Integrating Reinforcement Learning and Distributed Perception Networks for Mobile Sensor Control

Dissertation Thesis for the
Research Master of Science Degree in
Artificial Intelligence - Gaming Track

By Corrado Grappiolo - 0639818
Supervisor: Shimon Whiteson
Co-supervisors: Bram Bakker, Gregor Pavlin

August 15, 2008

Abstract

This thesis describes an approach that uses reinforcement learning together with distributed perception network fusion systems in order to perform mobile sensor control. A case study of such mobile sensors is the chemical leak detection problem. The proposed solution deals with partial observability of the true state and makes use of linear function approximation to learn a value function that maps belief-action pairs into values, distributed perception networks to create a correct, robust and computationally efficient system for the inference of gas leak given sensor observations, and bayesian inference to estimate the leak location. Results show that by keeping a continuous belief represented through entropies and representing information about both single cell and aggregation of cells, the system can learn successfully; furthermore, studies about the influence that the distributed perception networks give to the whole system are done. Discussions and future developments are also provided.

Contents

Acknowledgements

I would really like to thank Shimon for the supervision and its efforts to understand my Italanglish, Bram and Gregor for the valuable co-supervision.

Thanks to Patrick de Oude, who co-co-supervised me in the last months.

Many thanks to Mete Çakman, schoolmate, roommate, junglistic man and especially good friend.

Thanks to Hugin Expert for their Hugin Lite software, Lisa Cluster for their computational support, special thanks to Lilit Axner from the Lisa Cluster technical support.

Thanks also to Nuffic for the scholarship, Rogier Wouters and Ben Kröse for having given me a job; they helped me maintaining my academic life.

1 Introduction

Mobile sensor control is a problem in which autonomous agents, by efficiently moving around and using sensors, try to reduce uncertainty about some phenomenon.

A real life example of such a problem used to validate the thesis' achievements is the chemical leak detection problem. In a typical scenario, a highly industrialized area composed of many chemical processing factories, chemical waste depots, etc. is located next to a high density urban one. If a leak occurs, it is desirable to detect it as fast as possible, in order to stop it and minimize chemical contamination, which is manifested in the form of chemical gas that diffuses in the air. Three issues are present:

1. it is not known where the leak is,
2. the leak must be found in the shortest time possible, and
3. it must be found without error.

To cope with such a hazardous situation, some monitoring should be performed. Sensors can be placed in the environment and, from their readings, inference about the leak location can be done. Unfortunately, centalised control of this process cannot be done: apart from the fact that the computational load grows significantly with respect to the complexity of the scenario (size of the area monitored, volume of readings that need to be processed per time, etc), the fact that centralized control must have a full and exact knowledge about every sensor makes this approach infeasible.

A solution is the organization of such sensors in distributed networks that independently process raw sensor data and then share (fuse) their results. The process of combining distributed sensors can be achieved through the use of distributed perception networks (DPNs), a technology that is based on causal models and, through the fusion of distributed sensor perceptions into a unique network, generates high level inference about some phenomenon. Causal models can successfully represent the relationship between causes and effects: in the chemical leak problem, for example, sensors can perceive variations of ionization of the air (effects) which is caused by the gas contamination (cause). The causality is represented in a DPN in terms of causal Bayesian networks, where the belief of the root node, which represents the gas belief, is inferred from observation of the leaf nodes, that is, the sensor readings. Through the highest level of reasoning about gas contamination, then, it is possible to find the leak location. The distributed network organization and the fusion process provides a level of abstraction that makes DPNs robust with respect to sudden changes in composition of the network of sensors, that is, sensors can be added to the network or removed from it without hassle.

In real life, however, it is not possible to install every kind of sensor for all kind of chemical that are used and stored in all the factories: this requires huge resources, and every time a factory decides to change its production, to use some new chemical, it will require the installation of new sensors specific to that kind of new chemical. A solution to this problem is the use of more general purpose sensors, able to deal with effects that are generated by different causes. For example, different chemical leaks might produce a similar variation in ionization of the air. Unfortunately, the more general a sensor is, the more faulty it becomes. A tradeoff is given by a combined use of more general sensors placed in the environment (called fixed sensors) with more specific ones, which can be moved and take part it the fusion into distributed network of sensors present in the environment, exploiting in this way the capabilities of DPNs to be robust with respect to the composition of the network of sensors. Unfortunately, a DPN is just a fusion system that can efficiently infer about hidden events given evidence. It cannot decide where and when to move the mobile sensors. How, where and when the sensor should be moved represents a sequential decision problem.

Sequential decision problems can be successfully solved using reinforcement learning (RL), an artificial intelligence technique that has its roots in the study of animal trial-and-error behaviour under the influence of external stimuli. By using a helicopter agent equipped with sensors and giving it rewards according to the actions it performs, it is possible for the agent to learn a control policy, that is, a mapping from states into actions. Intuitively, the policy sought is the one that maximises the discounted cumulative reward.

Markov decision processes (MDPs) are the simplest type of RL problems for which the transition function of the states of the environment and the reward function are dependent only on the current state. One way to solve MDP problems is by finding an optimal value function that maps states into values. The result is a policy function which maps states to actions.

In a simple MDP the agent has perfect knowledge about the current state of the environment and performs actions conditioned to it. The chemical leak detection task, unfortunately, is not a simple MDP. The biggest obstacle to the agent is represented by its partial observability about the leak position: the agent does not know the state, and once it performs an action, the environment returns only an observation about the next state. Instead of dealing with states of the environment, the agent has to deal with a belief state, that is, a probability distribution over states; the task of the agent, now becomes to find a policy which maps belief states to actions. Problems of this kind are called partially observable Markov decision processes (POMDPs). In this thesis' scenario, the belief state is represented by the possible leak positions. By its integration with DPNs, however, the helicopter can keep a reliable belief and update it every time the DPNs perform inference.

Typically, POMDP problems are solved via planning, that is, an offline computation of the value function for all possible belief configurations; this is based on the assumption that the agent has a perfect knowledge of the model of the environment. In the chemical leak detection problem, the helicopter can have a perfect knowledge of the environment, hence planning seems to be applicable.

The problem with planning, however, is that it becomes intractable for large problems, and the chemical leak detection is one of them; the only way the helicopter can learn how to find the leak is by using a model-free approach, that is, by performing actions and directly interact with the environment.

Model free approaches are not usually applied to POMDP problems; however, when they are, they are usually based on techniques that consider the belief as a secondary characteristic of the problem: heuristics that aim to approximate the state. That is, they do not maintain the belief anymore. In the chemical leak task, instead, the uncertainty is central to the problem and cannot be considered secondary.

In the chemical leak detection problem, instead, since the helicopter can have a perfect knowledge of the environment, it can maintain beliefs; this turns the problem into a continuous space MDP, which allows the application of model-free methods and hence it can be solved.

Given that the helicopter must keep a belief about the leak location, the issue, now, is about its representation. Two approaches are possible: first, the belief can be discretized in order to make the belief space finite. Second, a continuous representation can be maintained. Moreover, different ways to represent the continuous belief are defined and compared to each other.

The next issue is the choice of a value function representation. Since a model-free approach is used, the value function will be based on q-values of belief-action pairs. Two methods for q-value learning are studied: one is based on the storage of discretized belief and action pairs in a table together with their value functions; the second one, instead, makes use of the continuous belief representation and action pairs and assumes a linear mapping to value functions.

It will be shown that the table-based approach with discretized belief is infeasible, since the size of the table grows exponentially with respect to the environment size; the linear function approximation approach can perform well and is robust with respect to the increase of size of the environment but is dependent on the continuous belief state representation used.

Other issues related to the leak belief inference arise: the agent must perform actions based on its belief about the leak location; it seems intuitive that, the more reliable the inference about the leak location is, the better the control policy. The same problem about centralized control discussed previously arises: the helicopter cannot know everything about the sensor models. The need for DPN integration becomes essential: this allows the helicopter not to know anything about the fixed sensors, from their models to their position in the environment; it just performs actions according to its own belief which is updated given the results of the inference of the DPNs. Experiments aiming to analyze the level of depencency between control policy and DPNs are also done.

The thesis is organized as follows: Chapters 2, 3 and 4 provide the basic backgrounds about Bayesian reasoning methods, distributed perception networks and reinforcement learning techniques needed to understand the work done later on. Chapter 5 analyzes the centralized approach for the chemical leak problem and ends with the definition of the DPN model structure which will the used in Chapter 6, when the integration of DPNs and RL is described. Experimental results which investigate different value function and belief representations, level of dependency between mobile agent (RL) and DPNs, robustness with respect to large environments are shown in Chapter 7. In Chapter 8 a brief description about the software integration between the helicopter agent (RL) and the DPN Toolkit, a fusion system based on DPNs developed by Universiteit van Amsterdam, is provided. Chapter 9 provides valuable discussions about the thesis work and achievements, together with related works and paths for future developments. Chapter 10 concludes.

2 Background: Bayesian Reasoning

This chapter provides the backgrounds about Bayesian reasoning concepts that are used in the rest of the thesis. Basic knowledge of probability theory and statistics are assumed.

The main problem that agents face during their execution task is given by the fact that they almost never have a complete view of the environment and hence the complete knowledge of the state they are in. This means that the agents have to deal with the problem of reasoning with uncertainty and knowledge representation. However the agent, through sensors, can gather data (perceptions) of the environment and use it in order to update (infer) its hypothesis. The hypothesis represent the belief of the agent.

Bayesian reasoning is a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data. It provides a quantitative approach to weighing the evidence supporting alternative hypotheses [9].

Before defining Bayes theorem and Bayesian networks, few notions of probability theory and statistics are provided.

2.1 Few Notions of Probability Theory and Statistics

In statistics, given a sample space Ω of a probability space $(\Omega, \mathcal{F}, P)$, where $\mathcal{F}$ is an algebra over Ω and $P : \mathcal{F} \rightarrow [0, 1]$ is a probability function, a random variable X is a function from the sample space to a real value: $X : \Omega \rightarrow \mathbb{R}$. Random variables can be of two kinds: continuous or discrete. A random variable is said to be continuous if its possible values extend over a continuum; it is said to be discrete if its possible values are partitioned into intervals. A boolean random variable, in the end, is a special case of discrete random variable and can assume only two values, $true$ or $false$. The domain D_X of a random variable X is the set of possible values it can have; the domain of a boolean random variable hence is represented by $D_X = \{x, \overline{x}\}$.

The notation $P(X = x_1)$ indicates the prior probability that the random variable X has value x_1; $P(X)$ denotes the prior probability distribution of the variables X according to the values of its domain. Given n random variables $X_1, \ldots X_n$, the joint probability distribution $P(X_1, \ldots X_n)$ assigns probabilities to all possible configurations of the domain of all possible variables; the joint probability $P(x_1, \ldots x_n)$ is computed by using the chain rule

$$P(x_1, \ldots x_n) = P(x_1)P(x_2|x_1) \ldots P(x_n|x_1, \ldots x_{n-1}) \tag{1}$$

The notion $P(x|y)$ indicates the conditional probability that x is manifested given that y holds.

From this, the conditional independence is defined: two random variables X and Y are conditionally independent given a randon variable Z if and only if

$$P(X|Y, Z) = P(X|Z) \tag{2}$$

and, analogously, $P(Y|X, Z) = P(Y|Z)$.

2.2 Bayes Theorem

The idea, as said previously, is to update the probability of hypothesis given observed data. Some notation is given. h is used to represent a possible hypothesis, where H is used to represent the hypothesis space, that is, the set of all the hypothesis that form the belief. Intuitively, the hypothesis space H is composed of n mutually exclusive hypothesis $h_i, i = 1, \ldots, n$, that is, the occurrence of any one of them precludes any of the others[1].

D represents the set of all data that can be observed. Before receiving $d \in D$, which is a configuration of observations for different variables, $d = \{d_1, d_2, \dots, d_n\}$, the agent has already some degree of knowledge of its belief: for all $h_i \in H, i = 1, \dots, n$, the prior probability $P(h_i)$ is defined; this represents the chance that hypothesis h_i holds. $P(H)$ is the probability distribution of the hypothesis space. Similarly, $P(D)$ is the prior probability distribution of all the data, and $P(d)$ is the prior probability of the single observation d.

The expression $P(h_i, d)$ represents the joint probability that both propositions h_i and d are occurring at the same time; analogously, $P(H, D)$ represents the joint probability distribution over the spaces H and D. The joint probability can be calculated through the chain rule (1) and is defined as follows:

$$P(h_i, d) = P(d|h_i)P(h_i) \qquad \forall i = 1, \dots n \tag{3}$$

where $P(d|h_i)$ represents the conditional probability that d is manifested given that the hypothesis h_i holds. Analogously

$$P(h_i, d) = P(h_i|d)P(d) \qquad \forall i = 1, \dots n$$

therefore

$$P(h_i|d)P(d) = P(d|h_i)P(h_i) \qquad \forall i = 1, \dots n$$

If $P(d) > 0, \forall d \in D$, the formula

$$P(h_i|d) = \frac{P(d|h_i)P(h_i)}{P(d)} \qquad \forall i = 1, \dots, n, P(d) > 0 \tag{4}$$

is obtained. This formula is also called Bayes rule [2]. Since the hypothesis space H is composed of n mutually exclusive hypothesis $h_i, i = 1, \dots, n$, the prior $P(d)$ can also be expressed through the total probability theorem

$$P(d) = \sum_H P(d|H)P(H) = \sum_{i=1}^{n} P(d|h_i)P(h_i) \tag{5}$$

leading to the marginalized version of Bayes theorem:

$$P(h_i|d) = \frac{P(d|h_i)P(h_i)}{\sum_{j=1}^{n} P(d|h_j)P(h_j)} \tag{6}$$

To conclude, given (4) and (6), it is possible to perform inference about the hypothesis space given observations.

2.3 Bayesian Networks

Bayesian networks (BNs) are a graphical structure for statistical inference on random variables; their computational power is given by the fact that BNs highlight the essential relationships among random variables, that is, their conditional dependence and independence.

Definition: a Bayesian network (BN) is a graph in which the following holds:

[1]This assumption of mutually exclusiveness can be clearly seen in the chemical leak problem of this thesis: the helicopter keeps hypothesis of possible leak locations and it is assumed that only one leak is occurring; this means that the hypothesis "the leak is in location A" and "the leak is in location B" are mutually exclusive.

1. A set of random variables makes up the nodes of the network;
2. A set of directed links or arrows connects pairs of nodes. The intuitive meaning of an arrow from node X to node Y is that X has a direct influence on Y;
3. Each node has a conditional probability table that quantifies the effects that the parents have on the node. The parents of a node are all those nodes that have arrows pointing to it;
4. The graph has no directed cycles (hence is a directed, acyclic graph, or DAG) [8].

