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

João Dias
Instituto Superior Técnico, University of Lisbon
Lisbon, Portugal

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

1 Introduction

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 [3] or circulatory system [4], [15]. To achieve the ideal goal of reducing pollution, it is crucial to monitor it at fine granularity and with the most detail possible.

Since it is impossible to cover every piece of land, we approximate the measurements by forecasting pollution in areas without sensors. Also, solutions with mobile sensors bring new advantages, such as better coverage and decision based on real-time. 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 explained before, it is crucial to ensure that these forecasts are reliable, i.e. to have a way of reducing the uncertainty. [28] 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) [22] using tools from [25].

This thesis seems to be the first to consider this pipeline, using [28] 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, we present a sequential order placement greedy algorithm that uses static sensors. To deal with mobile sensors we used a Reactive algorithm and one inspired on 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.

2 Background

2.1 Markov Decision Processes

In [40] a Markov Decision Process (MDP) 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 (1).

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

where s_t and a_t are the state and action respectively at time step t. The policy maps each state to the possible actions, being the optimal policy the one with the maximum accumulated discounted expected reward.

2.2 Reinforcement Learning

In this section, RL will be addressed based on [35]. In an environment described by an MDP one or more agents interact with the environment and take actions at each time step that may modify it. RL solves an MDP without information about the transition probabilities and the reward known a priori to maximize the accumulated discounted expected reward. For high-dimensional state spaces, Deep Reinforcement Learning (DRL) [27] can be used since it uses deep neural networks to deal with the complexity of the environment.

2.3 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 uses the estimates of every time step individually to update the value function as following

$$V(s_t) \leftarrow V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2}$$

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.

On-policy TD learning learns the action-value function Q directly from the current policy, being State Action Reward State Action (SARSA) the most well-known on-policy method. Instead of using the policy directly, in off-policy approaches such as Q-learning [40], 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 algorithms also use techniques of exploitation to choose actions based on the current policy, and exploration to explore the unknown seeking different paths prioritizing future rewards.

2.4 Policy gradient methods

Policy gradient methods [35] 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

$$\theta_{t+1} \leftarrow \theta_t + \alpha\gamma^t R \nabla \ln \pi(a_t|s_t, \theta_t) \tag{3}$$

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

2.5 Actor-Critic Models

To use both the advantages of action-value methods such as TD-learning and policy-based ones, we can use actor-critic methods [35], where the actor is responsible for learning the policy and the critic responsible for learning the value function. Therefore, we can use the state-value function as a baseline and apply the update in (4).

$$\theta_{t+1} \leftarrow \theta_t + \alpha A(s_t, a_t) \nabla \ln \pi(a_t|s_t, \theta_t) \tag{4}$$

$$\begin{aligned} A(s_t, a_t) &= r_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \\ &= Q(s_t, a_t) - \hat{v}(s_t) \end{aligned} \tag{5}$$

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

Other algorithms such as Trust Region Policy Optimization (TRPO) [31] limit the range of the policy updates, since some are so big that can lead to unexpected results in the performance. PPO [32] 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 (6), 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 $1 - \epsilon$.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \tag{6}$$

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta]|(s_t)] \tag{7}$$

The loss function (7) 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. PPO can also take advantage of parallel actors to collect data used afterwards to create the surrogate loss.

3 Related Work

3.1 Static sensors

Sensor placement for monitoring issues has been studied for a long time, whether related to water pollution (e.g. [29]), structural health issues [2], or air pollution (e.g. [33]), with cases studies done in multiple cities [33], [34], [29], and China [17].

In [33] they presented a greedy algorithm that chooses sensor by sensor the one that provides the highest reward, with the second one considering also a general cost constraint. In [34], it was also proposed two greedy algorithms (one of them also considered the cost of maintenance, construction, and operations) for the fixed-location sensors that would choose the location sensor by sensor according to the reward, but using weights. Other greedy solutions considered a division into batches placed sequentially [11]. [14] proposed a hierarchical algorithm for leak detection. Another

recent paper [42] approaches indoor crowdsensing with sequential sensor placement using signal strength Gaussian distributions.

