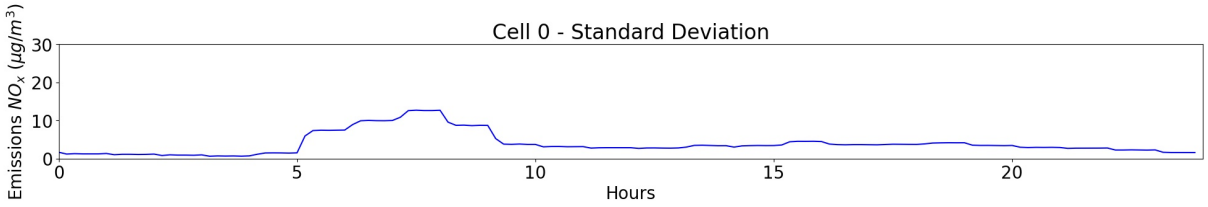


Figure 5.5: Standard deviation values of NO_x emissions of a single day from data collected over 61 days for one cell.

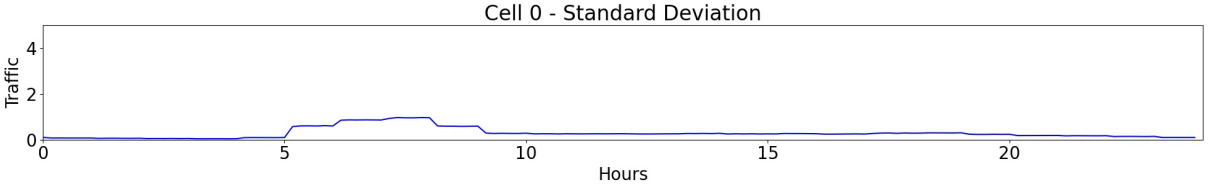


Figure 5.6: Standard deviation values of traffic of a single day from data collected over 61 days for one cell.

5.2 Uncertainty

Since the main goal of the project was not to have the best model to measure uncertainty, we used the values provided by the BNN model since they were acceptable as shown in Figures 5.7, 5.8 and 5.9.

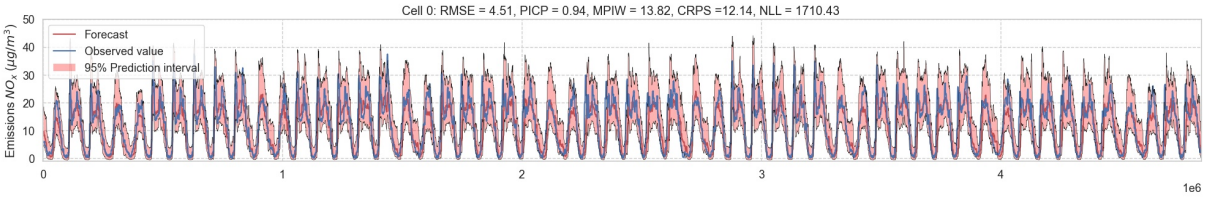


Figure 5.7: BNN output (real value, forecast, upper and lower bound) during 56 days for one cell.

Each point of Figure 5.7 shows us the mean value of the Gaussian output by the BNN model that is considered the forecast. It also has the real value measured and a lower and upper bound, already explained in section 4.1.

The next two figures show us the forecast and uncertainty values separately from the other data described in the last figure. Although the deviation from the forecast, and here called uncertainty, is insignificant in some cases, being the minimum 1.08 $\mu g/m^3$ of difference, we got a maximum value of uncertainty of 62.07 $\mu g/m^3$, which is more than half of the maximum forecast value of 110.83 $\mu g/m^3$. Therefore, the use of sensors to reduce this uncertainty is needed to ensure that these regions are being covered when the forecast is not reliable enough.

	uncertainty - all cells	uncertainty - cell 0	forecast - all cells	forecast - cell 0
max	62.07 $\mu g/m^3$	15.91 $\mu g/m^3$	110.83 $\mu g/m^3$	28.73 $\mu g/m^3$
min	1.08 $\mu g/m^3$	2.05 $\mu g/m^3$	0.21 $\mu g/m^3$	0.21 $\mu g/m^3$

Table 5.1: Maximum and minimum values of uncertainty and forecast of emissions.

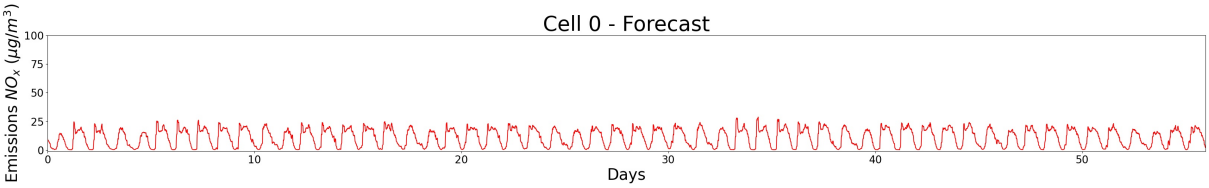


Figure 5.8: Emissions forecast during 56 days for one cell.

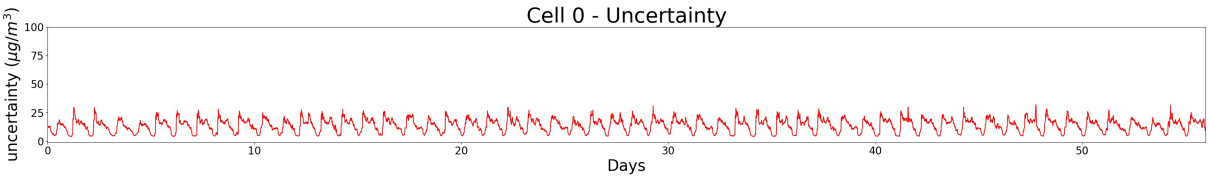


Figure 5.9: Uncertainty during 56 days for one cell.

5.3 Experiments with algorithms

After studying the data, the algorithmic experiments could now be conducted. Since the greedy algorithms always get the same results, we used a constant line to represent the static greedy and the reactive algorithms.

As the problem of this project is to determine the optimal placement of sensors or the policy for the movement of the mobile sensors, the following items were tested and will be discussed during this section.