An example of BN is given in Figure 1. Five nodes are present: X_1, X_2, X_3, X_4 and X_5. X_1 represents the root node, X_5 the leaf node, whilst X_2, X_3 and X_4 are the intermediate nodes of the network. Furthermore, X_1 is the parent node of X_2 and X_3, which are, at the same time, parent nodes of X_4. Analogously, X_4 is parent of X_5. More generally, the parent node of a node X_i is represented by $parent(X_i)$.

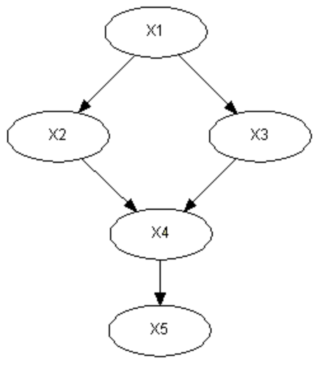

Figure 1: An example of BN with five nodes.

It is usually easy for a domain expert to decide what direct conditional dependence relationships hold in the domain. Once the topology of the belief network is specified, the conditional probabilities for the nodes that participate in direct dependencies are specified and used to compute any other probability values [8]. Mostly, BNs describe causal relationships among variables: from an initial cause, represented as the BN root node, a chain of random variables representing dependent events are defined, until the leaf nodes. This is due to the fact that humans tend to reason about events in terms of causes-effects. From now on, the attention is focused on causal Bayesian networks.

D-separation is a graphical notion that is used to represent conditional independence between random variables; this is a powerful property that gives the possibility to define independence of nodes given their parents and thus to efficiently represent the joint probability distribution between variables. The question now is whether it is possible to define independence of a set of nodes X from a set of nodes Y given a set of evidence nodes E.

A set of nodes E d-separates two sets of nodes X and Y if every undirected path from a node in X to a node in Y is blocked given E. A path is blocked given a set of nodes E if there is a node Z on the path for which one of three conditions holds:

(D1) Z is in E and Z has one arrow on the path leading in and one arrow out;

(D2) Z is in E and Z has both path arrows leading out;

(D3) Neither Z nor any descendant of Z is in E, and both path arrows lead in to Z [8].

If every undirected path from a node in X to a node in Y is d-separated by E, then X and Y are conditionally independent given E. A graphical explanation of d-separation is given in Figure 2.

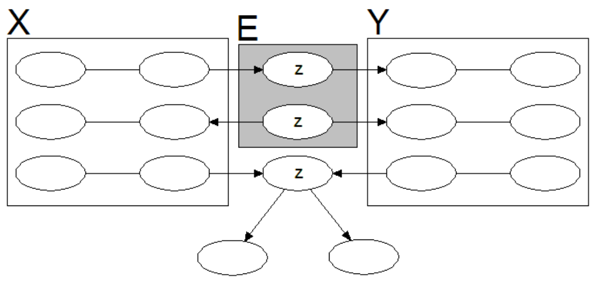

Figure 2: Three cases that define d-separation. The original image can be found in [8].

It is now possible to define the joint probability distribution of n random variables $X_1, \ldots X_n$ of a BN. Thanks to d-separation and the consequent conditional independence of nodes given by (2), some variables can be ruled out. The final joint probability distribution is reduced to

$$P(x_1, \ldots x_n) = \prod_{i=1}^{n} P\left(x_i | parent(x_i)\right) \tag{7}$$

Through Bayes rules (4) and (6) and the joint probability equation (7), it is now possible to perform inference on BNs. Given some evidence, that is, observations of the values that some random variables have, it is possible through Bayes rule to infer about the probability that other nodes assume values in their domain.

3 Background: Distributed Perception Networks

Causal Bayesian networks are a powerful tool to describe causal relationships among variables and infer about hidden variables given observation of some other variables; for example, given evidence of leaf nodes, through inference, it is possible to calculate the probability distribution of values defined by the hidden root node. A causal BN can be used to describe the effects generated by a chemical leak: the cause, that is the chemical leak, represents the root node of the network, and the sensor readings, that is the effects, represent the leaf nodes of the network. Between these two type of nodes, hidden variables representing the gas spread and sensor models are present.

A problem with using centralized inference, that is a monolithic BN that processes the whole information of the environment, is that it is highly dependent on the problem complexity. In real life, the sensors can be replaced, moved, modified, and so on; the monolithic BN must be updated everytime these events occur, otherwise it will represent an incorrect model and thus the inference will be wrong. Furthermore, if the Bayesian network used to represent the whole history of sensor readings is a quasi-static one[2], that is, all new sensor readings are represented as new instantiated leaf nodes of the network, the monolithic BN will be constantly modified. This makes centralized inference impractical. Decentralized inference provides a more flexible solution to cope with a possibly constant need to change the network structure.

D-separation described in Section 2.3 plays an important role: by identifying nodes that d-separate various nodes in a BN, it is possible to create partitions and distribute the control to them: ability to reorganize the local network by adding leaf nodes and to perform local inference about the partition's root given observations about partition's leaves. Then, by sharing (fusing) only the result of the local inference, the system becomes robust to changes. Clearly, different level of abstractions are present. The approach just illustrated is performed by distributed perception networks, a fusion system defined as follows:

Definition: Distributed perception networks (DPNs) are a multi-agent based approach to robust and efficient fusion of heterogeneous data and information that avoids centralized fusion systems. Its approach features simple building blocks implemented through agents of different types, which can dynamically be organized into fusion systems that can cope with partially unpredictable aspects of the real world [4].

In a DPN, two kinds of cooperative agents are present: at the lowest level, sensor agents process raw sensory data and perform a so called low level inference. At higher levels, fusion agents perform inference using the information provided by sensor agents or other fusion agents and pass their result to higher levels, up to the highest fusion agent.

Both sensor and fusion agents can perform their inference through arbitrary complex local BNs that represent basic world modeling building blocks [4]; the only constraint of DPNs, however, is given by the fact that each agent's local BN has only one service root node. The agent's service root node is the concept the agent provides to other agents. In a DPN, all the agents are organized as nodes of a simple tree; furthermore, each agent can have at most one agent-node as parent.

The inference performed in the agent's local BN is done according to the equations (4), (6) and (7) described in Sections 2.2 and 2.3, having the initial probability distribution on the local BN's root node set to uniform. If the priors about the local BN's root node are set to uniform, then the fusion system will compute exactly the same inference of the corresponding monolithic BN [5].

[2]The definition of quasi-static BNs goes beyond the scope of this thesis; more information about quasi-static Bayesian networks can be found in [5], [16], [17] and [18].

The fusion process of two agents A and B works as follows: let $\mathcal{L}_B$ be the leaf nodes or input concepts of the agent B's local BN; let R_A be the root node or service concept of agent A's local BN. Agents A and B can integrate their local networks if the service concept R_A of A is identical to an input concept of B:

$$\{R_A\} \cap \mathcal{L}_B \neq \emptyset \tag{8}$$

If this condition holds, then the estimated inferred distribution over the service variable R_A performed by A can be used as input for agent B.

An example of a DPN created from a quasi-static monolithic BN is given in Figure 3; the nodes E_i^C and E_i^D are dynamic nodes, that is, they are instantiated and added during time. The DPN is organized in four agents, two of them are sensor agents (those which deal with the dynamic nodes) and the other two are fusion agents. Thanks to the distribution of the control, the sensor agents will be the only ones that will modify their local BN structure. Further information is thoroughly provided in [5].

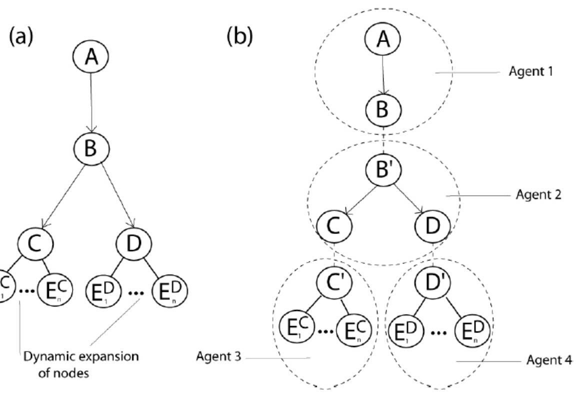

Figure 3: (a) Quasi-static monolithic Bayesian network with instantiated dynamic nodes E_i^C and E_i^D. (b) Correspondent DPN fusion system. Image taken from [5].

4 Background: Reinforcement Learning

Reinforcement Learning (RL) addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals [9]. This method has its roots in the study of animal trial-and-error behaviour under the influence of external stimuli [19]. RL provides a successful approach to sequential decision making problems.

The task of the agent is to perform a sequence of actions that would lead it to a goal state. In other words, the task is to learn a control policy

$$\pi : \mathcal{S} \rightarrow \mathcal{A} \tag{9}$$

Where $\mathcal{S}$ is the set of states of the environment and $\mathcal{A}$ the set of actions available to the agent. In order to learn a control policy, the agent receives from the environment a reward, that is quantitative information about the action just performed. Some other information about the current state of the environment, furthermore, can be received, as it will be explained later in this chapter. However, intuitively, the policy learnt should be the optimal one, that is, a policy π^* that maximizes the cumulative discounted reward received by performing actions from the initial state to the goal state.

The agent learns by directly interacting with the environment and this interaction is the result of the agent's actions performed in particular states; this means that the agent influences the distribution of states that are visited. Ideally, the agent should explore in an exhaustive way all the states of the environment; on the other hand, the agent should visit more those states that are “better” than others, that is, that let it reach the goal state in an optimal way. In other words, the agent must face the problem of exploitation of known states versus the exploration of the environment.

4.1 Markov Decision Processes

It is now investigated the simplest type of RL scenario, the Markov decision process (MDP), and its elementary solution method, the dynamic programming method's value iteration, is provided.

An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \Delta, \mathcal{R})$ where:

- $\mathcal{S}$ represents the state space of the environment; $s \in \mathcal{S}$ is the single state representation;
- $\mathcal{A}$ represents the finite set of actions of the agent; $a \in \mathcal{A}$ is the representation of a single action;
- Δ is the state transition function, that is, a mapping between state-action pairs to a probability distribution over the state space: $\Delta : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$; $\delta(s, a, s') = P(s'|s, a)$ represents the state transition function related to a single state-action pair;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a real valued function over state-action-state triplets; the function

  $$r(s, a) = \sum_{s' \in \mathcal{S}} \delta(s, a, s')\mathcal{R}(s, a, s') \tag{10}$$

  is the expected immediate reward for a given state-action pair.

In a Markov Decision Process, the functions δ and r depend only on the current state and action, and not on the earlier states and actions [9].

The MDP scenario is the following: at each discretized timestep t the agent, being in current state s_t, chooses an action a_t and performs it. The environment responds at timestep $t + 1$ by producing the succeeding state s_{t+1} and by giving the agent a reward $r_{t+1} = r(s_t, a_t)$. A representation of this scenario is given in Figure 4.

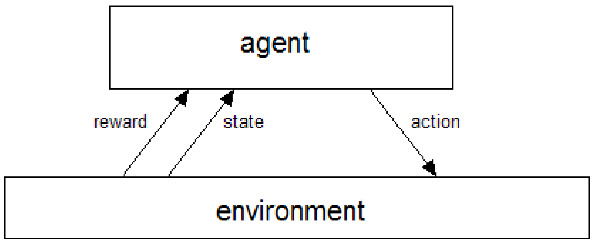

Figure 4: MDP scenario.

Almost all reinforcement learning algorithms are based on estimating value functions (also called functions of states or state values) that estimate how good it is for the agent to be in a given state [11]. It represents the expected discounted cumulative value achieved by following an arbitrary policy π from an arbitrary initial state s_t. It is defined as follows [9]:

$$\begin{aligned} V^\pi(s) &= E_\pi\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s\right] = \\ &= E_\pi\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s\right] \end{aligned} \tag{11}$$

where $E[.]$ denotes the expected value given that the agent follows policy π, $\gamma \in [0,1)$ is a discount rate that ensures that even with infinite sequences the sum is finite [19], and t is any timestep. A fundamental property of value functions used throughout reinforcement learning and dynamic programming is that they satisfy particular recursive relationships. For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states:

$$\begin{aligned} V^\pi(s) &= E_\pi\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s\right] = \\ &= E_\pi\left[r_{t+1} + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+i+2} | s_t = s\right] = \\ &= \sum_{s'} \delta(s, \pi(s), s') \left[\mathcal{R}(s, \pi(s), s') + \gamma E_\pi\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i+2} | s_{t+1} = s'\right]\right] = \\ &= \sum_{s'} \delta(s, \pi(s), s') \left[\mathcal{R}(s, \pi(s), s') + \gamma V^\pi(s')\right] \end{aligned} \tag{12}$$

where $s' \in \mathcal{S}$ are the next states. Equation (12) is the Bellman equation for V^π [9].

Value functions define a partial ordering over policies. A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of for all states. In other words, $\pi > \pi'$ if and only if $V(\pi) > V(\pi')$ for all $s \in \mathcal{S}$. There is always at least one policy that is better than or equal to all other policies. This is an optimal policy and it is denoted by π^*. There can exist more than one optimal policy, nonetheless, they all share the same state-value function, called the optimal state-value function, denoted V^*, and defined as:

$$V^* = \max_\pi V^\pi(s) \tag{13}$$

for all $s \in \mathcal{S}$. The policy that selects actions in order to maximise the value function for the current state is called greedy policy and it is an optimal policy [19]. The greedy policy is then defined as:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} V^*(s) \tag{14}$$

for all $s \in \mathcal{S}$. If the agent has an exact knowledge of the model of the environment, that is δ and r, and $\mathcal{S}$ is finite, then dynamic programming method's value iteration can be used to compute (13). Value iteration operates by using the greedy policy π^* in (12) and changing it into an update rule; the algorithm becomes the following:

$$\begin{aligned}\widehat{V}(s) &\leftarrow \max_a E\left[r_{t+1} + \gamma \widehat{V}(s_{t+1}) | s_t = s, a_t = a\right] = \\ &= \max_a \sum_{s'} \delta(s,a,s') \left[\mathcal{R}(s,a,s') + \gamma \widehat{V}(s')\right]\end{aligned} \tag{15}$$

where $\widehat{V}$ represents the current estimates for V^*. For arbitrary initialization of the value functions, it can be shown that $\widehat{V}$ converges to V^* under the same conditions that guarantee the existence of V^*.