The study done by [16] 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. Submodularity allows them to use a greedy algorithm to achieve a near-optimal solution with an approximation guarantee from the optimal solution, and also to bound the difference between their results and the optimal solution as well as search for tighter bounds.

Some studies also used Genetic Algorithm (GA) to detect a sudden release of a chemical in a ventilation system [10] and for structural health monitoring [12]. A recent study [2] studied how to achieve faster convergence in fewer generations using neural networks that would update the set of variables used in GA.

[29] also had into consideration the uncertainty and noise that can come from the measurements, trying to minimize errors in water distributed networks. It used Non-dominated Sorting Genetic Algorithm II (NSGA-II) which finds the optimal solution in multi-objective domains. [1] also adopted NSGA-II for vibration detection to minimize the number of sensors placed and maximize the amount of information collected.

3.2 Mobile sensors

In the literature, mobile sensors appear as a possible solution to have better spatial-temporal coverage of some region. When all the agents are homogeneous it is easier to scale the system with a multi-agent architecture [30]. With Multi-agent Reinforcement Learning (MARL), we can achieve speedup improvements, deal with the curse of dimensionality, and if one agent fails, the others can substitute their work [5]. [21] used Weighted Relative Frequency Of Obtaining The Maximal Reward (MRFMR) to balance exploration and exploitation in MARL to faster convergence. This algorithm outperformed other independent learning algorithms, which also happened in [41] that used a learning automata-based MARL.

Used for crowdsensing with sensors in vehicles, [39] used a Deep Q-network (DQN) to achieve the maximum coverage in space and time in the regions of interest. The agent was responsible for all the vehicles and selected the ones that would provide the best coverage. [37] used a Double Deep Q-network (DDQN) solution, where the mobile sensors would be allocated to tasks with a time limit. The agent would be assigned a reward for each assignment. and each assignment would be an action. Some explored the monitoring problem having into account that vehicles have limited energy capacities. [19] used Deep Deterministic Policy Gradient (DDPG), which they refer to overcome DQN that is limited to small action spaces.

Asynchronous solutions such as IMPALA-based [20] can use multiple actors each with its Central Processing Unit (CPU) and one learner Graphics Processing Unit (GPU), achieving better speed than the normal CPU implementation.

Other frameworks [26] considered 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. [13] used 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, which was proved to outperform DDPG.

[7] used Graph Convolutional Cooperative Multi-agent Reinforcement Learning (GCC-MARL) to change the route of taxis or other for-hire vehicles without having a negative impact on the orders of the clients. They considered segments of the roads in a graph instead of cells in a grid. The environment was partially observable and the algorithm works with actor-critic, with centralized training and decentralized execution. Their algorithm shows to surpass Independent Actor-Critic (IAC) [36] with decentralized critic, Central-V that does not consider the influence of every agent in the global reward [9], and Learning Individual Intrinsic Reward (LIIR) that considers that every agent has an intrinsic reward [8]. [24] found that decentralized critics can have better performance in some domains since it would create 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.

Other modern studies in the literature used PPO or variations of it to learn policies for monitoring tasks and routing vehicles. [6] 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, [23] made use of PPO to cover information in water reservoirs while minimizing the uncertainty of measurements and avoiding obstacles.

4 Methodologies

The algorithms presented in this section enable us to address 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. Mobile sensors are expected to outperform by a great margin static sensors in the task of covering the areas that we give the most relevance. Then, we discuss the situations in which we should

use a reactive agent or RL, giving different importance to the pollution measurements and the uncertainty of these measurements.

4.1 Environment Design & Interaction

We used Python [38] to implement all the algorithms since it has all the tools and frameworks needed, and the RLlib [18] library for the Reinforcement Learning algorithm. The majority of the studies about sensor placement resort to simplifications of the environment to reduce computational complexity. We also made use of a grid shown in Figure 1 to represent the map of the region we want to study and where we perform the sensor deployment. It is composed of 20 cells in regions of interest, 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. Despite the fact that we can lose information with this discretization, we reduce the complexity of the computations.