- Pollution coverage
- Reduction of uncertainty in pollution measurements
- Number of sensors
- Different combinations of the above experiments
- Test each algorithm with different parameters
- Running time of the algorithms

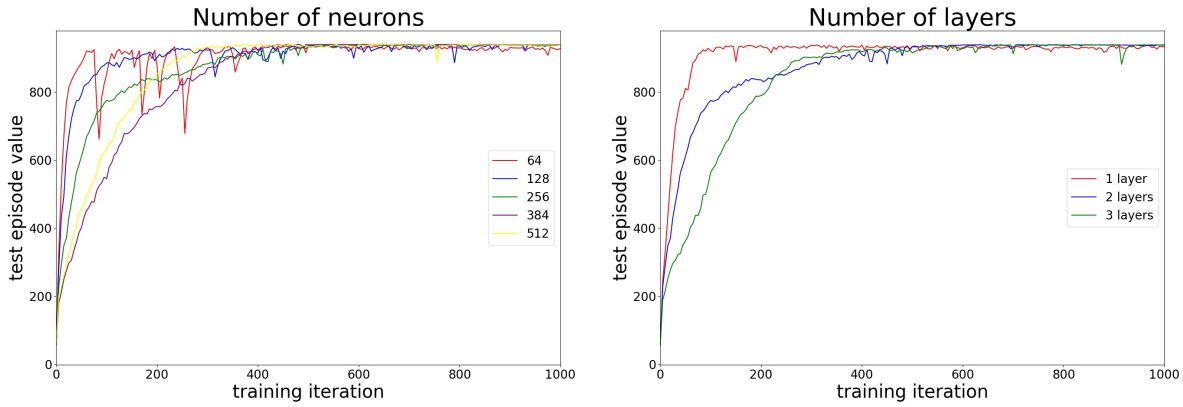
To test the pollution coverage and reduction of uncertainty in pollution measurements we changed the models' parameters described in the architecture. The bigger values were given to the parameters that we wanted to isolate. The cost of the system had into account the costs of installation and maintenance of the sensors and the costs of moving those sensors in the case of vehicles.

Other tests covered the number of sensors that were considered for the algorithms. This implied different results for the aforementioned experiments. This experiment provided an understanding of the implications of the use of more or fewer sensors in the performance of the algorithms. Other variations combining different combinations of coverage for pollution and uncertainty, as well as the number of sensors were tested. Also, the architecture of the algorithms, including the number of epochs, parameters of neural networks etc. were analyzed in more detail in order to achieve the best results. These changes also complemented the results of the experiments mentioned.

The last experiment was related to the running time of the algorithms for specific hardware and software characteristics to provide useful insights for future implementations of such algorithms.

5.4 Architecture of the Reinforcement Learning algorithm

This subsection addresses some choices of parameters considered to be the most relevant ones in the architecture of the Reinforcement Learning algorithm. We used different numbers of layers and neurons. Each point of the next two graphs represents the result of cumulative reward during 4029 time steps, each time step corresponding to 10 minutes in real life. Each point represents the result of the algorithm in a testing environment, using a trained model after x iterations of training.



(a) Different number of neurons.

(b) Different number of layers.

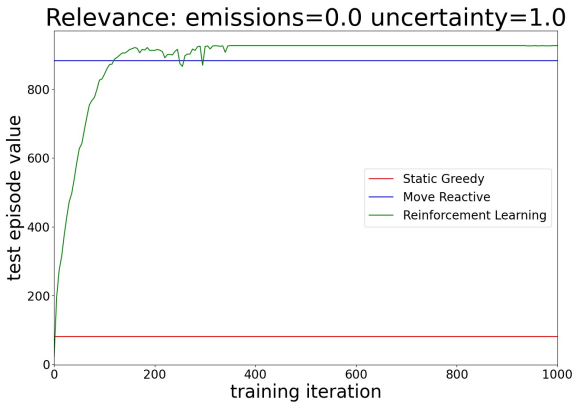
As shown in Figures 5.10(a) and 5.10(b) we can understand that the algorithm converges much faster with fewer neurons and fewer layers. However, with more training, the other alternatives start to converge to the same value. Then we chose 256 neurons and 2 layers for our experiments since the problem is not complex enough to use more parameters, but at the same time, it is important to ensure that we still get the most out of the information that comes as input. According to that, the algorithm can also learn faster, which helped us in the other experiments and can useful in future applications.

5.5 Pollution coverage and uncertainty reduction

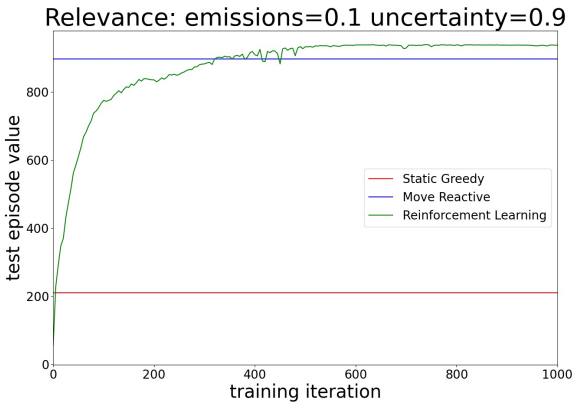
As mentioned before the main goal is to study how we can impact the pollution coverage and reduce the uncertainty of those measurements. Then it is important to understand how the main algorithm used in this project deals when we give different relevance to pollution and uncertainty. Recalling the reward $R(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t)$, explained before, this section studies how changes in parameters δ_1 and δ_2 change the behavior of the algorithm, being these parameters the relevance given to the covered pollution $\overline{p}_t$ and the reduction of uncertainty $\overline{u}_t$.

Once again, each point of the graphs represents the result of cumulative reward during 4029 time steps, and represents the result in a testing environment, using a trained model after x iterations of

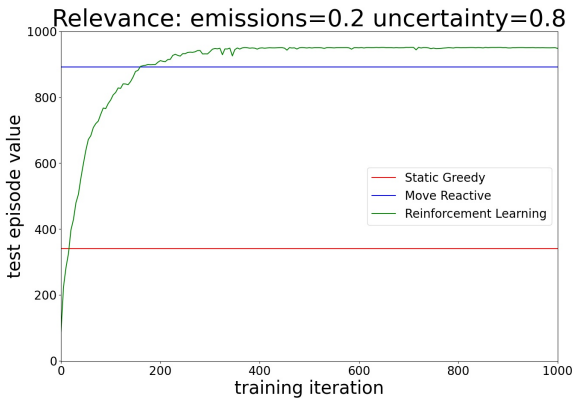
training. The Figures in 5.10, show us that the RL algorithm outperforms all the other baselines for every combination, which presents good versatility for different combinations of parameters. We could also observe that the difference between the RL algorithm and the Reactive is not huge. This is a result of a high correlation between the values of emissions and uncertainty of two consecutive time steps. Also the difference to the algorithm of static sensors decreases with more importance given to the pollution.