4.2 Partially Observable Markov Decision Processes

A Partially Observable Markov Decision Process (POMDP) is a generalization of a Markov Decision Process [10]. A POMDP models a decision process in which it is assumed that the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state. Instead, it must infer a distribution over the state based on a model of the world and some local observations.

A POMDP, hence, is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{Z}, \Delta, \mathcal{R}, \mathcal{O})$ where, in addition to the elements of the original MDP definition, a finite set of observations $\mathcal{Z}$ and the observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{Z})$ are added. The observation function maps the state action pairs to a probability distribution of $\mathcal{Z}$. $o(s,a,z) = P(z|s,a)$ is the individual probability distribution of the observation given state s and action a.

Given the partial observability, the control policy (9) and the value iteration algorithm for policy learning (15) defined in the previous paragraph become infeasible. Since the state is not known, the complete history of observations should be kept and the optimal policy should be based on that, making the problem not Markovian anymore.

A solution is to keep a belief state, that is, a summary statistic of the entire process. Unlike the entire history, the belief state is of fixed dimension; nonetheless, it is a sufficient statistic for the history, which means that optimal behaviour can be achieved using the belief state in place of the history [10]. The belief state b represents a probability distribution over the set of states $\mathcal{S}$ where $b(s)$ represents the probability of being in state s. Furthermore, $\mathcal{B}$ is the probability space over the beliefs b. The task now is to learn a control policy

$$\pi : \mathcal{B} \rightarrow \mathcal{A}$$

The scenario, at this point, is the following: after having performed action a_t at time t, the agent receives, at time $t+1$, an observation z_{t+1}, together with the reward $r_{t+1} = r(s_t, a_t)$. The representation of this scenario is given in Figure 5.

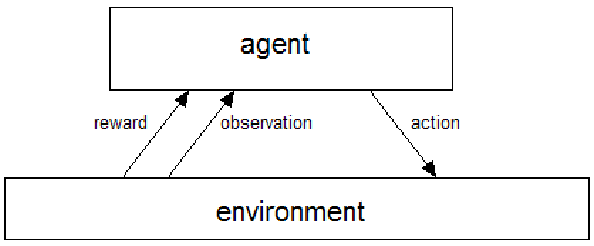

Figure 5: POMDP scenario.

At timestep $t+1$, the agent must update its belief state b_t; it turns out that Bayes rule, defined in (4) and (6), can successfully compute

$$b_z^a(s') = P(s'|b,a,z)$$

where s' are the next states; the notions a_t, z_t and b_t are represented without the underscript t for a better interpretation of the formula just provided.

Through the use of the belief state, the POMDP can be treated as a continuous space MDP [10], where the continuous state space is represented by $\mathcal{B}$ and the action set still remains $\mathcal{A}$. Since $\mathcal{A}$ and $\mathcal{Z}$ are finite, the number of successor belief states are finite.

Before approaching the new value function and its related value iteration algorithm, new components must be defined. The state transition function, now, defines the probability of a particular successor belief state, given an initial belief state and action. Since each observation can yield a different succeeding belief state, the belief state transition is defined as following:

$$\psi(b,a,b') = \sum_{z \in \mathcal{Z}} P(z|b,a) I(b', b_z^a) \tag{16}$$

where

$$I(x,y) = \begin{cases} 1 & if x = y \\ 0 & else \end{cases} \tag{17}$$

that is, the sum of the probabilities of all the observations that would lead to this belief state [10]. The set of possible successor belief states is represented by

$$\mathcal{B}'(b,a) = \{b_z^a\} \subset \mathcal{B}$$

and the reward function is now defined as

$$\omega(b,a) = \sum_{s \in \mathcal{S}} b(s) r(s,a) \tag{18}$$

which simply uses the belief state in an expectation over all states [10]. By replacing s with b, $\delta(s,a)$ with $\psi(b,a,b')$ and $r(s,a)$ with $\omega(b,a)$ in (12), the value function equation for POMDP is obtained:

$$V^\pi(b) = \omega(b, \pi(b)) + \gamma \sum_{b' \in \mathcal{B}'(b,\pi(b))} \psi(b, \pi(b), b') V^\pi(b') \tag{19}$$

and its related value iteration update is defined as follows [10]:

$$\widehat{V}(b) \leftarrow \max_a \left[\omega(b,a) + \gamma \sum_{b' \in \mathcal{B}'(b,a)} \psi(b,a,b') \widehat{V}(b') \right] \tag{20}$$

4.3 TD-Methods

The value iteration approaches defined in Sections 4.1 and 4.2 are based on the assumption that the agent has an exact knowledge of the model of the environment. When the state transition is unknown, value iteration cannot be applied; moreover, even if it can be applied to continuous state spaces, for example in a POMDP, the exact solution is generated only in special cases [21]. Temporal difference (TD) methods are a class of algorithms that can learn directly from raw experience without a model of the environment's dynamics [11] by reducing discrepancies between estimates made by the agent at different times [9]. The formulas provided in this paragraph are based on MDP problems.

Before introducing Q-learning, a TD method for control policy learning, the Q-function is introduced. This function can be used, unlike V, when the environment's dynamics, that is the state transition function and the reward function, are not known to the agent.

Similarly to (11), the q-value of taking action a in state s under a policy π, denoted $Q^\pi(s,a)$, is the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q^\pi(s,a) = E_\pi\left[\sum_{i=0}^{\infty}\gamma^i r_{t+i+1}|s_t = s, a_t = a\right] \tag{21}$$

hence the optimal action-value function denoted by Q^* is defined as

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{22}$$

for all $s \in \mathcal{S}, a \in \mathcal{A}$. For the state-action pair (s,a), this function gives the expected return for taking action a in state s and thereafter following an optimal policy. The greedy policy, now, is the policy that selects the action that maximises the q-value for the current state:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{23}$$

By skipping the intermediate steps, which can be found in [11], the Bellman optimality equation for Q^* is:

$$\begin{aligned} Q^*(s,a) &= E\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a')|s_t = s, a_t = a\right] = \\ &= \textstyle\sum_{s'} \delta(s,a,s')\left[\mathcal{R}(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \end{aligned} \tag{24}$$

Q-learning is the rule that updates the current q-values as follows:

$$\widehat{Q}(s_t,a_t) \leftarrow \widehat{Q}(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a \widehat{Q}(s_{t+1},a) - \widehat{Q}(s_t,a_t)\right] \tag{25}$$

where α is a constant step-size parameter and the notation $\widehat{Q}$ indicates the estimates of the optimal Q-function Q^*.

If all states are visited infinitely often and α decreases slowly with time, Q-learning can be shown to converge to the optimal $Q^*(s,a)$ [19]. This brings back to the issue of exploration vs exploitation described at the beginning of this chapter. The approach used in this thesis is the so called ϵ-greedy exploration: the agent performs a random action with probability ϵ, where ϵ is a small number, in order to explore the state space of the environment; with probability $1-\epsilon$, instead, the agent will perform action $a = \arg\max_{a'} \widehat{Q}(s,a')$, in order to exploit the states it has already visited. In case of ties, one of the actions that leads to the highest q-value is selected randomly.

4.4 Function Approximation

Q-learning defined in (25) can be successfully used when both the state space and action space are finite. If so, then, the q-values can be stored in a table, where at each state-action entry, the correspondent q-value is given. In this way, all the current estimates are kept and updated. Unfortunately, the representation of state-action pairs in table entries is limited to tasks with small tables. The problem is not just the memory needed for large tables, but the time and data needed to fill them accurately. In other words, the key issue is that of generalization. The question is: how can experience with a limited subset of the state space be usefully generalized to produce a good approximation over a much larger subset?

Generalization from examples has already been extensively and successfully studied, so it is possible to combine reinforcement learning methods with existing generalization methods. The kind of generalization required is often called function approximation because it takes examples from a desired function (e.g., a value function) and attempts to generalize from them to construct an approximation of the entire function [21].

Analogously to what is presented in other sections, the attention is first given to the study of the value function V. The approximate value function at time t, $\widehat{V}(s_t)$, is no longer represented as a table but as a parameterized function with parameter vector $\overrightarrow{\theta}$. This means that the value function $\widehat{V}(s_t)$ depends totally on $\overrightarrow{\theta}$, varying from timestep to timestep only as $\overrightarrow{\theta}$ varies [21].

Typically, the number of parameters (the number of components of $\overrightarrow{\theta}$) is much less than the number of states in $\mathcal{S}$, and changing one parameter of $\overrightarrow{\theta}$ changes the estimated value of many states. Consequently, when a single state is used to update the value function, the change generalizes from that state to affect the values of many other states [21].

Due to generalization, however, it might happen that the value function for some states is no longer optimal. Some error between the approximated value function $\widehat{V}$ and the true value function V^{π} is introduced. What is aimed, at this point, is the minimization of such error. Since the distribution of states visited P influences the learning process, then, the idea is to find a parameter vector $\overrightarrow{\theta}$ that minimises the squared error

$$MSE(\overrightarrow{\theta}) = \sum_{s \in \mathcal{S}} P(s) \left(V^{\pi}(s) - \widehat{V}(s) \right)^2 \tag{26}$$

Because of that, instead of a global optimum, a local optimum parameter $\overrightarrow{\theta}^*$, which $MSE(\overrightarrow{\theta}^*) \leq MSE(\overrightarrow{\theta})$ for all $\overrightarrow{\theta}$ in some neighborhood of $\overrightarrow{\theta}^*$ is sought [21].

4.4.1 Gradient Descent Method

Gradient descent is one of the most widely used function approximation techniques in reinforcement learning. The parameter vector is a column vector with a fixed number of real valued components

$$\overrightarrow{\theta} = \begin{pmatrix} \theta(1) \\ \theta(2) \\ \vdots \\ \theta(n) \end{pmatrix}$$

and $\widehat{V}(s_t)$ is a smoothed differentiable function of $\overrightarrow{\theta}$ for all $s \in \mathcal{S}$ [21].

The value function $\widehat{V}$ that is estimated during learning, intuitively, should be the closest possible to the real one, V^{π} for all states; furthermore, it is assumed that the states are visited according to the probability distribution P. The task is the minimisation of (26). A good strategy in this case is to try to minimise error on the observed states. Gradient-descent methods do this by adjusting the parameter vector after each example by a small amount in the direction that would most reduce the error on that example:

$$\begin{aligned} \overrightarrow{\theta} &\leftarrow \overrightarrow{\theta} - \tfrac{1}{2}\alpha \nabla_{\overrightarrow{\theta}} (V^{\pi}(s_t) - \widehat{V}(s_t))^2 = \\ &= \overrightarrow{\theta} + \alpha (V^{\pi}(s_t) - \widehat{V}(s_t)) \nabla_{\overrightarrow{\theta}} \widehat{V}(s_t) \end{aligned} \tag{27}$$

where α is a positive step-size parameter, and $\nabla_{\overrightarrow{\theta}} f(\overrightarrow{\theta})$, for any function f, denotes the vector of partial derivatives

$$\nabla_{\overrightarrow{\theta}} f(\overrightarrow{\theta}) = \begin{pmatrix} \frac{\partial f(\overrightarrow{\theta})}{\partial \theta(1)} \\ \\ \frac{\partial f(\overrightarrow{\theta})}{\partial \theta(2)} \\ \vdots \\ \frac{\partial f(\overrightarrow{\theta})}{\partial \theta(n)} \end{pmatrix}$$

This derivative vector is the gradient of f with respect to $\overrightarrow{\theta}$. This kind of method is called gradient descent because the overall step in $\overrightarrow{\theta}$ is proportional to the negative gradient of the example's squared error. This is the direction in which the error falls most rapidly [21].

Usually, the true value function V^{π} is unknown, therefore, the update formula (27) cannot be used anymore. However, it is possible to approximate it by substituting v_t in place of $V^{\pi}(s_t)$. This yields the general gradient-descent method for state-value prediction:

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \alpha(v_t - \widehat{V}(s_t))\nabla_{\overrightarrow{\theta}} \widehat{V}(s_t) \tag{28}$$

If v_t is an unbiased estimate, that is, if $E[v_t] = V^{\pi}(s_t)$, for each t, then $\overrightarrow{\theta_t}$ is guaranteed to converge to a local optimum under the usual stochastic approximation conditions for decreasing the step-size parameter α [21].

4.4.2 Control with Function Approximation

It is now sought a method that aims to learn to approximate, instead of the value function V^{π}, the q-value function Q^{π}, for the same reasons that were illustrated in Section 4.3. The Q-function, clearly, is represented as a functional form of state-action pairs with parameter vector $\overrightarrow{\theta}$. From such function, then, a control policy can be defined.

The parameter vector $\overrightarrow{\theta}$ is then built in a similar way of equation (28):

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \alpha(v_t - \widehat{Q}(s_t, a_t))\nabla_{\overrightarrow{\theta}} \widehat{Q}(s_t, a_t) \tag{29}$$

where

$$v_t = r_{t+1} + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') \tag{30}$$

where $\widehat{Q}$ denotes the current estimate of the action-value function. By using this method, then, the agent can learn an approximate function for state action pairs that minimizes the true action-value function. Furthermore, this method is model-free, in accordance with what said in Section 4.3.

5 Method: DPNs for Chemical Leak Detection

This chapter analyses the chemical leak detection problem by defining causal relationships between leak and sensor readings; from this, it defines the DPN model that will be used in the rest of the thesis.

It is important that, when a chemical leak occurs in some factory, depot, etc. it is detected quickly, in order to avoid massive contamination of the environment. Several sensors can be placed next to factories in industrialized areas, and through their readings, some reasoning about the chemical leak can be done.

A chemical leak generates toxic gas which typically spreads in the environment; the consequence of diffusion and air flow makes this event perceived even far away from the leak location. The presence of gas generates effects that are perceived by sensors. For example, they can be specialized into the detection of different phenomena, such as variation of the ionization or condensation of the air, etc. The sensors perform physical readings which are then compared to a set of thresholds which represent the sensor calibration. In this thesis's simple case, one threshold and two discrete intervals are assumed. Two boolean values can represent the intervals: $true$ if the physical reading has a value equal or greater than the threshold, $false$ otherwise.

The area where a chemical leak might occur is usually large; for example industrialised areas or harbours are usually composed of many factories. If the area is partitioned into m clusters, then, in order to have an optimal monitoring, each type of sensors should be placed in each cluster.

In real life, the sensors are faulty: the result of the inference, hence, can lead to false positives (the leak is detected in a location where it is not occurring) or false negatives (the leak is not detected where it is occurring). This is due to the fact that the phenomenon the sensors detect can be generated by other causes, such as natural events or they can be the result of some non dangerous chemical transformation; another reason could be the fact that the concentration of gas spread is very high because the leak is in a neighboring cluster. In order to have a robust monitoring system, many readings should be performed and processed all together.