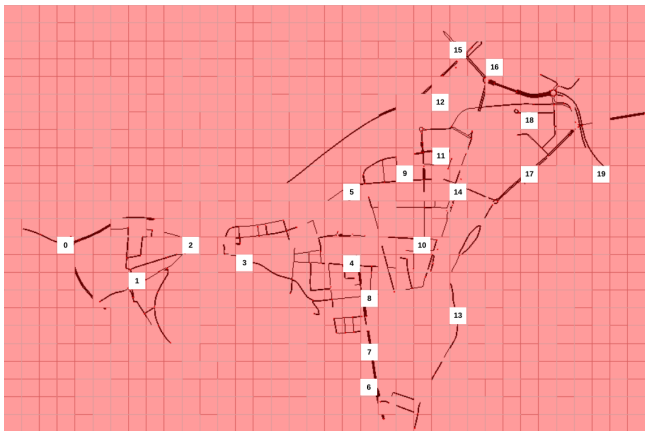

Figure 1. Simplification of the map using a grid.

The simulator enabled us to measure the pollution and the number of vehicles that were moving and contributing to the pollution of the entire cell each second. Since we considered a time step length of 10 minutes, the values that we use for each time step are the averages of the pollution and the traffic measurements of every second during the 10 minutes. We also used an average for the number of vehicles, 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. The values were also normalized with the formula in (8), because of their different magnitudes.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{8}$$

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.

Our pipeline is represented in Figure 2, with every component of the thesis pipeline and the flows between them. The components are represented as black boxes and although they 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.

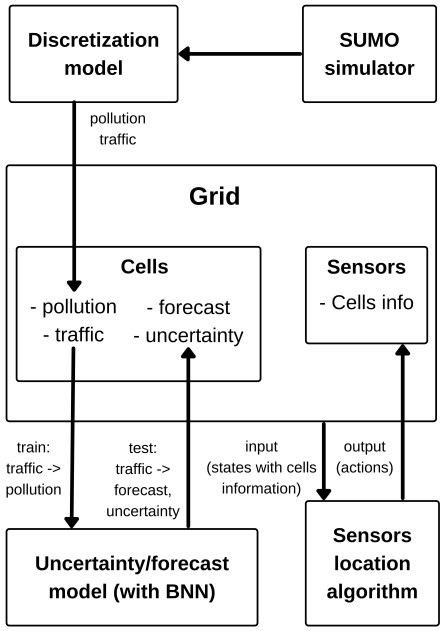

Figure 2. Environment scheme .

The pipeline uses the SUMO simulator with the discretization of data models from [25] to extract the data. We do this only once to speed up the process of training and testing since interaction with the SUMO simulator can take a long period of time. This data is used to train the Bayesian Neural Network (BNN) model [28], being the traffic the input and pollution the target. The 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 (9), chosen only based on practical trials and suitable for this specific case.

$$u_{t+1} = u_t * 3 + 0.02 \tag{9}$$

where u_t is the uncertainty at time step t. The uncertainty increases according to (9) 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.

The reward for each time step is similar for every algorithm and it is described by the formula in (10).

$$R(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t) \tag{10}$$

where δ_i is the relevance given to each of the variables, $\overline{p_t}$ 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 is only applicable to mobile sensors and 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 $|x_1 - x_2| + |y_1 - y_2|$.

4.2 Static Sensors

We started with a simple greedy algorithm for static sensor placement to have a good benchmark for comparison with the mobile sensors algorithms. 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 (11).

$$Gain = \delta_1 \cdot \overline{p} + \delta_2 \cdot \overline{u} \tag{11}$$

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 .1.