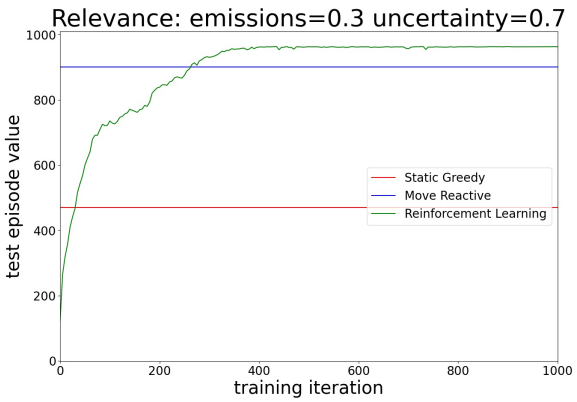
(c) emissions=0.0; uncertainty=1.0



(d) emissions=0.1; uncertainty=0.9



(e) emissions=0.2; uncertainty=0.8

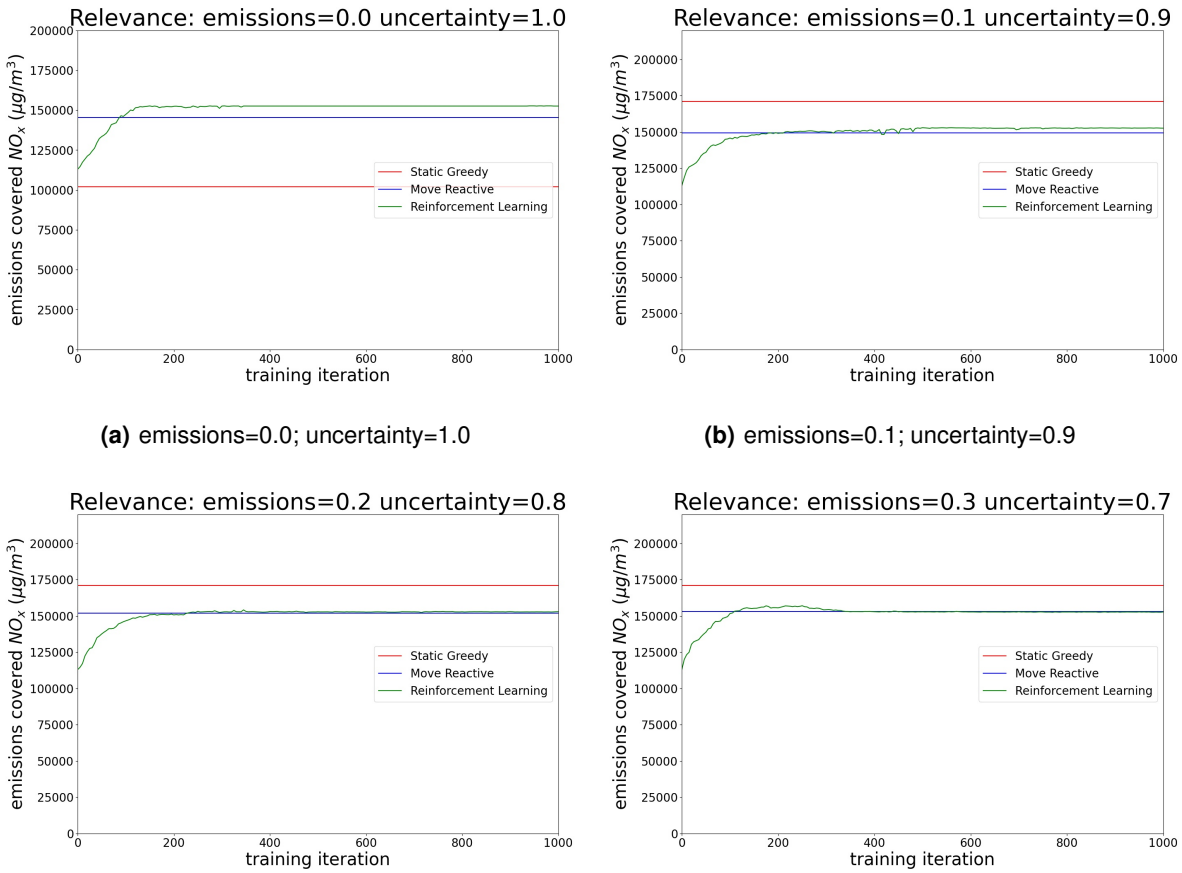


(f) emissions=0.3; uncertainty=0.7

Figure 5.10: Comparison of the reward of the algorithms for 1 sensor.

Despite the reward value, to figure out what was the real behavior of the algorithms related to the concrete values, i.e. the emissions and uncertainty, we plotted two other sets of Figures (5.11 and 5.12). Figure 5.11 shows that although all the algorithms increase the coverage of pollution with the increase of importance of this parameter, the difference between RL and Reactive decreases due to the correlation referred to before in the pollution measurements in two consecutive time steps. The uncertainty is less dependent on a certain cell and varies more. The static sensors have a greater value of pollution coverage in the three combinations where the emissions are in the reward since it is all the

time in the cell with the most pollution. Of course we do not need to have a sensor in a cell with a big pollution value if the uncertainty of the forecast is very low. As the mobile sensors also move to cover the uncertainty, the parameter with the most relevance, they are not every time in the cell with the most pollution, but also in the ones with the greatest uncertainty as well as the cost of moving associated.