A causal model for the chemical leak is shown in Figure 6; it is based on m clusters and n types of sensors; each type of sensor is present in each cluster. The root node represents the leak location; its domain is composed of the possible m locations where it can occur. According to a conditional probability distribution, which describes the physical laws of chemical contamination in the environment, Gas in different clusters can be present. For each cluster, n types of sensors are modeled; the leaf nodes represent the $k + 1$ instantiated sensor readings; new instantiated leaves are added each time new readings occur. The temporal property of the readings makes the network a quasi-static one.

The goal is to perform inference about leak position given sensor readings. Clearly, the load of work done when using the monolithic BN is huge, especially if n, m and k grow. Distributed inference is advised. In order to define the DPNs, a top down view of the causal process, that is, from the root to the leaf nodes, is adopted.

If a leak occurs, then, the node *leak position* of the BN is instantiated; according to property (D2) defined in Section 2.3, the node d-separates the Gas-es; the result is a partition into m sub-networks, as shown in Figure 7.

Through this separation, a distribution into clusters is achieved: all the sensor readings for all the sensors of each cluster can be considered in the inference about the Gas presence in that cluster, independently on the result of the other cluster models.

It has been said that when the leak occurs, a toxic gas spreads in the environment. This corresponds to the instantiation of the m Gas nodes in the BN. In the same cluster, then, Gas d-separates

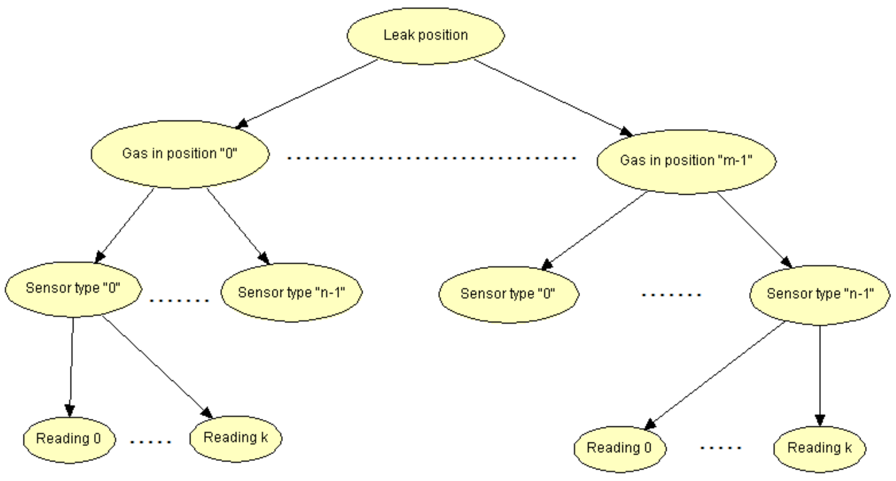

Figure 6: Quasi-static causal Bayesian network for the leak detection problem.

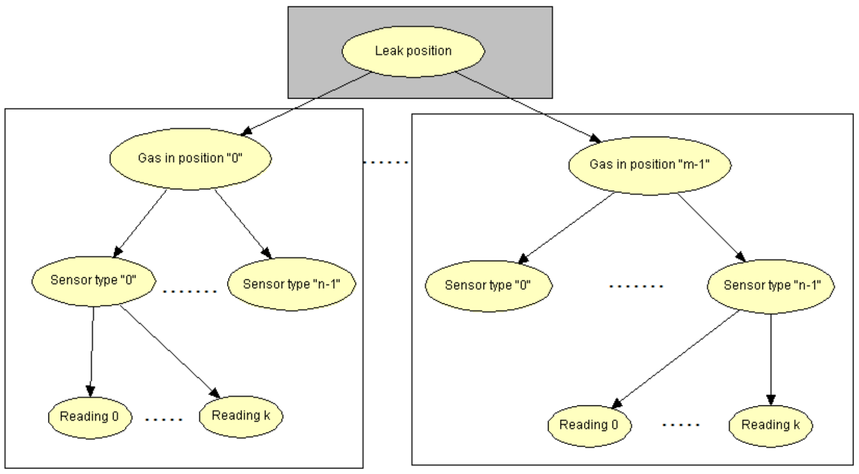

Figure 7: The *leak position* root node d-separates the *Gas* nodes.

the n sensor models, still according to property (D2). This is shown in Figure 8.

All these d-separated parts of the true monolithic network can be represented by independent inference modules using partial BNs; these models are organized into DPNs, one for each m possible cluster. Each DPN is composed of n sensor agents, and each agent deals with its own readings. The agent organization is given in Figure 9.

This distribution of the inference is such that only the sensor agents will deal with a change of their network topology. The inference of *Gas* given sensor readings is performed according to what said is in Sections 2 and 3.

The global inference of *leak position* given *Gas* is not present in any of the DPNs; as it will be shown in the next chapter, this inference will be performed by the helicopter to update its leak belief.

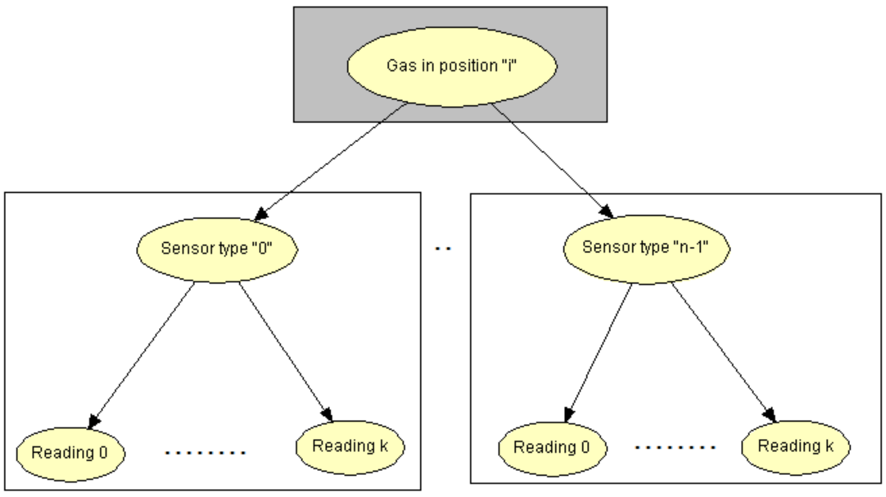

Figure 8: The *Gas in position "i"* node d-separates the n possible sensor type nodes.

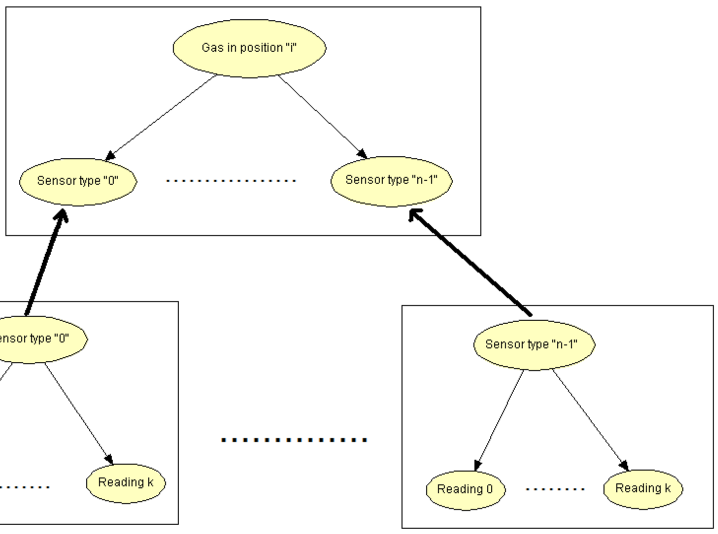

Figure 9: DPN's agent organization.

6 Method: Integrating RL with DPNs

This chapter investigates methods that aim to obtain mobile sensor control though the integration of RL techniques described in Chapter 4 and DPN systems presented in Chapter 3. The case scenario is the chemical leak detection problem introduced in Chapter 5: several DPN systems are present in the environment and, according to sensor readings, perform inference about *Gas* presence; the integration between the two systems is represented by the RL interpretation of the DPN inferences as observations that are sent to the helicopter agent, which performs a higher inference about leak location.

Before going any further, an ambiguity should be clarified: all the RL techniques described in Chapter 4 consider the single agent case; on the other side, it has been said in Chapter 3 that a DPN system is a multi-agent approach for inference. Under the RL perspective, the ambiguity is solved by the fact that the observation z the helicopter receives is the result of the causal inference performed by the fusion system, that is, the most abstract level of information possible. This means that the helicopter does not have to communicate with all the other agents, hence it can ignore their existence. On the other perspective, the fusion system treats the helicopter agent as a sensor agent, that is, it considers it as part of its multi-agent system. Thanks to the fact that the DPN system does not perform any control on the sensors, and the fact that is robust with respect to the agent network composition, it can treat the mobile sensor as a sensor that "sometimes" it fuses into the system.

It is now provided the formal description of the problem. The point of view chosen is the helicopter's one, hence, the DPN systems will be described as a component of the environment. The setting adopted is episodic: the helicopter agent starts an episode at timestep 0 and performs actions until it reaches a goal state. Then, the episode ends and a new one starts. In these episodes, the agent learns the control policy. The following assumptions are defined and hold for each separate episode:

(A1) Only one leak is occurring and in a random position of the environment;

(A2) The leak is never changing its position;

(A3) The leak is not changing its contamination area;

(A4) There is one DPN for each of the $n \cdot m$ cells of the environment.

The meaning of these assumptions will become clear through the rest of the chapter.

Environment Organization

The environment is represented as two dimensional grid of squared cells (also called clusters) divided into n rows and m columns. Each cell is univocally identified by its (y, x) coordinates according to the y and x cartesian axis of the grid. The coordinates assume positive integer values, with the origin situated on the top left corner of the grid. The top-down point of view of the helicopter is taken. The environment has boundaries, corresponding to the four sides of the grid: the helicopter cannot leave the environment (this is also called border condition). The coordinate system of the grid is used to determine the coordinates of the leak, the helicopter and the DPNs. The helicopter knows its own position[3] but does not know where the leak is. The task of the helicopter is to determine the coordinates of the leak. At timestep 0 of each episode, the helicopter is placed randomly at one of the four corners of the grid[4]. An example of the episode at timestep 0 is given in Figure 10.

[3]It can be assumed that the helicopter through GPS retrieves its World's coordinates and maps them in the environment.

[4]This is done in order to simulate the fact that the helicopter can be kept in a hangar far away from the monitored area: during the time the helicopter takes to get there, an initial contamination occurs.

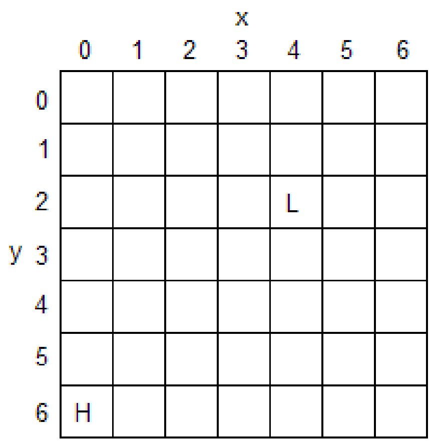

Figure 10: An example of a 7×7 celled environment at timestep 0: the leak L is randomly spawn in position $(2,4)$, the helicopter H is randomly placed at one of the four corners, $(6,0)$.

State Space

As said before, the helicopter does not know anything about the fixed sensors and the DPN systems. Due to that, and assumption $(A4)$, the full state space $\mathcal{FS}$ is defined as followings:

$$\mathcal{FS} = \{[(y_h, x_h), (y_l, x_l)], \forall y_h, y_l = 0, \dots n-1, x_h, x_l = 0, \dots m-1\} \tag{31}$$

Where (y_h, x_h) represents the helicopter position and (y_l, x_l) represents the leak position. $fs \in \mathcal{FS}$ represents a single full state. Alternative ways of representations are also $H_{y,x}$ and $L_{y,x}$, hence $fs = (H_{y,x}, L_{y',x'})$ or simply $fs = (H, L)$; they will be interchanged during the thesis in order to avoid confusions in formulas.

The full state space is composed of all the possible combinations of helicopter position and leak position. However, given (A1), in each episode, the full state space is reduced to the true state space

$$\mathcal{S} = \{[(y_h, x_h), (y_l, x_l)], \forall y_h = 0, \dots n-1, x_h = 0, \dots m-1\} \subset \mathcal{FS} \tag{32}$$

with (y_l, x_l) fixed at the beginning of the episode. In such way, $n \cdot m$ partitions of $\mathcal{FS}$ are defined. $s \in \mathcal{S}$ represents a single true state.

Action Space

The helicopter can perform three kind of actions: it can move, it can activate its sensor (it will be said that the helicopter senses or performs sensing action) and it report a leak location (alternatively said that it performs report action, or simply reports). The state space, therefore, is the following:

$$\mathcal{A} = \{n, s, e, w, l, r\} \tag{33}$$

where $a \in \mathcal{A}$ represents a single generic action of the action space. l represents sensing action and r represents report action. The moving actions are four and are relative to the direction the helicopter takes according to the environment axis: the helicopter can move north, south, east and west, and are represented by n, s, e, w. Only these four actions perform a state transition: they can change the helicopter's position, according to the border condition. The sensing action does not generate any state transition; as it will be shown later on, it implies the fusion of the helicopter sensor model to the DPN located in the same cell of the helicopter; this has as result a sharper inference about the gas presence in that cell. The report action, finally, is the only one that let the helicopter reach a goal state and terminate the episode. When the helicopter performs a report action, it will also declare a leak position (Ld_y, Ld_x) based on its belief.

State Transition Function

The state transition δ is assumed to be deterministic and is the following:

$$\begin{aligned}
\delta\left(\left[H_{y,x}, L_{y',x'}\right], n\right) &= \begin{cases} \left[H_{y-1,x}, L_{y',x'}\right] & if \quad H_y > 0 \\ \left[H_{y,x}, L_{y',x'}\right] & if \quad H_y = 0 \end{cases} \\
\delta\left(\left[H_{y,x}, L_{y',x'}\right], s\right) &= \begin{cases} \left[H_{y+1,x}, L_{y',x'}\right] & if \quad H_y < n-1 \\ \left[H_{y,x}, L_{y',x'}\right] & if \quad H_y = n-1 \end{cases} \\
\delta\left(\left[H_{y,x}, L_{y',x'}\right], e\right) &= \begin{cases} \left[H_{y,x+1}, L_{y',x'}\right] & if \quad H_x < m-1 \\ \left[H_{y,x}, L_{y',x'}\right] & if \quad H_x = m-1 \end{cases} \\
\delta\left(\left[H_{y,x}, L_{y',x'}\right], w\right) &= \begin{cases} \left[H_{y,x-1}, L_{y',x'}\right] & if \quad H_x > 0 \\ \left[H_{y,x}, L_{y',x'}\right] & if \quad H_x = 0 \end{cases} \\
\delta\left(\left[H_{y,x}, L_{y',x'}\right], l\right) &= \left[H_{y,x}, L_{y',x'}\right] \\
\delta\left(\left[H_{y,x}, L_{y',x'}\right], r\right) &= \text{episode ends.}
\end{aligned} \tag{34}$$

Clearly, the boundary conditions are defined.