Algorithm .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 Reactive algorithm with mobile sensors

Since the approach with static sensors have certain limitations, mainly related to the area that they could cover, mobile sensors could be used to cover a bigger area. 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 Algorithm .2, the algorithm will get a gain associated with the cells that are currently being covered by sensors at each time step, which is used to compare results with the other algorithms.

The next cell to move to is decided 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.

Algorithm .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 (12), 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 (13) 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.

$$Gain(s_t, a_t) = \delta_1 \cdot \overline{p}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_{t-1}, s_t) \quad (12)$$

$$Gain_{potential}(s_t, a_t) = \delta_1 \cdot \overline{f}_t + \delta_2 \cdot \overline{u}_t - c \cdot d(s_t, s_{t+1}) \quad (13)$$

4.4 Reinforcement Learning algorithm with mobile sensors

Reinforcement learning is one of the main fields that aim for long-term results and where an agent learns how to act by exploring the environment and developing its model. It was chosen for this problem 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 variation.

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. It is described as:

- **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 (9) 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.

We used an actor-critic-based structure with PPO due to its simplicity, and its updates control mechanism. By looking at the pseudocode in .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 .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;
```

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.

5 Experiments & Evaluation

In this section, we will cover the experiments 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. We used Python and Matplotlib to simplify the visualization of information.

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.

5.1 Data Analysis

We considered the pollution and the traffic to be realistic enough for our experiments, which can be verified in Figures 3 and 4 where it is represented a single day mean values of data collected over 61 days, 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.

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.

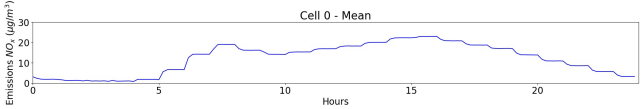

Figure 3. Mean values of NO_x emissions of a single day from data collected over 61 days for one cell.

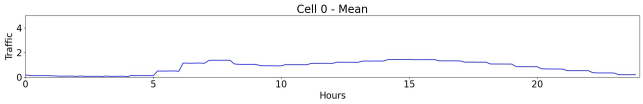

Figure 4. Mean values of traffic of a single day from data collected over 61 days for one cell.

To understand if the values are not the same every day, we plotted the standard deviations for each hour of the day corresponding to the same 61 days. The Figures 5 and 6 provide us good insights that the data is not constant and have temporal variation.

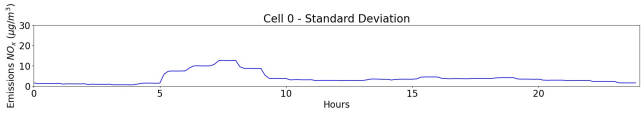

Figure 5. Standard deviation values of NO_x emissions of a single day from data collected over 61 days for one cell.

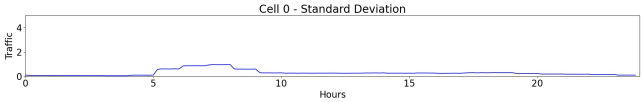

Figure 6. Standard deviation values of traffic of a single day from data collected over 61 days for one cell.

6 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 Figure 7.

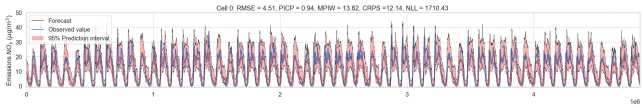

Figure 7. BNN output (real value, forecast, upper and lower bound) during 56 days for one cell.