(a) emissions=0.0; uncertainty=1.0

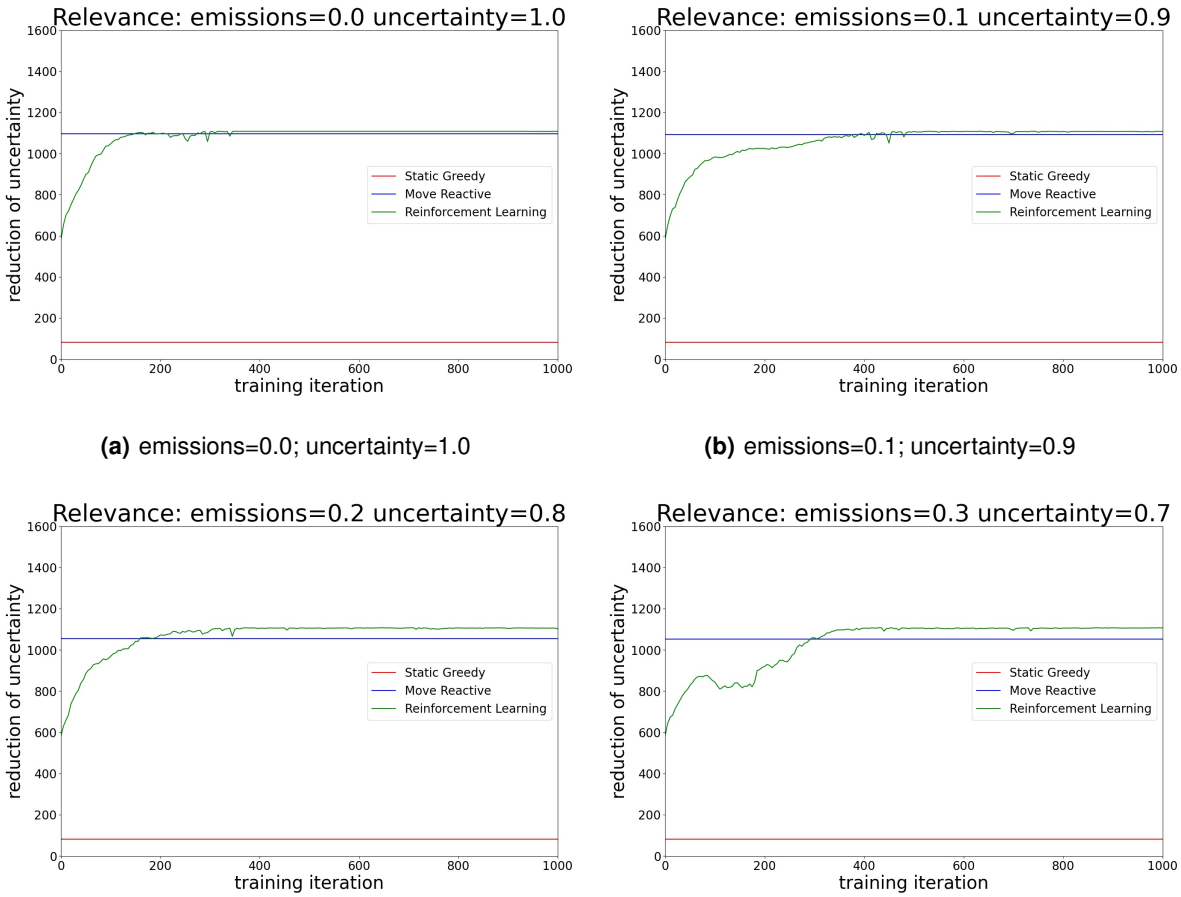
(b) emissions=0.1; uncertainty=0.9

(c) emissions=0.2; uncertainty=0.8

(d) emissions=0.3; uncertainty=0.7

Figure 5.11: Comparison of the pollution coverage of the algorithms for 1 sensor.

The reduction of uncertainty is also what we expected, being more related to the reward, since it has the main importance in it, and so, the Reinforcement Learning algorithm outperforms the reduction of uncertainty of the other algorithms as shown in the set of figures 5.12. The decrease in uncertainty with the decrease of its relevance follows the variation of the reward in the figures in 5.10.



(a) emissions=0.0; uncertainty=1.0

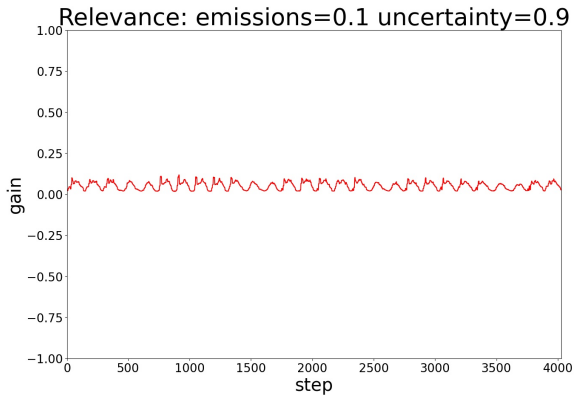
(b) emissions=0.1; uncertainty=0.9

(c) emissions=0.2; uncertainty=0.8

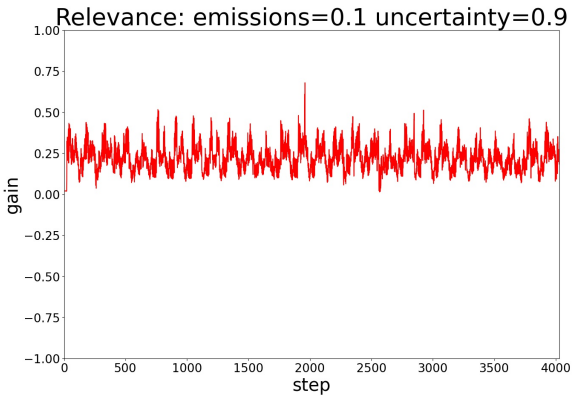
(d) emissions=0.3; uncertainty=0.7

Figure 5.12: Comparison of the reduction of uncertainty of the algorithms for 1 sensor.