Reward Function

The reward function must be defined such that the helicopter will report and terminate the episode the fastest possible but, at the same time, the location declared must be correct. It is defined as follows:

$$\begin{aligned}
r(s, \{n, e, s, w, l\}) &= -1 \\
r(s, r) &= \begin{cases} 0 & if \quad Ld = L \\ -1000 & else \end{cases}
\end{aligned} \tag{35}$$

where $Ld = L$ means that the two coordinates are the same along both axis: $Ld_y = L_y, Ld_x = L_x$. With such definition the helicopter must learn when it is better to move or to perform sensing action, since they return the same reward; it also has to learn how to determine Ld in order to have the highest reward.

Belief State

The belief state, as said in Section 4.2, represents the probability distribution over the state space. Due to the assumption (A2), during an episode, the sequence of true states visited will be

$$s_0 = (H_0, L), s_1 = (H_1, L), \ldots s_t = (H_t, L), \ldots, s_T = (H_T, L)$$

where the underscript represents the timestep and at timestep T the helicopter performs report action. The only difference between true states in an episode is given by the helicopter position, but this is known to itself. This means that the probability distribution of the belief state must deal with the probability distribution about the leak location only. Because of assumptions (A1) and (A2), the true probability distribution P_L of the leak position during an episode where the leak is in occurring in $(l_{y'}, l_{x'})$ is the following:

$$P_L(l_y, l_x | l_{y',x'}) = \begin{cases} 1 & if \quad l_y = l_{y'}, l_x = l_{x'} \\ 0 & else \end{cases} \tag{36}$$

for all $y = 0, \dots n-1, x = 0, \dots m-1$. The helicopter, therefore, keeps a belief as a probability distribution with the same dimension as the number of cells of the environment:

$$b = [P(L_{0,0}), P(L_{0,1}), \dots P(L_{1,0}), \dots P(L_{n-1,m-1})] \tag{37}$$

where each possible location is represented through a boolean random variable. b will, form now on, be called the leak belief. The belief state, then, is defined as follows:

$$B = (H_{y,x}, b) \tag{38}$$

During an episode, according to what said in Section 4.2, the belief state will be updated according to a belief transition function. This update generates, at each timestep, the following sequence of belief states:

$$B_0, B_1, \dots, B_T$$

Where the underscript represents the timestep and T. The same sequence can be expressed for the leak belief:

$$b_0, b_1, \dots, b_T$$

The initialization of the leak belief is set to uniform:

$$b = b_0 = \left[P_0(L_{x,y}) = \frac{1}{n \cdot m} \right] \tag{39}$$

for all $y = 0, \dots n-1, x = 0, \dots m-1$. This initialization represents the highest level of uncertainty about the leak position. The task of the helicopter, now, can be seen as the one that aims to reduce the uncertainty about the leak location, that is, from the leak belief (39) it must converge to the true probability distribution (36).

At this point, Ld can be formally defined. It has been said that the helicopter, when performs report action, declares a leak location. Since the task is to reduce the uncertainty about the leak position, the leak position declared Ld should be the location with highest probability according to b:

$$Ld_{y,x} = \arg \max_{(y',x')} (b = b_T) = \arg \max_{(y',x')} P_T(L_{y',x'}) \tag{40}$$

In case of locations with same highest probability, a random one is used in order to break ties.

The observation function and the belief transition function constitute the integration between RL and DPNs; because of that, now, the DPN organization and the sensor reading sampling method are defined.

DPN Organization

The DPN organization follows the intuitions of Chapter 5. Three kinds of sensors are present: two of them are fixed in the environment, the other one is installed in the helicopter. The two fixed sensors are more general purpose and have a lower performance than the mobile sensor equipped in the helicopter; moreover, the two fixed sensors have different performance. The performance of a sensor is the measure of how well it can detect the phenomenon it is specialised on. The two fixed sensors are able to detect condensation and ionization of the air and are called $Cond$ and Ion; Ion has the lowest performance. The mobile sensor is of a new type and is called $MEMS$[5]. By calling $\mathcal{P}$ the performance, the following assumption is done:

[5] $MEMS$ is a technology and not specific sensor. Nonetheless, it can be considered as the technology used to perceive a special phenomenon $\star$. For more information about MEMS technology, check [14].

(A4.1) $\mathcal{P}(Ion) < \mathcal{P}(Cond) < \mathcal{P}(MEMS)$

There is one DPN per cell of the environment; each DPN is composed at least of one fusion agent and the two sensor agents for the two sensors *Cond* and *Ion*. Eventually, if the helicopter performs sensing action, its sensor model will be fused in the DPN of its current cell. The sensor models are defined by means of a quasi-static BN: at each sampling, a new instantiated leaf node is added. The agent organization for the DPN in Figure 11. The CPTs will be defined in Chapter 7, when experimental results are compared.

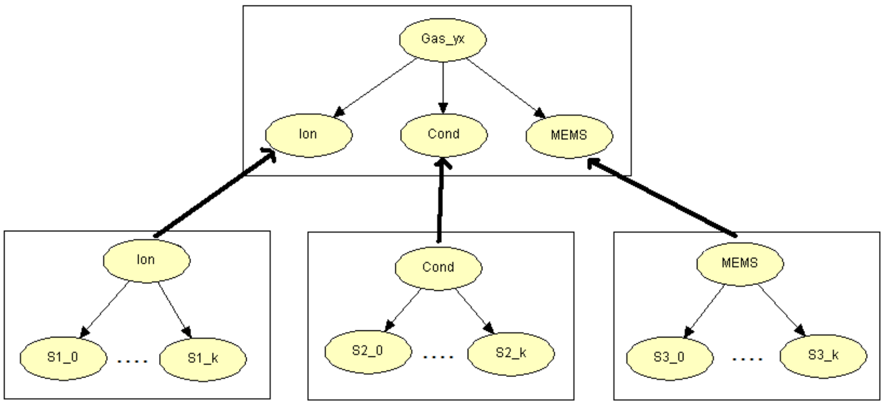

Figure 11: DPN organization. The local BN with Gas_{yx} root is kept by a fusion agent, the other three local BNs are kept by sensor agents.

In each DPN, the inference about *Gas* given observation on the leaf nodes, that is, the sensor readings, is performed. It is assumed that *Ion* generates 5 samples per timestep, *Cond* generates 3 samples per timestep, and *MEMS*, when sensing, generates only one sample.

Mother Nature

Mother Nature is a monolithic BN which is assumed to simulate exactly the physical characteristic of the World and the sensors and it is shown in Figure 12. In accordance with what said for the DPN organization, the CPTs for *Mother Nature* are described in Chapter 7.

According to assumption (A3), only at the beginning of each episode and for each DPN in the environment, the root node *Gas* is instantiated. Then, at each timestep, 5 readings for *Ion* and 3 readings for *Cond* are generated for all the $n \cdot m$ DPNs present in the environment; in case of sensing action, one reading for *MEMS* is generated for the DPN located in the same cell of the helicopter. These readings are then passed to the DPNs which will perform inference.

The way *Gas* is istantiated is now described. Saying that the leak has already contaminated some areas means that the probability of having *Gas* = *true*, for cells that do not correspond to the leak position, is greater than zero. This chance is described by the probability distribution

$$P(Gas_{y,x}|L_{y',x'})$$

for all $y = 0, \dots n-1, x = 0, \dots m-1$ and $L_{y',x'}$ fixed at the beginning of the episode. The distribution should follow the laws of physics of fluid dynamics, gas dynamics and so on. Nonetheless, the intuitive idea is that the closer the DPN in cell (y, x) is to $(l_{y'}, l_{x'})$, the higher the chance of having *Gas*. In other words, $P(Gas_{x,y}|L_{y',x'})$ is inversely proportional to $d((y, x), (y', x'))$, where d is the

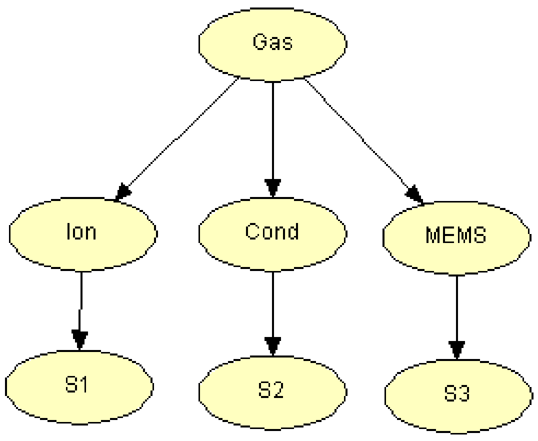

Figure 12: Causal BN model for the sensor reading sampling.

distance defined in some metric space. The distance that is used in this thesis is is the Manhattan distance d_M[6] defined as follows:

$$d_M((y_p, x_p), (y_q, x_q)) = |x_q - x_p| + |y_q - y_p| \tag{41}$$

By fixing the probability of having *Gas* at the leak location at p, the conditional probability distribution is defined as follows:

$$P(Gas_{y,x}|L_{y',x'}) = p \cdot \rho^{d_M(Gas_{y,x}, L_{y',x'})} \tag{42}$$

where $0 \le \rho \le 1$ is a smoothing factor. When $\rho = 0$ and $p = 1$ then the probability distribution is deterministic; when $\rho = 1$ then the contamination is present in the whole environment with same probability p.

Observation Set and Observation Function

The DPNs perform inference about *Gas* given sensor readings. Let $\xi_{y,x}$ be the set of all observations for the DPN in cell (y, x), that is, all the sensor readings from timestep 0 to the current timestep and for all the three sensors; $\xi_{y,x}$ is called hard evidence. The DPN, hence, computes

$$P(Gas_{y,x}|\xi_{y,x})$$

The fusion process in the DPN is done according to Chapter 3, that is, the priors about the root node are kept uniform. The important consequence is the following:

$$P(\xi_{y,x}|Gas_{y,x}) \propto P(Gas_{y,x}|\xi_{y,x}) \tag{43}$$

The beliefs about *Gas* for all cells are called soft evidence and represent the observations that are delivered to the helicopter at each timestep:

$$z = \{P(Gas_{0,0}|\xi_{0,0}), P(Gas_{0,1}|\xi_{0,1}), \dots, P(Gas_{n-1,m-1}|\xi_{n-1,m-1})\} \tag{44}$$

The observation set $\mathcal{Z}$, therefore, is the probability space of all the probability distributions of the soft evidence $P(Gas|\xi)$. The observation function $\mathcal{O}$, intuitively, is defined as the whole sampling/fusion process of *Mother Nature* and the DPNs.

[6]Intuitively, the laws of physics that regulate the chemical contamination are not dependent on such distance; it is used only for simplicity: the pourpose of the distance is to define a probability distribution, and given that the environment is divided in discrete cells, Manhattan distance represents the easiest choice.

Leak Belief Update

Now that $\mathcal{Z}$ and $\mathcal{O}$ are described, it is possible to specify the leak belief update that leads to the belief transition function $\mathcal{B}$. The target is the computation of

$$P(L_{y,x}|\xi) \qquad \forall y = 0, \dots n, x = 0, \dots m$$

where $\xi = \{\xi_{0,0}, \xi_{0,1}, \dots, \xi_{n-1,m-1}\}$ is the set of the set of all the sensor readings for all sensors in all clusters of the environment. By using (3):

$$P(L_{y,x}|\xi) = \frac{P(L,\xi)}{P(\xi)}, \quad P(\xi) > 0 \tag{45}$$

for all $y = 0, \dots n$ and $x = 0, \dots m$. The joint probability distribution $P(L, \xi)$ can be computed through chain rule (1):

$$P(L_{y,x}, \xi) = P(L_{y,x})P(\xi_{0,0}|L_{y,x}), \dots, P(\xi_{n-1,m-1}|L_{y,x}, \xi_{0,0}, \dots \xi_{n-1,m-2}$$

for all $y = 0, \dots n$ and $x = 0, \dots m$. Because of conditional independence, this is reduced to the following:

$$P(L_{y,x}, \xi) = P(L_{y,x}) \prod_{(y',x')} P(\xi_{y',x'}|L_{y,x}) \tag{46}$$

for all $y, y' = 0, \dots n$ and $x, x' = 0, \dots m$. By explicating $P(\xi_{y',x'}|L_{y,x})$ in (46) the result is:

$$P(L_{y,x}, \xi) = P(L_{y,x}) \prod_{(y',x')} \sum_{Gas_{y',x'}} P(Gas_{y',x'}|L_{y,x})P(\xi_{y',x'}|Gas_{y',x'}) \tag{47}$$

and, because of (43), this can be written as follows:

$$P(L_{y,x}, \xi) = P(L_{y,x}) \prod_{y',x'} \left[\sum_{Gas_{y',x'}} P(Gas_{y',x'}|L_{y,x})P(Gas_{y',x'}|\xi_{y',x'})k \right] \tag{48}$$

where $P(Gas_{y',x'}|\xi_{y',x'})$, for all $y' = 0, \dots n$ and $x' = 0, \dots m$, is supplied by the DPN in (y', x') and represents the observation z defined in (44), $P(Gas_{y',x'}|L_{y,x})$ is the probability distribution defined in (42) and k is a constant valid for all cells and states of $Gas_{y',x'}$. It is reasonable to assume that the helicopter knows (42), since it is based on physical laws which are independent on the chemical leak detection problem.

At this point, the leak belief update defined in (45) can be replaced with the following:

$$\begin{aligned} P(L_{y,x}|\xi) &= \tfrac{k^{n \cdot m} P(L_{y,x},\xi)}{k^{n \cdot m} \sum_{y',x'} P(L_{y',x'},\xi)} = \\ &= \tfrac{P(L_{y,x},\xi)}{\sum_{y',x'} P(L_{y',x'},\xi)} \end{aligned} \tag{49}$$

for all $y, y' = 0, \dots n$ and $x, x' = 0, \dots m$.