Each point of Figure 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.

We proved the need for using sensors to cover regions with unreliable forecasts since our data 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 $\mu g/m^3$.

6.1 Architecture of the Reinforcement Learning algorithm

To choose the best parameters in the architecture of the Reinforcement Learning algorithm, we tried different combinations of them. We refer mainly to the number of layers and neurons, but other hyperparameters of the algorithm were also combined and tested to stick with the ones which provided better performance.

We understood 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. Other hyperparameters of the algorithm were also combined and tested to stick with the ones which provided better performance.

6.2 Pollution coverage and uncertainty reduction

It is important to understand how the algorithms perform to cover pollution and reduce uncertainty with different relevance given to pollution and uncertainty. This can be done by changing the parameters δ_1 and δ_2 in 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, being these parameters the relevance given to the covered pollution $\overline{p}_t$ and the reduction of uncertainty $\overline{u}_t$.

Each point of the 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. The Figures in 8, 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.

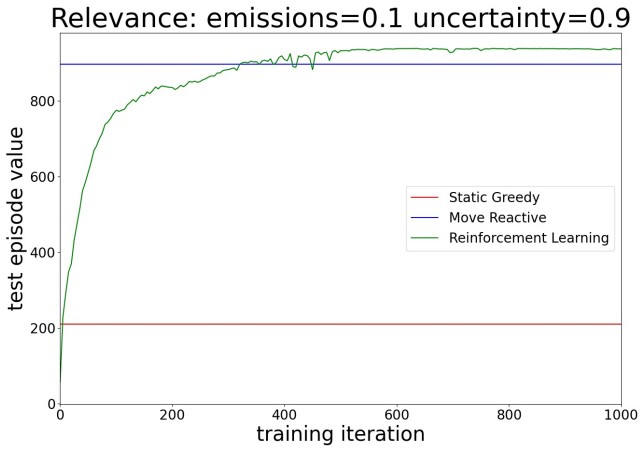

Figure 8. Comparison of the reward of the algorithms for 1 sensor.

The behavior of our algorithms related to the concrete values, i.e. the emissions and uncertainty, also followed our expectations and 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 in the pollution measurements in two consecutive time steps. The reduction of uncertainty is 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.

Figure 9 shows us the reward gathered on each time step of a complete episode by applying the RL algorithm for the case of the relevance of 0.1 in emissions and 0.9 in uncertainty. The episode corresponds to the one that gave us the best results. We figured out that the difference between RL and reactive is small, being the reward peaks are located in the middle of each day when pollution and uncertainty are bigger.

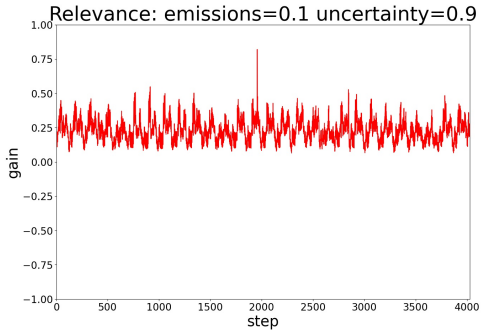

Figure 9. Step reward in a full episode of RL.

We verified that the reduction of uncertainty was significant for certain cells, such as cell 12, since these were the cells with the most uncertainty, and therefore a priority for

the sensors to move to. This reduction is showed by comparing Figure 11 with 10. On the other hand, we do not see any reduction in some cells since it some of them have much less uncertainty and not a priority for the sensors.

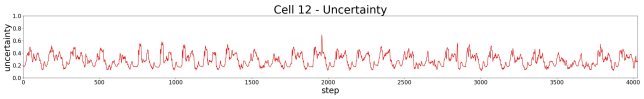

Figure 10. Normalized uncertainty in the test environment for 4029 steps without sensors for cell 12.

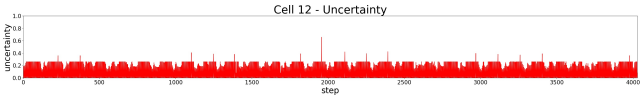

Figure 11. Normalized uncertainty in the test environment for 4029 steps with sensors for cell 12.

Figures 12 and 13 represent the movement of a single sensor and each point represents the cell where the sensor is present at a certain time step. They show us that the sensors are the majority of the time in certain cells, which was expected since these cells are the ones with the biggest values of pollution and uncertainty. We can also notice that the RL algorithm still moves fewer times than the reactive one.

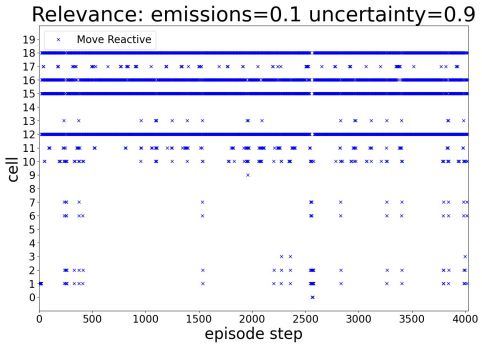