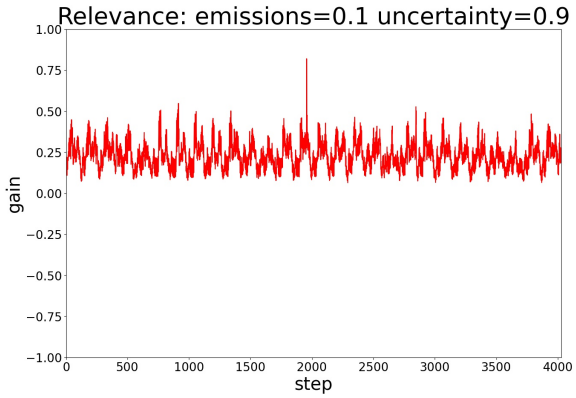
The next three Figures 5.13(a), 5.13(b) and 5.13(c) show us the reward gathered on each time step of a complete episode applying each of the three algorithms for the case of relevance of 0.1 in emissions and 0.9 in uncertainty. In the case of the RL algorithm, the episode corresponds to the one that gave us the best results. As we can understand, as the difference between RL and reactive is small, visually there is not much difference in the step gain pattern, being the reward peaks are located in the middle of each day when pollution and uncertainty are bigger.



(a) Static



(b) Reactive



(c) Reinforcement Learning

Figure 5.13: Comparison of the step reward in a full episode.

We also show a graph (Figure 5.15) for cell 12 that shows us a significant reduction in the uncertainty of this cell when compared to Figure 5.14, since it is one of the cells with the most uncertainty, and therefore a priority for the sensors to move to.

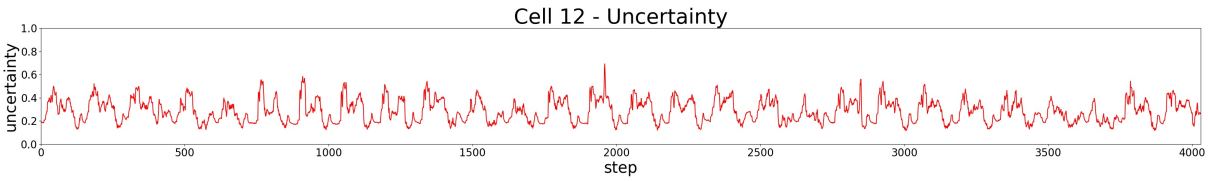


Figure 5.14: Normalized uncertainty in the test environment for 4029 steps without sensors for cell 12.

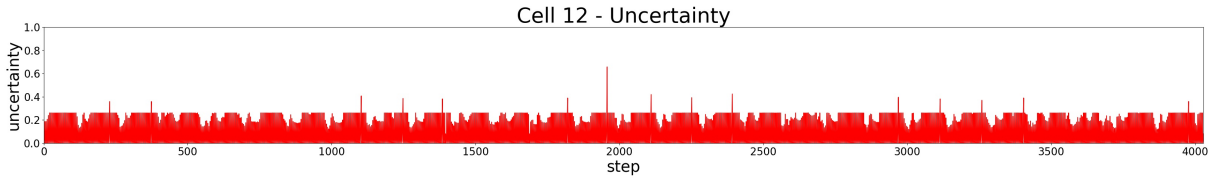


Figure 5.15: Normalized uncertainty in the test environment for 4029 steps with sensors for cell 12.

On the other hand, cell 0 in Figures 5.16 and 5.17 we do not see any reduction since it is a cell with much less uncertainty and not a priority for the sensors.

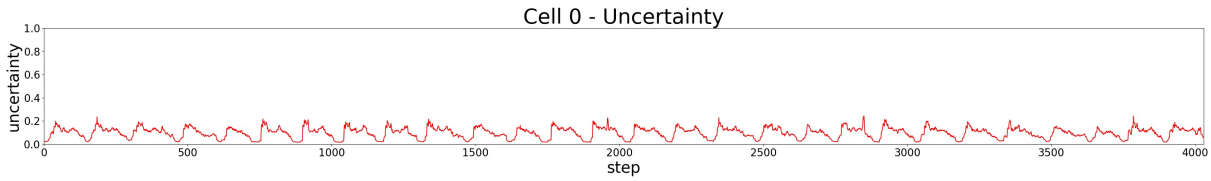


Figure 5.16: Normalized uncertainty in the test environment for 4029 steps without sensors for cell 0.

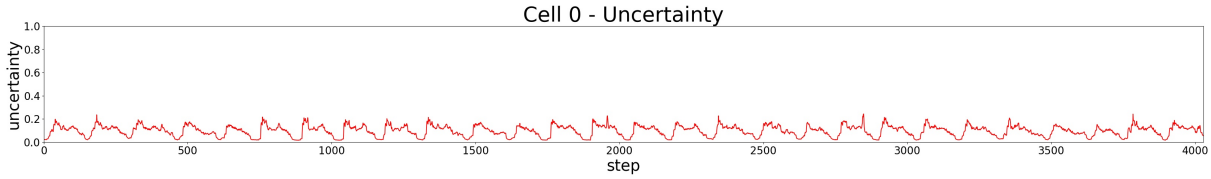
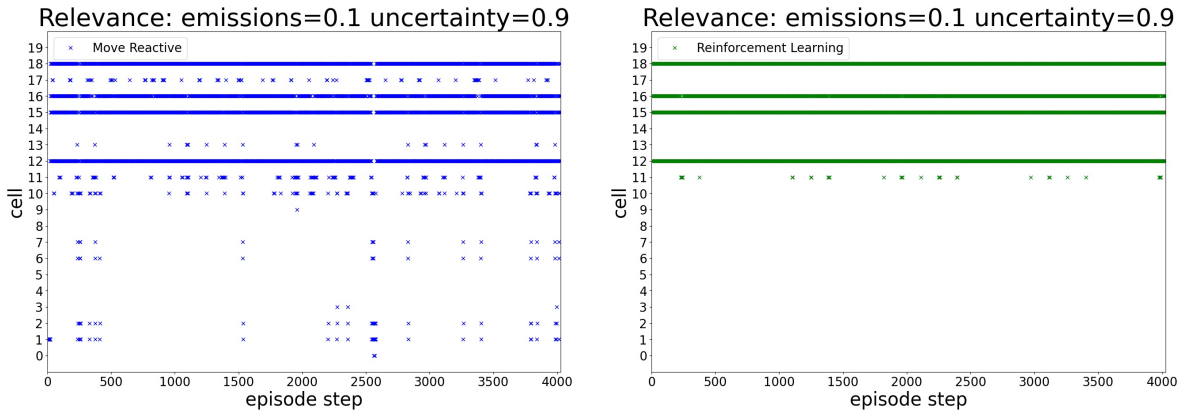


Figure 5.17: Normalized uncertainty in the test environment for 4029 steps with sensors for cell 0.