A final consideration about the integration between the helicopter and the MDPs is done. The helicopter is at the same time the bottom and the top of the global fusion process: it is the bottom when it decides to sense, which makes it fuse with the DPN present in the same location as the helicopter; the helicopter is the top of the global fusion process every time it updates its leak belief through (49), which corresponds to the final global *leak position* root node described in Chapter 5.

Learning Technique

Now that the problem has been defined, a discussion about how the value function should be represented and how it should be calculated must be done. The chemical leak detection task is an atypical POMDP problem, for many reasons that are now illustrated.

POMDP are usually solved through planning, for example, by using value iteration, meaning that the agent has perfect knowledge of the model of the environment. The helicopter has indeed a perfect knowledge of the model of the environment: the leak is never changing its position according to assumption (A2), so the agent can know the state transition function, since it performs only a change of the helicopter's coordinates; the helicopter can also know the reward function, given its characteristic of being constant for all actions apart from the report one; it can also "know" the belief transition function, since it is aware of the existence of the DPNs and it can query them anytime it wants. At first, planning seems to be applicable.

However, planning in POMDPs is intractable for large problems; unfortunately the chemical leak detection is one of them, since the solution sought is supposed to be scalable to large grids. Since planning is not feasible, the only way the helicopter can learn how to find the leak is given by learning through interaction with the environment, in other words, a model-free approach should be used.

Model-free approaches are not usually applicable on the POMDP; when they are, model-free approaches for POMDPs are based on heuristics that aim to give a secondary importance to the uncertainty: for example, [10] describes most likely state (MLS), a heuristic which assumes as current state the one with highest probability in the belief state; it also describes dual control, which is based on two tasks: the first one that focuses its attention on uncertainty reduction, and the second one which is based on performing actions that generate the highest reward. These heuristics are applicable for POMDP problems where the partial observability is only an obstacle the agent faces when tries to solve another task; a typical POMDP problem is robot motion, where the robot's task is to move in an environment and its partial observability is given by the noise introduced via sensor observations, such as images of the environment acquired through cameras.

In the chemical leak detection task, instead, the uncertainty is central to the task that must be solved; this means that heuristic approaches that do not fully condition actions on belief cannot be used: the policy sought must reason about the uncertainty and not just cope with it. However, given that the helicopter has perfect knowledge of the model of the environment, the helicopter can maintain the beliefs and update them; this turns the problem into a continuous space MDP, where the state space is represented by the belief state; this means that mode-free methods are applicable and hence the problem can be solved: the use of belief state-action pairs leads the helicopter to plan conditioned on the variations of the belief state, which changes over time.

As it will be shown in the following section, the belief state representation can be kept continuous or can be discretized; for this last case, a tabular approach for Q-learning is used. If the belief is kept continuous, instead, other techniques should be used; function approximation provides a way for generalizing among states, and this is very useful, since the behaviour the helicopter should have, ideally, is the same in case of similar leak beliefs/level of uncertainty about the leak presence. Gradient descent, a function approximation technique, furthermore, is an incremental step technique, where the parameters of the approximated value function are adjusted of small quantities, as it was shown in Section 4.4.

6.1 Belief State Representations

It is now investigated a set of possible representations for the belief state B defined in (38). Two alternatives are possible: the belief state can be discretized or kept continuous. If it is kept continuous, then, different ways of representing it are possible. In this thesis, four of them are investigated.

In the rest of the section, a squared environment of n^2 cells is assumed in order to have simpler representations of the relationship between environment and belief state size.

6.1.1 Discretized Belief Representation

This representation aims to partition the probability space that defines the leak belief b in order to obtain discrete intervals in accordance with the discrete representation of the helicopter position. The idea is to split the probability interval $[0,1]$ into k partitions of equal size and to assign a reference probability value to each of them; all the probabilities of b will then be represented by the reference probability value of the bin where they fall. Formally, the interval defined by a $bin_i, i = 0 \dots k-1$ is the interval

$$bin_i = \begin{cases} \left[\frac{i}{k}, \frac{i+1}{k}\right) & if \;\; i < k-1 \\ \\ \left[\frac{i}{k}, \frac{i+1}{k}\right] & if \;\; i = k-1 \end{cases} \tag{50}$$

where the k-th bin is closed at the right in order to include the certain probability. The reference probability value can be any of the values within the bin; the value

$$ref_i = \frac{i + \frac{1}{2}}{k}$$

for the i-th bin is considered. All the probability values $P(L_{y,x})$ of b are then represented with the reference value of the bin in which the probability falls:

$$db = \left[ref_{\arg_i(P(L_{0,0}) \in bin_i)}, \dots, ref_{\arg_i(P(L_{n-1,n-1}) \in bin_i)}\right] \tag{51}$$

The discretized belief state, then, is represented by

$$B_d = [(H_y, H_x), db] \tag{52}$$

and, with respect to the grid size, it grows proportional to $O(2 + n^2) = O(n^2)$, that is, linear with respect to the grid size.

6.1.2 Continuous Belief Representations

This paragraph explores different ways to represent the belief state B by keeping the leak belief b composed of continuous probability values.

Representation 1: Vector of Bits and Probabilities

The first representation considers the leak belief as it is defined in (37). Due to this, hence, the belief state space becomes infinite. Some method for generalization over similar states must be found in the value function definition. This approach is shown in Section 6.2.2 and makes use of linear function approximation techniques to learn value functions.

Although the leak belief does not represent a problem anymore to the value function representation, a new issue seems to arise: the representation of the helicopter position with $(H_{y,x})$ seems no longer feasible: the value function is considered linear with respect to the state space, and since the helicopter coordinates are all expressed by positive integer values, it might be possible that the helicopter will not learn the boundary conditions.

An example is the following: at timestep 0 the helicopter is placed in $(0,0)$. Its leak belief is set to uniform, that is, its values greater than zero. At this point, according to the current greedy policy, it performs action e, which leads it to location $(0,1)$; b remains unchanged. It will be shown, in Section 6.2.2, that the function approximation structure aimed to be learnt is such that the gradient is composed of the belief state B only. Because of that, since the helicopter coordinates are $(0, 0)$, the only weights that are updated are those for b. At timestep 1 the greedy policy, since it is still at an early stage, is such that the report action is performed, and the leak declared is the correct one. The gradient of B this time considers also the feature for H_x, but since the reward is 0, no weight update is provided for any of the weights. Now a new episode starts, and the helicopter is spawn in $(1, 1)$. Because of what happened in the previous episode, the action with highest q-value will still be e, making the helicopter hitting the environment boundaries.

Instead of using the coordinates $(H_{y,x})$, the visible part of the true state is now represented by a binary vector $\overrightarrow{H}$ of n^2 cells, one per each cell of the environment, where the entry i is filled as follows:

$$H[i] = \begin{cases} 1 & if \quad i = n \cdot y_h + x_h \\ 0 & else \end{cases} \tag{53}$$

The belief state becomes

$$B_c^I = \left[\overrightarrow{H}, b\right] \tag{54}$$

that is, a vector of two vectors of n^2 elements. This means that the belief state is growing $O(2 \cdot n^2) = O(n^2)$, linearly with respect to the grid size.

Representation 2: Vector of Bits and Entropies

The task of the helicopter is to reduce its uncertainty about the leak location. Its policy should let it go in cells where the uncertainty is high and reduce it by sensing action: when the uncertainty is low, instead, it should not investigate it any further, but focus on other cells of the environment. No reference to the probabilities are mentioned. This suggests the representation of the leak belief through another indicator of uncertainty; the information entropy is used.

In information theory, the Shannon entropy or information entropy is a measure of the uncertainty associated with a random variable. The information entropy of a boolean random variable X is

$$H(X) = -\left[P(x) \log P(x) + P(\overline{x}) \log P(\overline{x})\right] \tag{55}$$

where, of course, $P(\overline{x}) = 1 - P(x)$. Figure 13 shows the relationship between probabilities and entropy for a boolean random variable X.

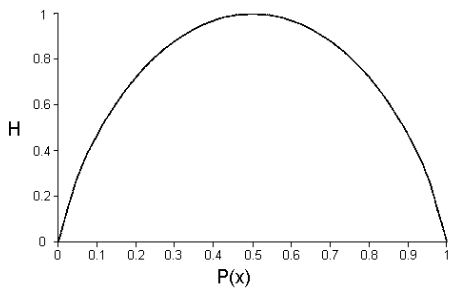

Figure 13: The entropy function relative to a boolean variable X as the value for $P(x)$ varies between 0 and 1.

Since the entropy is a measure which is linear with respect to the uncertainty, the hypothesis now is that, through this representation of the leak belief, linear function approximation might perform better. This hypothesis will be validated in Chapter 7. The entropies are computed for all the n^2 entries of the leak belief; the belief state then becomes

$$B_c^{II} = \left[\overrightarrow{H}, \mathcal{H}(b)\right] \tag{56}$$

where $\overrightarrow{H}$ is the binary vector representing the helicopter position defined in (53) and $\mathcal{H}(b)$ represents the vector of entropies obtained from b.

Such representation does not change the belief state size, which remains linearly dependent to the grid size: $O(n^2)$. However, the leak belief b is not thrown away: it is still needed in order to define the leak position declared when report action is performed, as defined in (40).

Representation 3: Egocentric View

So far, the discretized and the two continuous belief state representations considered the leak belief as a vector of beliefs (either probabilities or entropies) ordered according to their position with respect to the environment origin: the first entry is the belief of cell $(0, 0)$, the second one is the belief of cell $(0, 1)$ and so on, until the last entry, the belief of cell $(n-1, n-1)$.

This third new representation, instead, orders the belief entries according to the helicopter position, that is, represents the belief according to the egocentric view of the helicopter. The hypothesis arguing this representation is the following: with the non egocentric representations, the entry of the leak belief which corresponds to the helicopter's position changes according to the helicopter's movements: the connection between cells of the grid and helicopter position is not represented; with the egocentric representation, instead, the belief/entropy is ordered in order to have the belief/entropy about the current position and its neighboring cells stored always in the same position. In this way the helicopter can know what the uncertainty about that particular far away cell is and decide, according to its policy, whether it is better to move there or maybe perform other actions.

The cells are sorted by increasing Manhattan distance, from distance 0, that is the helicopter's current cell, until the maximum Manhattan distance possible in the environment, that is, $(n-1)+(n-1) = 2n-2$. At a given radius-distance r, all the cells with that distance from the helicopter's position are sorted starting from the cell at the north of the helicopter (that is, in position $(y_h - r, h_x)$) following a clockwise direction. An example of how the cells are sorted is given in Figure 14.

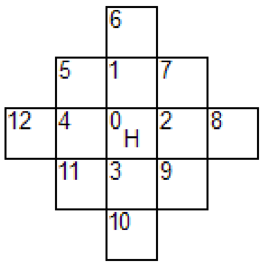

Figure 14: Example of egocentric view ordering of cells when max distance = 2.

The belief state is defined as follows:

$$B_c^{III} = \left[\overrightarrow{H}, \mathcal{H}(b_e)\right] \tag{57}$$

Where $\overrightarrow{H}$ is the binary vector that represents the helicopter position defined in (53), b_e is the leak belief ordered according to the egocentric view, and $\mathcal{H}(b_e)$ is the correspondent vector of entropies ordered according to the egocentric view.

In order to have an extensive coverage of the environment, as said, the maximum distance is $2n-2$ in all four direction; the leak belief then becomes a rhombus of $\frac{(4n-3)^2-1}{2}+1$ entries as shown in Figure 15.

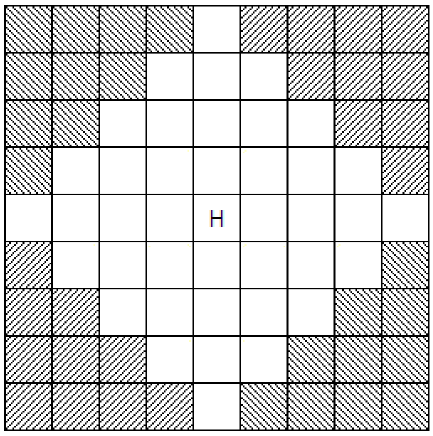

Figure 15: Helicopter's view for a 3×3 grid. The maximum distance is $2n-2=4$; the rhombus can be encompassed by a square with side equal to the diagonal of the rhombus, that is $(2n-2)+(2n-2)+1=4n-3=9$ cells; the square, $(4n-3)^2$, is twice as big as the rhombus; because of the discretised environment, the -1 and $+1$ in the formula are required in order to have a formula which returns a discretised number of cells. The result is 41 cells.

The belief state size is $O(n^2+\frac{(4n-3)^2-1}{2}+1)=O\left(n^2\right)$, still, linear with respect to the grid size.

Representation 4: Reduced Egocentric View

This representation performs aggregations of the two state features defined in (57) in order to shorten the belief state size.

The boundary condition is such that the helicopter cannot leave the environment, as defined in the state transition (34). This means that the helicopter, at the grid boundaries, should not perform some of the four moving actions. According to the available moving actions per each cell of the environment, nine different partitions are defined (sensing and report actions do not affect the helicopter position: for them, the boundary condition does not apply): four of them corresponding to the grid's corners, four of them corresponding to the grid's sides, and finally the central area, where no boundaries are present. An example of this partition is given in Figure 16. The helicopter position feature of the belief state, then, can be represented by a binary vector of eight entries only, with value 1 for the partition of the grid where the helicopter is, and 0 for the others.

The aggregation technique can be done for the leak belief too. Firstly, by analyzing Figure 15, it comes to the attention that many of the cells of the rhombus are always out of the environment, as shown in Figure 17. Given a grid of n^2 cells, A square of $(2n-1)^2$ cells is sufficient to represent all the cells of the environment according to the helicopter's egocentric view.

Now, a second aggregation is defined. The idea is based on the behaviour the helicopter should intuitively have when the environment becomes very big. Let assume the helicopter is at the centre of a large grid at some timestep t. The helicopter is pretty sure about the leak position: it reduced

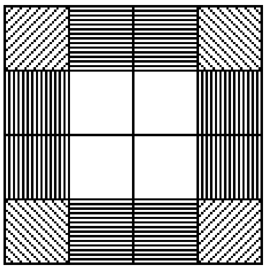

Figure 16: A 4×4 grid partitioned according to the boundary conditions. The cells with same lines belong to the same partition.

its uncertainty to two locations: the first one, called A, located near the top left corner, and the second one, called B, located near the bottom right corner; they also have the same level of entropy. Furthermore, the cells neighboring with A have a higher uncertainty than the neighboring cells of B. The helicopter must decide what to do, but time is running, and cannot go in both areas: it must chose one. If the information about the neighboring cells of A and B could be grouped together with the informations of cells A and B, then, the helicopter would see that general higher uncertainty is present in the area around A and then it should go there to further investigate.