Figure 12. Sensor movement of the reactive algorithm.

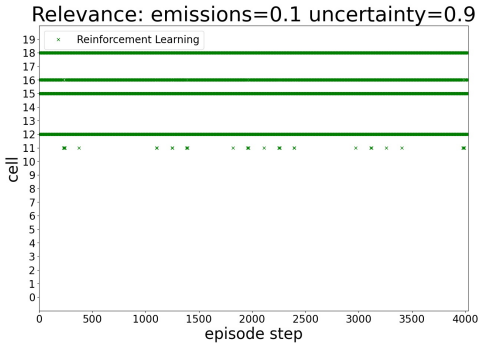

Figure 13. Sensor movement of the RL algorithm.

6.3 Variation in the number of sensors

We fixed the importance of pollution at 0.1 and uncertainty at 0.9 to compare the performances when using more sensors in the environment. The RL algorithm outperformed the other algorithms with 1 and 2 sensors as shown in Figures 8 and 14. However, if we increased the number of sensors, we did not achieve better performance as shown in Figure 15. 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.

Also, although we get better performance by adding more sensors, the increase in performance does not follow.

Figure 14. Comparison of the algorithms with 2 sensors.

Figure 15. Comparison of the algorithms with 3 sensors.

6.4 Running Time

We understood 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. We needed 4 h 33 m 54 s for 1 sensor, 7 h 41 m 32 s for 2 sensors, and 12 h 21 m 42 s for 3 sensors.

7 Conclusion

In this project, 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. 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.

We could easily understand that, as expected, the use of mobile sensors was a great advantage and led to substantially better performance, when compared to static sensors. 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, since a time step was strongly related to the following one. 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.

For future work, we could test the algorithm with other baselines besides the ones mentioned before. Also, as in this case, the pollution measurement was Nitrogen Oxides (NOx), it would be interesting to find out if the algorithm gets similar results with different compounds. We could also try the algorithm in scenarios with reduced or no communication and adjust the simulator to include more variables, such as the wind or the rain to make the propagation of the pollution more realistic.

Acknowledgments

The student supervision was under the responsibility of professor Tiago Veiga on 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 Thesis.

A special thanks to these two professors, who were available to support me with the project 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.

References

[1] Haichao An, Byeng D. Youn, and Heung Soo Kim. 2021. A methodology for sensor number and placement optimization for vibration-based damage detection of composite structures under model uncertainty. *Composite Structures* 279 (2021), 114863. https://doi.org/10.1016/j.compstruct.2021.114863

[2] Munni Rani Banik and Tonmoy Das. 2020. Application of Neuro-GA Hybrids in Sensor Optimization for Structural Health Monitoring. In *Proceedings of the International Conference on Computing Advancements* (Dhaka, Bangladesh) *(ICCA 2020)*. Association for Computing Machinery, New York, NY, USA, Article 34, 7 pages. https://doi.org/10.1145/3377049.3377131

[3] Jonathan A Bernstein, Neil Alexis, Charles Barnes, I Leonard Bernstein, Andre Nel, David Peden, David Diaz-Sanchez, Susan M Tarlo, and P Brock Williams. 2004. Health effects of air pollution. *Journal of allergy and clinical immunology* 114, 5 (2004), 1116–1123.

[4] Robert D Brook. 2008. Cardiovascular effects of air pollution. *Clinical science* 115, 6 (2008), 175–187.

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[6] Jingxi Chen, Amrish Baskaran, Zhongshun Zhang, and Pratap Tokekar. 2021. Multi-Agent Reinforcement Learning for Visibility-based Persistent Monitoring. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2563–2570. https://doi.org/10.1109/IROS51168.2021.9635898