However, if we compare the movement of the sensors in Figures 5.18(a) and 5.18(b) we find that the RL algorithm moves fewer times than the reactive one. We plotted the movement of a single sensor. Each point represents the cell where the sensor is present at a certain time step. Also, there are certain cells that encompass the majority of the presence of the sensors. This result was expected since these cells are the ones with the biggest values of pollution and uncertainty.



(a) Reactive

(b) Reinforcement Learning

Figure 5.18: Comparison of the sensor movement.

5.6 Variation in the number of sensors

For this experiment, we wanted to both compare the performance of different algorithms with different numbers of sensors and also know how useful it is to use more sensors in our environment. We fixed the importance of pollution at 0.1 and uncertainty at 0.9.

The RL algorithm outperformed the other algorithms with 1 and 2 sensors as shown in Figures 5.19(a), 5.19(b). However, if we increased the number of sensors, we did not achieve better performance as shown in Figure 5.19(c). As our reactive algorithm makes a decision mainly based on the last state of the environment, we can conclude for these small differences in return between reactive and RL that the current state information is highly correlated with the previous state and with more sensors the advantages of our RL algorithm cease to be noticed so much, since with more sensors it is not necessary to account for these variations, as the reactive algorithm already has enough resources to deal with the problem. Also, the RL algorithm only gets a reward used for learning according to the cells where it is, contrasting with the reactive agent that creates a potential gain for every cell even if the sensor is not there because of other sensors.

(a) 1 sensor

(b) 2 sensors

(c) 3 sensors

Figure 5.19: Comparison of the algorithms with different numbers of sensors.

As we can conclude by looking and Figure 5.20, although we get better performance by adding more sensors, the increase in performance does not follow.

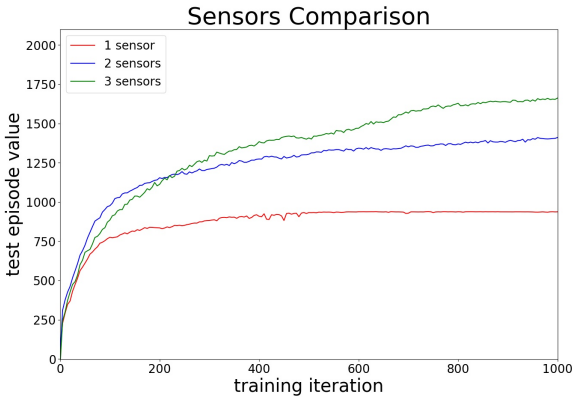


Figure 5.20: Reward comparison between different number of sensors.

5.7 Running Time

The algorithm's running time is mainly dependent on the number of sensors and the architecture of the algorithms. For that matter, we used the best-tested architecture of each algorithm and varied the number of sensors.

As shown in Figure 5.21 and Table 5.2, we can understand that the RL algorithm needs much more time than the other algorithms to learn the policy. While the others need a few seconds, RL needs hours to train. Also, the number of sensors will have a greater impact on the RL algorithm when compared to the other algorithms since we would have to learn a policy for each new sensor.

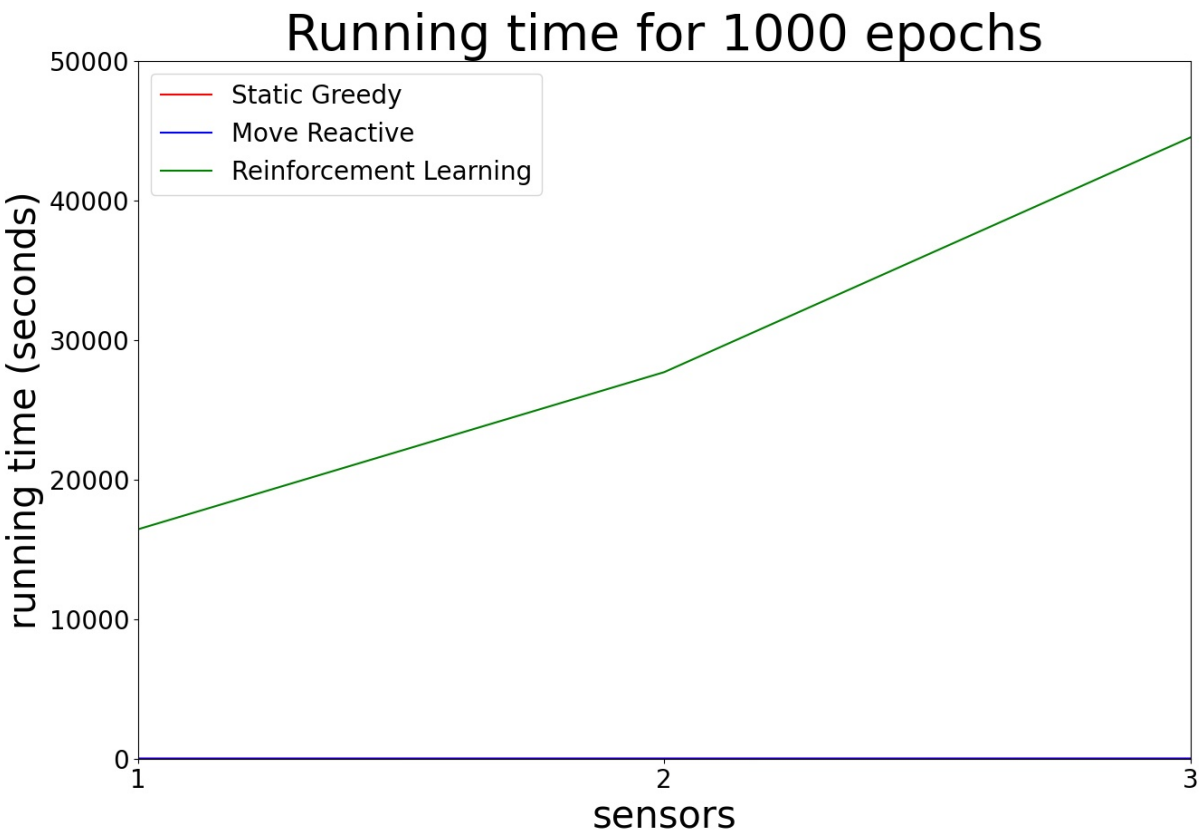


Figure 5.21: Running time regarding the number of sensors.