This aggregation is performed by dividing the egocentric view of the helicopter into 9 areas, eight of them representing the 8 far away areas (called suburbs), and 1 representing the central cells (called centre). The areas are defined according to the so called reduced view of the helicopter. A reduced view v defines the centre area as a square of $(2v+1)^2$ cells with center cell the helicopter; all the cells of the centre are represented as separated cells according to the egocentric representation described previously; all the cells outside the centre, instead, will be grouped with the other cells of the same suburb and their mean will be used for the leak representation. An example of how the reduced view is build is given in Figure 18.

The belief state is then defined as follows:

$$B_c^{IV} = \left[\overrightarrow{H_r}, \mathcal{H}(b_{er})\right] \tag{58}$$

where $\overrightarrow{H_r}$ represents the reduced binary vector of 8 partitions of the grid, b_{er} the reduced egocentric leak belief, and $\mathcal{H}(b_{er})$ the vector of entropies of the reduced egocentric leak belief. The state belief, now becomes dependent on the reduced view that defines the size of the centre area: $O\left(8 + (2v+1)^2 + 8\right) = O\left(v^2\right)$. If $v = 0$, that is if the centre area is composed of the helicopter position only, the belief state becomes a vector of 18 features only, making it independent of the grid size: $O(1)$.

6.2 Value Function Representations

The value function aimed to be learned is the one that considers belief state-action pairs, that is, learns q-values. Two techniques are investigated: the first one uses the discretized belief representation defined in Section 6.1.1 and learns q-values through the use of a table as defined in Section 4.3; the second one uses the continuous belief representations defined in 6.1.2 and considers the value function as a linear mapping of belief state-action pairs into values; gradient descent method defined in Section 4.4.2 is used in order to learn the weights of such mapping.

Analogously to what said in Section 6.1, a squared environment of n^2 cells is assumed in the rest of the section and subsections.

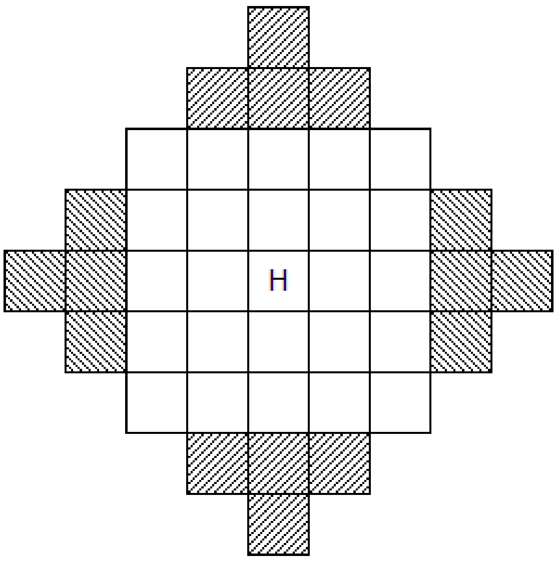

Figure 17: A complete view for the helicopter in a 3×3 grid; the dashed cells are those that will never represent cells of the environment. $(2n-1)^2 = 25$ cells are sufficient to represent all the cells of the environment according to the helicopter's egocentric view.

6.2.1 Tabular Representation

At a glance, the use of the discretized belief state defined in (52) seems to be a successful representation for tabular Q-learning, thanks to the fact that an infinite space is converted into a discrete one. The issue, however, is the number of entries needed in order to represent the state space.

The helicopter can be in one of the possible n^2 cells, and its leak belief can have an infinite combination of probabilities; however, since they are discretized in k intervals, then there will be $O(k^{n \cdot n})$ possible probability combinations. Some of those combinations will never exist, given that the sum of the continuous probabilities of b must sum up to one[7]. The upper bound of number of entries, for a n^2 sized grid, and k probability intervals is $n^2 \cdot k^{n \cdot n} \cdot 6 = O(k^{n \cdot n})$, where 6 is the number of possible actions of the state space.

This formula clearly shows that if the environment is too big and/or the number of bins is too high, then the number of entries needed will be enormous. This goes in contrast with the need of having a precise representation of the leak belief: the higher the number of bins, the closer to the continuous representation, the more precise the true state is approximated. Another problem with the table approach is exploration: the helicopter should visit the entries often enough in order to learn the correct q-values; if the entry are too many, then this will not happen.

6.2.2 Linear Function Approximation

When the belief state B is infinite, that is the leak belief keeps real values, the table approach is no longer applicable. Function approximation works instead. In order to keep the problem solution the simplest possible, it has been assumed that the value function is a linear mapping of the belief

[7]This property, not explicitly stated previously in the thesis, is trivially derived from the marginalized Bayes rule defined in (6).

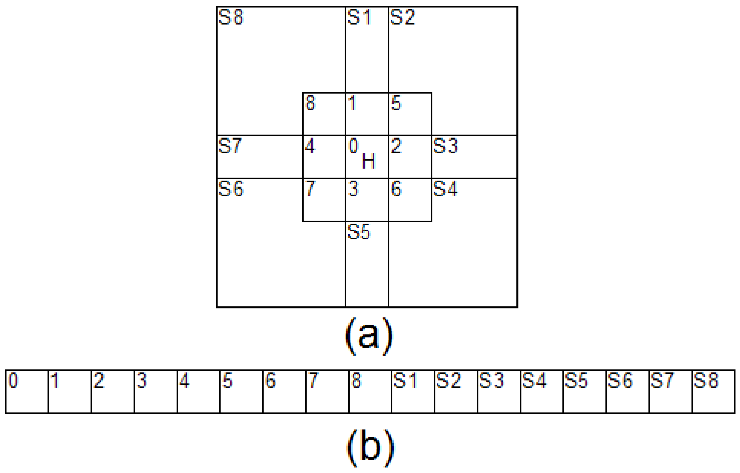

Figure 18: (a) A 7×7 grid divided into 8 suburbs and 1 centre area with $v = 1$ and the helicopter position defined by H. (b) Leak belief representation according to the reduced egocentric view.

space-action pairs.

The first question is: how many vectors $\overrightarrow{\theta}$ should be created? This can be answered by analyzing the action space. Each action has not much to do with the other ones, the effects of their application lead to belief states that are different from the others; not even the movement actions can be grouped together. It has been chosen to use six different vector of weights, one per action; hence, six different functions are defined.

The second question is: how many features should the vectors $\overrightarrow{\theta}$ have? The answer is trivial: as many as the number of features of the belief state B used, where B can be one of those defined in (54), (56), (57) and (58).

To conclude, the formula for the vector of weights $\overrightarrow{\theta}$ update is provided. The gradient descent formula defined in (29) is used. The only component that must be defined, at this point, is the gradient of the Q-function, $\nabla_{\overrightarrow{\theta}}\widehat{Q}(B,a)$. Fortunately, since the value function is linear with respect to belief state-action pairs, and since there is one value function per action, the gradient of the q-value function of belief state-action pairs is reduced to the gradient of the function with belief state state as unique variable. Thus, the gradient is defined as:

$$\Delta_{\overrightarrow{\theta}}\widehat{Q}(B,a) = B \qquad \forall a \in \mathcal{A} \tag{59}$$

The number of weights of $\overrightarrow{\theta}$ are the same as the number of features of B; the total number of features is $O\left(6 + size(B)\right) = O\left(size(B)\right)$; this means that the features can be linear with respect to the grid size in case the belief state B defined in (54), (56) or in (57); it can be a linear function with respect to the reduced view v if B is defined as (58), with the minimum number of features equal to $6 \cdot 18 = 108$ if the reduced view $v = 0$, making the number of features for the leak detection task independent of the grid size.

7 Results

In order to investigate which belief state representation and related value function leads to the best policy, three sets of experiments are done.

In the first one, A, the best belief state and value function representation is investigated; this means comparing various level of discretizations and tabular approach for value function representation described in Sections 6.1.1 and 6.2.1 with the various continuous belief representations and linear value function approximation method described in Sections 6.1.2 and 6.2.2. The results of various discretisations are presented in Section 7.1; then, the best discretised representations are compared with the four possible continuous representations in Section 7.2. The goal of this set of experiments is to perform an initial selection of best representations; these will then be further investigated in the second set of experiments.

The second set of experiments, B, presented in Section 7.3, investigates the robustness of the best methods found in A with respect to the integration with the DPNs; the goal is to determine if the helicopter needs the DPNs and what is the influence that incorrect sensor models have on the policy learnt.

The third set, C, presented in Section 7.4, aims to investigate the robustness of the best methods defined in B with respect to larger grids.

In order to have comparable results, the experiments of the same set have the same parameters. Experiments of A are based on a 2×2 and 3×3 grid size, experiments of B are based on a 5×5 one; experiments of C are based on a 10×10 one. All the experiments are composed of two alternating phases: in the first one, the helicopter uses the ϵ-greedy exploration to learn q-values and explore the belief space; at fixed intervals, the greedy policy learned that far is tested: no exploration is present, and no q-value updates are performed. A special action is introduced for the testing phase: it is an abort action that terminates the episode when the number of timesteps reaches a certain defined limit; this is done because it might happen, especially at early stages of the experiments, that the helicopter does not learn when to report, hence, it never reaches a goal state that terminates the episode. The abort action generates a large negative reward; however, since at testing phase no q-value update is performed, all the rewards are used only for statistical purposes. The reward given when the abort action occurs, for all the experiments, is $-2,000$; the maximum number of actions allowed is $1,000$ for experiments of A and B; it is set to $2,500$ for C. For A and B, $50,000$ learning episodes are performed, with 100 testing episodes performed every $5,000$ learning episodes; the experiments of C, instead, are composed of $100,000$ learning episodes, with 100 testing episodes performed every $10,000$ learning ones. The total number of testing episodes for the three sets, hence, is $1,000$. For all the experiments, $\epsilon = 0.05$, $\gamma = 0.99$ and $\alpha = 0.01$.

10 runs for each experiment were made; the average over these 10 runs was then showed through an averaged window of 100 testing episodes.

Mother Nature, the monolithic BN used to generate the sensor readings, is the same for all experiments and is one shown in Figure 19. Each timestep, for all the DPNs in the environment, 5 samples for Ion and 3 samples for $Cond$ are generated. If the helicopter performs a sensing action, then, one sample for $MEMS$ is generated. The CPTs are defined in Tables 1, 2, 3 and 4. The CPT definitions satisfy assumption (A4.1) of Chapter 6 and the sampling frequency makes the problem even harder.

The experiments of A and C are mainly focused on belief and value function representation. Therefore they adopt perfect sensor modelling, that is, the local BNs of the fusion system have the same CPTs and topology as *Mother Nature* (see again Tables 1, 2, 3, 4); the agent organization for correct fusion models is shown in Figure 11 and is again given in Figure 20. Experiments of B, instead, investigate the performance when imperfect models are used, and so they define specific CPTs and topologies, described in the related section.

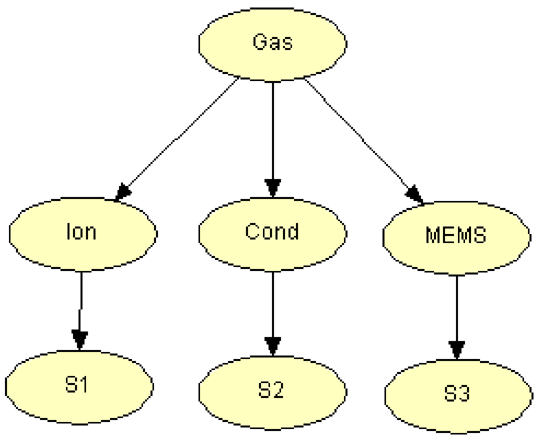

Figure 19: Monolithic BN used to generate sensor readings.

		Gas	
		true	false
Ion,Cond,MEMS	true	0.8	0.4
	false	0.2	0.6

Table 1: CPTs of *Mother Nature* describing $P(Cond|Gas), P(Ion|Gas)$ and $P(MEMS|Gas)$.

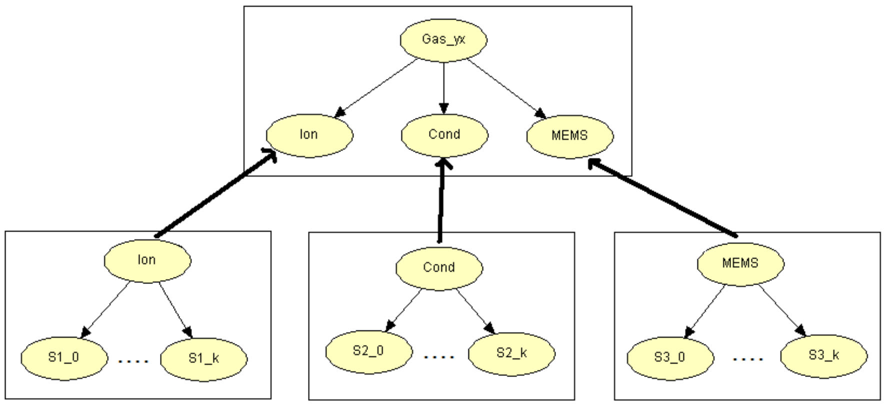

Figure 20: Agent organization for correct sensor models.

The conditional probability $P(Gas|L)$, used to instantiate the *Gas* node of *Mother Nature*, defined in (42), is set with $p = 1$ and $\rho = 0.2$.

7.1 Set A: Comparing Belief Discretizations

This set of tests aims to analyze the effect that the discretized beliefs have on the control policy. Due to the fact that the table size is exponentially proportional to the grid size, in order to have them fit in memory, the experiments could consider only small environments, specifically 2×2 and 3×3. 10 runs of the same experiments were done.

		Ion	
		true	false
S1	true	0.6	0.4
	false	0.4	0.6

Table 2: CPT of *Mother Nature* describing $P(S1|Ion)$.

		Cond	
		true	false
S2	true	0.75	0.25
	false	0.25	0.75

Table 3: CPT of *Mother Nature* describing $P(S2|Cond)$.

2×2 **Grid Size Results**

Five different discretisations were investigated: the leak belief (51) was discretised with 2, 3, 4, 5 and 10 bins. Figure 21 shows the result obtained representig them through an averaged window of 100 episodes.

None of the five different discretisations learned well. It seems clear that the more the bins used to represent the leak belief, the better the performance. This is also seen by analyzing the policy the helicopter has learnt at the end of the experiment: all the representations learn to perform many sensing actions at first; and if they have the luck to be located at the same position as the leak, then they report correctly; it is just a matter of time to reach the threshold according to which the uncertainty about the leak position is low enough, and this is dependent on the level of discretisation. If the helicopter is not spawn at the same location as the leak, instead, most of the time the helicopter keeps performing the same movement action until abort occurs.