[7] Rong Ding, Zhaoxing Yang, Yifei Wei, Haiming Jin, and Xinbing Wang. 2021. Multi-Agent Reinforcement Learning for Urban Crowd Sensing with For-Hire Vehicles. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488713

[8] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. (2019).

[9] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[10] Jun Gao, Lingjie Zeng, Changsheng Cao, Wei Ye, and Xu Zhang. 2018. Multi-objective optimization for sensor placement against suddenly released contaminant in air duct system. *Build. Simul.* 11, 1 (Feb 2018), 139–153. https://doi.org/10.1007/s12273-017-0374-z

[11] Fateme Ghayem, Bertrand Rivet, Rodrigo Cabral Farias, and Christian Jutten. 2021. Robust Sensor Placement for Signal Extraction. *IEEE Transactions on Signal Processing* 69 (2021), 4513–4528. https://doi.org/10.1109/TSP.2021.3099954

[12] Guilherme Ferreira Gomes, Sebastiao Simões da Cunha, Patricia da Silva Lopes Alexandrino, Bruno Silva de Sousa, and Antonio Carlos Ancelotti. 2018. Sensor placement optimization applied to laminated composite plates under vibration. *Struct. Multidiscip. Optim.* 58, 5 (Nov 2018), 2099–2118. https://doi.org/10.1007/s00158-018-2024-1

[13] Jingzhi Hu, Hongliang Zhang, Lingyang Song, Robert Schober, and H. Vincent Poor. 2020. Cooperative Internet of UAVs: Distributed Trajectory Design by Multi-Agent Deep Reinforcement Learning. *IEEE Transactions on Communications* 68, 11 (2020), 6807–6821. https://doi.org/10.1109/TCOMM.2020.3013599

[14] Zukang Hu, Wenlong Chen, Beqing Chen, Debao Tan, Yu Zhang, and Dingtao Shen. 2021. Robust Hierarchical Sensor Optimization Placement Method for Leak Detection in Water Distribution System. *Water Resour. Manage.* 35, 12 (Sep 2021), 3995–4008. https://doi.org/10.1007/s11269-021-02922-3

[15] Marilena Kampa and Elias Castanas. 2008. Human health effects of air pollution. *Environmental pollution* 151, 2 (2008), 362–367.

[16] Andreas Krause, Ajit Singh, and Carlos Guestrin. 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research* 9, 2 (2008).

[17] Xiuhong Li, Meiying Sun, Yushuang Ma, Le Zhang, Yi Zhang, Rongjin Yang, and Qiang Liu. 2021. Using Sensor Network for Tracing and Locating Air Pollution Sources. *IEEE Sensors Journal* 21, 10 (2021), 12162–12170. https://doi.org/10.1109/JSEN.2021.3063815

[18] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3053–3062.

[19] Chi Harold Liu, Zheyu Chen, and Yufeng Zhan. 2019. Energy-Efficient Distributed Mobile Crowd Sensing: A Deep Learning Approach. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1262–1276. https://doi.org/10.1109/JSAC.2019.2904353

[20] Chi Harold Liu, Zipeng Dai, Haoming Yang, and Jian Tang. 2020. Multi-Task-Oriented Vehicular Crowdsensing: A Deep Learning Approach. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 1123–1132. https://doi.org/10.1109/INFOCOM41043.2020.9155393

[21] Hui Liu, Zhen Zhang, and Dongqing Wang. 2020. WRFMR: A Multi-Agent Reinforcement Learning Method for Cooperative Tasks. *IEEE Access* 8 (2020), 216320–216331. https://doi.org/10.1109/ACCESS.2020.3040985

[22] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO, In The 21st IEEE International Conference on Intelligent Transportation Systems. *IEEE Intelligent Transportation Systems Conference (ITSC)*. https://elib.dlr.de/124092/

[23] Samuel Yanes Luis, Daniel Gutiérrez Reina, and Sergio Toral. 2022. Maximum Information Coverage and Monitoring Path Planning With Unmanned Surface Vehicles Using Deep Reinforcement Learning. (2022).