	1 sensor	2 sensors	3 sensors
Static Greedy	4.39 s	4.44 s	4.22 s
Move Reactive	0.95 s	1.26 s	2.82 s
Reinforcement Learning	4 h 33 m 54 s	7 h 41 m 32 s	12 h 21 m 42 s

Table 5.2: Running time of the algorithms.

5.8 Discussion

After analyzing the results that our experiments provided, we can summarize the conclusions reached for each experiment and compare the results to our expectations.

The main outcome regarding the performance of the algorithms was between static and mobile sensors, and we could easily understand that, as expected, the use of mobile sensors was a great advantage and led to substantially better performance. The same was not so evident between the Reactive algorithm and the RL algorithm. Although the RL algorithm had better performance with 1 sensor and 2 sensors, the difference was not so significant, and one could justify that for some specific cases the Reactive algorithm is preferred for being less complex and easier to implement. Also, when we increased the number of sensors, this advantage started to vanish and the Reactive algorithm surpassed the RL algorithm.

As explained during the chapter, we understood that a time step was strongly related to the following one, and therefore we were already expecting a simple algorithm to be able to also have a good performance. In this case, reactive also takes advantage of its potential gain, which can be done for every cell even if the sensor is not there because of other sensors, alike the RL algorithm that learns with the rewards of the cells where the sensor is. With a different data set, the results could be different. The movement of the sensors was also similar between these two algorithms, and they focused their effort on a reduced number of cells, but the RL algorithm showed to move less than the Reactive algorithm to cells different from these ones. There were cells where neither of them have moved during an entire episode.

Lastly, as the complexity of the algorithm that used RL was bigger than the other algorithms, the running time was also superior to the others. Then, it would always impact the choice of one of the algorithms provided in a real-world situation.

6

Conclusion

Contents

6.1 Conclusions

In this thesis, optimal deployment of sensors was studied for an environment where we want to cover the regions more prone to high pollution levels at the same time as we reduce the uncertainty about the measurements in that regions. Both static and mobile sensors networks were approached to achieve this objective. We presented a greedy algorithm with the sequential placement of the sensors for static sensors. To deal with mobile sensors, which are considered by the literature the ones that usually provide the best coverage solutions, we used a reactive algorithm and a PPO reinforcement learning algorithm.

Early in the first chapters the goals and research questions are defined to guide the research for literature. The structured literature review uses a protocol to both support the research and make it reproducible. Nonetheless, it tries to answer the research questions in an explicit or implicit way, i.e. not every question had its query since most of the studies found also presented references for other relevant studies, which at the end permitted to answer all the questions. As referred before, different studies about sensors placement were found both for static and mobile sensors, with some preferences for the mobile sensors to accomplish better coverage. The static placement served as baseline to compare with the mobile sensors algorithms owing to their simplicity. The last goal was related to the mobile sensors and since there were already some notions of reinforcement learning and how these types of techniques can solve similar problems, first we tried to find how competitive it is to other algorithms, reaching the conclusion that for its simplicity and efficiency it was the most suitable algorithm to use in this thesis for a small number of sensors, but not with more than three, which is a limitation of the algorithm for our data.

Also, when comparing the RL algorithm with the Reactive one, we found that the sensors were moving less in RL, prioritizing certain cells with the most relevance, unlike reactive which had a greater spatial presence.

6.2 System Limitations and Future Work

Although our algorithm was shown to outperform our baselines in the referred situations, we could test the algorithm with other baselines besides the ones mentioned before. Also, as in this case, the pollution measurement was NOx, it would be interesting to find out if the algorithm gets similar results with different compounds.

Since the project's scope was not to make the best uncertainty model but to reduce this uncertainty through the positioning of sensors, we could investigate other algorithms in this area in further work.

The Reinforcement Learning algorithm used independent learning, restricting agents from moving to positions with other agents to avoid collisions. Despite the good performance of this algorithm, in

scenarios with reduced or no communication, it would be impossible to perform such an approach, which brings a topic for investigation as well.

The simulator could also be adjusted to include more variables, such as the wind or the rain to make the propagation of the pollution more realistic.

Bibliography

[1] M. Marfeychuk, “Learning control policies in smart cities from physical data,” 2020.

[2] A. Murad, F. A. Kraemer, K. Bach, and G. Taylor, “Probabilistic deep learning to quantify uncertainty in air quality forecasting,” *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/8009

[3] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction*, 2nd ed. The MIT Press, 2014.

[5] J. A. Bernstein, N. Alexis, C. Barnes, I. L. Bernstein, A. Nel, D. Peden, D. Diaz-Sanchez, S. M. Tarlo, and P. B. Williams, “Health effects of air pollution,” *Journal of allergy and clinical immunology*, vol. 114, no. 5, pp. 1116–1123, 2004.

[6] R. D. Brook, “Cardiovascular effects of air pollution,” *Clinical science*, vol. 115, no. 6, pp. 175–187, 2008.

[7] M. Kampa and E. Castanas, “Human health effects of air pollution,” *Environmental pollution*, vol. 151, no. 2, pp. 362–367, 2008.

[8] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic traffic simulation using sumo,” in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[9] C. J. C. H. Watkins, “Learning from delayed rewards,” 1989.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction*, 2nd ed. The MIT Press, 2018.

[11] R. McFarlane, “A Survey of Exploration Strategies in Reinforcement Learning,” 2003. [Online]. Available: https://www.

semanticscholar.org/paper/A-Survey-of-Exploration-Strategies-in-Reinforcement-McFarlane/02761533d794ed9ed5dfd0295f2577e1e98c4fe2

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning - Nature," *Nature*, vol. 518, pp. 529–533, Feb 2015.

[13] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[15] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 514–529, 2018.

[16] A. Kofod-Petersen, "How to do a structured literature review in computer science," Oct 2018. [Online]. Available: https://research.idi.ntnu.no/aimasters/files/SLR_HowTo2018.pdf

[17] C. Sun, V. O. K. Li, J. C. K. Lam, and I. Leslie, "Optimal citizen-centric sensor placement for air quality monitoring: A case study of city of cambridge, the united kingdom," *IEEE Access*, vol. 7, pp. 47 390–47 400, 2019.

[18] C. Sun, Y. Yu, V. O. Li, and J. C. Lam, "Optimal multi-type sensor placements in gaussian spatial fields for environmental monitoring," in *2018 IEEE International Smart Cities Conference (ISC2)*, 2018, pp. 1–8.

[19] M. Quiñones Grueiro, C. Verde, and O. Llanes-Santiago, "Multi-objective sensor placement for leakage detection and localization in water distribution networks," in *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, 2019, pp. 129–134.

[20] X. Li, M. Sun, Y. Ma, L. Zhang, Y. Zhang, R. Yang, and Q. Liu, "Using sensor network for tracing and locating air pollution sources," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 12 162–12 170, 2021.

[21] F. Ghayem, B. Rivet, R. C. Farias, and C. Jutten, "Robust sensor placement for signal extraction," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4513–4528, 2021.

[22] J. Gao, L. Zeng, C. Cao, W. Ye, and X. Zhang, "Multi-objective optimization for sensor placement against suddenly released contaminant in air duct system," *Build. Simul.*, vol. 11, no. 1, pp. 139–153, Feb 2018.

[23] C. Hu, M. Li, D. Zeng, and S. Guo, “A survey on sensor placement for contamination detection in water distribution systems,” *Wireless Netw.*, vol. 24, no. 2, pp. 647–661, Feb 2018.

[24] G. F. Gomes, S. S. da Cunha, P. da Silva Lopes Alexandrino, B. Silva de Sousa, and A. C. Ancelotti, “Sensor placement optimization applied to laminated composite plates under vibration,” *Struct. Multidiscip. Optim.*, vol. 58, no. 5, pp. 2099–2118, Nov 2018.

[25] E. Q. Shahra and W. Wu, “Water contaminants detection using sensor placement approach in smart water networks,” *J. Ambient Intell. Hum. Comput.*, pp. 1–16, Jun 2020.

[26] A. Musa, V. Gonzalez, and D. Barragan, “A new strategy to optimize the sensors placement in wireless sensor networks,” *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 4, pp. 1389–1399, Apr 2019.

[27] G. Tajnafoi, R. Arcucci, L. Mottet, C. Vouriot, M. Molina-Solana, C. Pain, and Y.-K. Guo, “Variational Gaussian Process for Optimal Sensor Placement,” *Appl. Math.*, vol. 66, no. 2, pp. 287–317, Apr 2021.

[28] Z. Hu, W. Chen, B. Chen, D. Tan, Y. Zhang, and D. Shen, “Robust Hierarchical Sensor Optimization Placement Method for Leak Detection in Water Distribution System,” *Water Resour. Manage.*, vol. 35, no. 12, pp. 3995–4008, Sep 2021.

[29] M. R. Banik and T. Das, “Application of neuro-ga hybrids in sensor optimization for structural health monitoring,” in *Proceedings of the International Conference on Computing Advancements*, ser. ICCA 2020. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3377049.3377131

[30] Y. Zhen, M. Sugasaki, Y. Kawahara, K. Tsubouchi, M. Ishige, and M. Shimosaka, “Ai-bpo: Adaptive incremental ble beacon placement optimization for crowd density monitoring applications,” in *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 301–304. [Online]. Available: https://doi.org/10.1145/3474717.3483964

[31] H. An, B. D. Youn, and H. S. Kim, “A methodology for sensor number and placement optimization for vibration-based damage detection of composite structures under model uncertainty,” *Composite Structures*, vol. 279, p. 114863, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263822321013027

[32] H. Liu, Z. Zhang, and D. Wang, “Wrfmr: A multi-agent reinforcement learning method for cooperative tasks,” *IEEE Access*, vol. 8, pp. 216 320–216 331, 2020.

[33] R. Ding, Z. Yang, Y. Wei, H. Jin, and X. Wang, “Multi-agent reinforcement learning for urban crowd sensing with for-hire vehicles,” in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[34] S. Meng and Z. Kan, “Deep reinforcement learning-based effective coverage control with connectivity constraints,” *IEEE Control Systems Letters*, vol. 6, pp. 283–288, 2021.

[35] Z. Zhang, D. Wang, and J. Gao, “Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4639–4652, 2021.

[36] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, “Cooperative internet of uavs: Distributed trajectory design by multi-agent deep reinforcement learning,” *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[37] H. Luo, C. Liu, Y. Luo, X. Wang, Z. Xu, and Y. Liang, “Sdma: A sdn-based architecture of multi-modal auvs network,” in *Proceedings of the ACM Turing Celebration Conference - China*, ser. ACM TURC’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 181–184. [Online]. Available: https://doi.org/10.1145/3393527.3393558

[38] L. Busoniu, R. Babuska, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[39] C. Wang, X. Gaimu, C. Li, H. Zou, and W. Wang, “Smart mobile crowdsensing with urban vehicles: A deep reinforcement learning perspective,” *IEEE Access*, vol. 7, pp. 37 334–37 341, 2019.

[40] X. Tao and W. Song, “Task allocation for mobile crowdsensing with deep reinforcement learning,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–7.

[41] C. H. Liu, Z. Dai, H. Yang, and J. Tang, “Multi-task-oriented vehicular crowdsensing: A deep learning approach,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1123–1132.

[42] C. H. Liu, Z. Chen, and Y. Zhan, “Energy-efficient distributed mobile crowd sensing: A deep learning approach,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1262–1276, 2019.

[43] X. Lyu, Y. Xiao, B. Daley, and C. Amato, “Contrasting centralized and decentralized critics in multi-agent reinforcement learning,” *arXiv preprint arXiv:2102.04402*, 2021.

[44] J. Chen, A. Baskaran, Z. Zhang, and P. Tokekar, “Multi-agent reinforcement learning for visibility-based persistent monitoring,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2563–2570.

[45] S. Y. Luis, D. G. Reina, and S. Toral, “Maximum information coverage and monitoring path planning with unmanned surface vehicles using deep reinforcement learning.”

[46] A. Krause, A. Singh, and C. Guestrin, “Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies.” *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.

[47] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[48] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[49] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[50] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, “Liir: Learning individual intrinsic reward in multi-agent reinforcement learning,” 2019.

[51] G. Van Rossum and F. L. Drake, *Python 3 reference manual*. CreateSpace, 2009.

[52] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, “Rllib: Abstractions for distributed reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 3053–3062.