[24] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. 2021. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402* (2021).

[25] Mykhaylo Marfeychuk. 2020. Learning Control Policies in Smart Cities from Physical Data. (2020).

[26] Shaofeng Meng and Zhen Kan. 2021. Deep Reinforcement Learning-Based Effective Coverage Control With Connectivity Constraints. *IEEE Control Systems Letters* 6 (2021), 283–288. https://doi.org/10.1109/LCSYS.2021.3070850

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning - Nature. *Nature* 518 (Feb 2015), 529–533. https://doi.org/10.1038/nature14236

[28] Abdulmajid Murad, Frank Alexander Kraemer, Kerstin Bach, and Gavin Taylor. 2021. Probabilistic Deep Learning to Quantify Uncertainty in Air Quality Forecasting. *Sensors* 21, 23 (2021). https://doi.org/10.3390/s21238009

[29] Marcos Quiñones Grueiro, Cristina Verde, and Orestes Llanes-Santiago. 2019. Multi-objective sensor placement for leakage detection and localization in water distribution networks. In *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*. 129–134. https://doi.org/10.1109/SYSTOL.2019.8864746

[30] Yara Rizk, Mariette Awad, and Edward W. Tunstel. 2018. Decision Making in Multiagent Systems: A Survey. *IEEE Transactions on Cognitive and Developmental Systems* 10, 3 (2018), 514–529. https://doi.org/10.1109/TCDS.2018.2840971

[31] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *CoRR* abs/1502.05477 (2015). arXiv:1502.05477 http://arxiv.org/abs/1502.05477

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[33] Chenxi Sun, Victor O. K. Li, Jacqueline C. K. Lam, and Ian Leslie. 2019. Optimal Citizen-Centric Sensor Placement for Air Quality Monitoring: A Case Study of City of Cambridge, the United Kingdom. *IEEE Access* 7 (2019), 47390–47400. https://doi.org/10.1109/ACCESS.2019.2909111

[34] Chenxi Sun, Yangwen Yu, Victor O.K. Li, and Jacqueline C.K. Lam. 2018. Optimal Multi-type Sensor Placements in Gaussian Spatial Fields for Environmental Monitoring. In *2018 IEEE International Smart Cities Conference (ISC2)*. 1–8. https://doi.org/10.1109/ISC2.2018.8656676

[35] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning An Introduction* (2 ed.). The MIT Press.

[36] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[37] Xi Tao and Wei Song. 2020. Task Allocation for Mobile Crowdsensing with Deep Reinforcement Learning. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–7. https://doi.org/10.1109/WCNC45663.2020.9120489

[38] Guido Van Rossum and Fred L Drake. 2009. *Python 3 reference manual*. CreateSpace.

[39] Chaowei Wang, Xiga Gaimu, Chensheng Li, He Zou, and Weidong Wang. 2019. Smart mobile crowdsensing with urban vehicles: A deep reinforcement learning perspective. *IEEE Access* 7 (2019), 37334–37341.

[40] Christopher John Cornish Hellaby Watkins. 1989. Learning from delayed rewards. (1989).

[41] Zhen Zhang, Dongqing Wang, and Junwei Gao. 2021. Learning Automata-Based Multiagent Reinforcement Learning for Optimization of Cooperative Tasks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 10 (2021), 4639–4652. https://doi.org/10.1109/TNNLS.2020.3025711

[42] Yang Zhen, Masato Sugasaki, Yoshihiro Kawahara, Kota Tsubouchi, Matthew Ishige, and Masamichi Shimosaka. 2021. 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* (Beijing, China) *(SIGSPATIAL '21)*. Association for Computing Machinery, New York, NY, USA, 301–304. https://doi.org/10.1145/3474717.3483964