Nonetheless, the performance of all the representations is improving slowly; this is due to the large size of the table which must be explored and exploited; Table 5 provides the number of entries needed for each type of discretisation used.

3×3 **Grid Size Results**

It has been tested the performance about different discretisations also for a 3×3 grid. Because of the number of entries of the table, which grew consistently, due to the same need to fit them in the memory, it has been possible to test only two kinds of discretisation: 2 and 3 bins, and their results, averaged with a window of 100 episodes, are shown in Figure 22.

This should be enough to conclude that the discretised representation and the tabular approach are not scalable. This conclusion gets even empowered by comparing the performance of the same discretisations with respect to the grid size: both of them perform worse in the 3×3 case. The 3 bin representation is improving slowly and, apart from the initial performance level, the 2 bin case is slowly improving too.

The policy they both learn at the end of the episode is the same: most of the time, the first two actions are such that the helicopter moves to the centre of the environment; in other rare cases the helicopter keeps hitting the boundaries until abort action occurs. Once the helicopter reaches the centre of the environment, it keeps performing sensing action; if the leak is present at the centre of the environment, then, the helicopter reports it correctly, and as well as what said for the 2×2 case, it is just a matter of time to reach the threshold according to which the uncertainty is low enough.

		MEMS	
		true	false
S3	true	0.9	0.1
	false	0.1	0.9

Table 4: CPT of *Mother Nature* describing $P(S3|MEMS)$

bins	entries
2	384
3	1,944
4	6,144
5	15,000
10	240,000

Table 5: Number of entries per different number of bins for a 2×2 grid.

Unfortunately, if the leak is not located at the centre, the helicopter keeps performing sensing action until abort action occurs.

With a higher level of discretisation, the performance would be surely better, but the number of entries of the table would become too big and this would create, besides the problem to fit in memory, the exploration problem. Table 6 shows the number of entries the different discretisations of the 2×2 case would require for the 3×3 case.

The final conclusion, clearly is that the discretised representation is not scalable.

7.2 Set A: Comparing Value Function Representations

In this section, the two best discretized representations of Section 7.1 are compared with the four continuous belief representations (54), (56), (57) and (58) for the 2×2 and 3×3 case.

2×2 **Grid Size Results**

The first scenario considered is based on a 2×2 grid; the results, averaged over a window of 100 episodes, are shown in Figure 23. 10 bins represents the discretized belief representation, prob is the continuous belief representation (54), entropy is the representation defined in (56), ego_full is the egocentric representation defined in (57); ego_0 is the reduced-egocentric representation defined in (58) with $v = 0$.

What is clearly seen is the fact that the continuous representation prob (54), which is the closest representation to the discretized one (10 bins), is the worst. This is due to the fact that the probabilities are not linear with respect to the level of uncertainty, and the uncertainty is what is meant to

bins	entries
2	27,648
3	1,062,882
4	14,155,776
5	105,468,750
10	54,000,000,000

Table 6: Number of entries per different number of bins for a 3×3 grid.

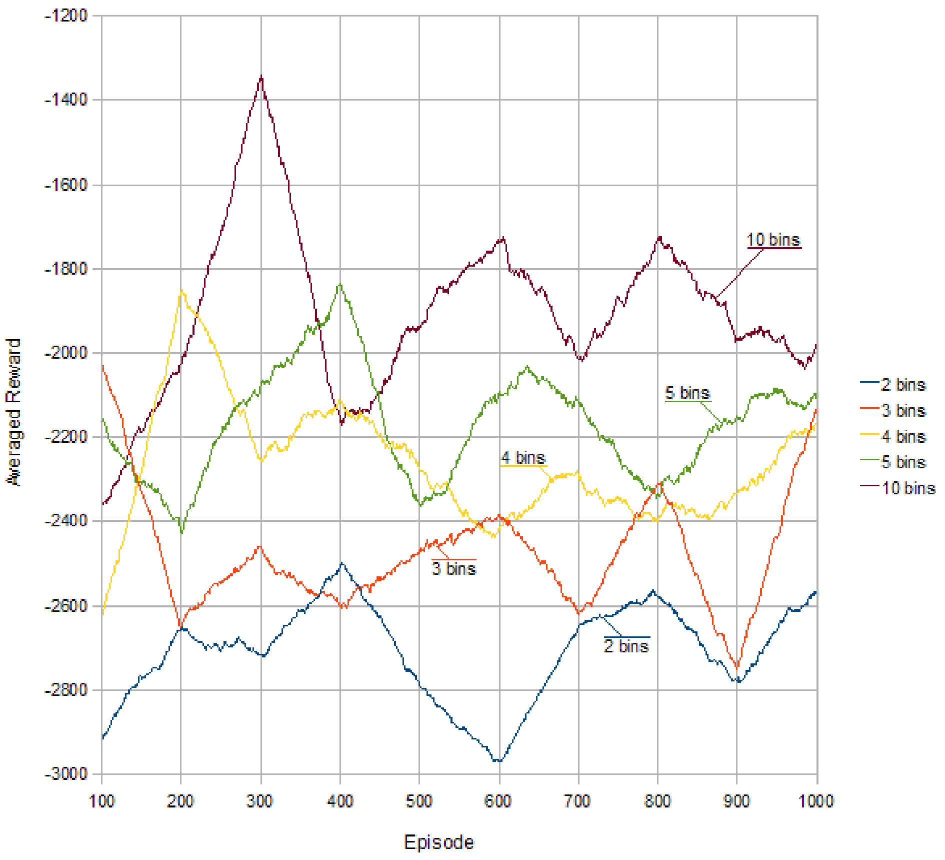

Figure 21: Discretized belief comparisons for a 2 × 2 averaged through a window of 100 episodes.

be reduced by learning a linear value function; this means that the probabilities are not linear with respect to the value function aimed to be learnt.

The belief representation defined in (56), represented in the plot by entropy, seems to be a very good one. However, the behaviour is completely passive: the helicopter does not do anything but just move between cells, many times it performs the same movement action and hits repetitively the boundaries, and then reports the correct leak location. The helicopter learns only when to report. This is due to the fact that the connection between the two components of the belief state is not possible with this representation: every time the helicopter moves the entropy related to the current cell changes, and the weights for that feature should be able to provide a value which should suggest at the same time what to do when the entropy is at the helicopter's position and when it is not. Since the policy learned tells only when to report, this makes the value function completely influenced by the fixed sensors: if in a learning set they perform on average well, then the threshold learnt will be very low, and the helicopter will report soon. If the fixed sensors perform on average bad, instead, the threshold will be very high, and since the policy is passive, either many abort actions or wrong reports will occur.

By using the egocentric representations defined in (57) and (58), instead, the helicopter performs sensing actions as expected. ego_0 performs more constantly than ego_full, and its optimal performance is reached soon. The two representations are more or less the same, they differ only by four features about the leak belief, that is, the features that will never represent cells of the environment. The policy learnt has an initial behaviour similar to what said for entropy, that is, the helicopter learns only when

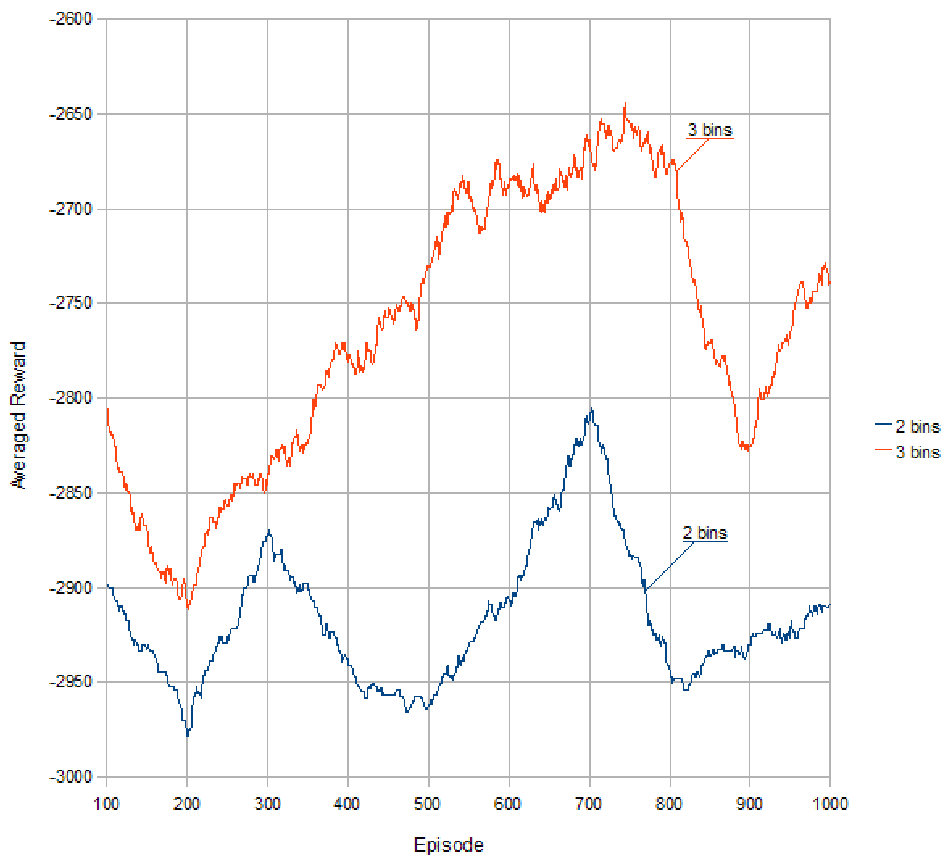

Figure 22: Discretized belief comparisions for a 3×3 environment.

to report; then, in the rest of the experiments, the best policy is learnt. The fact that in both cases the performance goes very low, between -500 and -600, is given by the fact that there are few runs, out of 10, in which the policy is not learnt and many abort actions, together with wrong reports, occur. Considering that the reward for abort action is -3,000 and there are 10 runs, the averaged reward of -500 over 10 runs for example is reached with one abortion, two wrong reports and seven good policies.

The final conclusion is that the egocentric representation seems to be the only one that can learn a good control policy, but the grid is too small to determine which egocentric representation is best.

3×3 Grid Size Results

The results for a 3×3 grid, averaged over a window of 100 episodes, are shown in Figure 24. 3 bins represents the performance of the best discretized belief representation; prob represents the continuous belief representation defined in (54); entropy is the trend of the continuos belief representation with entropies defined in (56); ego_full is the egocentric full representation defined in (57) and, finally, ego_0 and ego_1 are two reduced egocentric belief representation defined in (58) with respectively reduced view 0 and 1.

The intuitions about the 2×2 case can be extended to this case: the discretised belief is unfeasible, especially if compared with the continuous belief representations. prob has a representation of uncertainty which is not linear with respect to the value function aimed to be learnt, entropy still

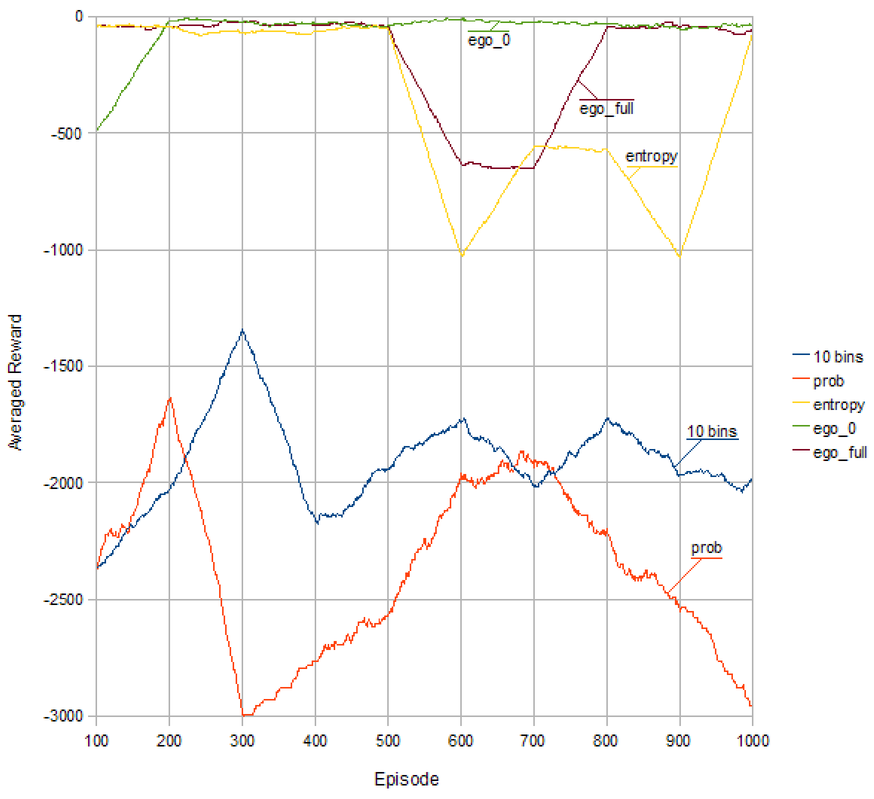

Figure 23: Discretized and continuous belief representation comparisons for a 2×2 grid size.

does not learn to perform sensing action, making its policy wrong.

Further consideration, instead, should be done about the three different egocentric representations. A magnification of their performance is provided in Figure 25.

The hypothesis is that it seems it exists a relationship between number of cells of the environment and the number of features of the leak belief which leads to good policies. ego_full keeps a leak belief of 41 features, ego_1 keeps a leak belief of 17 features, and ego_0 uses only 9 features. This means that for ego_full, apart from 16 features that will never been used, at each timestep, 14 features out of the remaining 25 will not represents cells of the environment; ego_1 will have either 10 or 8 unused features, ego_0 from 4 to 0.

The representation for ego_full is too extensive for a 3×3 grid, and it needs more time to learn the weights for its leak belief features, since most of the time they do not represent cells of the environment; same thing but less strict applies for ego_1; ego_0, instead, since it keeps a very compact representation, seems to have less problems. This is actually reflected in the plot: ego_0 keeps improving its performance which is almost always better than the other two; due to the sparse representation of ego_full and ego_1, instead, it happens that sometimes the policy learnt is very bad, which leads to abort actions in few runs, as well as it was described for the 2×2 case.

The policy learnt for ego_0 is such that the helicopter, at each episode, moves to the centre of the grid in two actions, then performs sensing actions and then reports. ego_1 does the same thing, but the tame it takes to reach the centre is longer: it tends more to switch among neighboring cells;

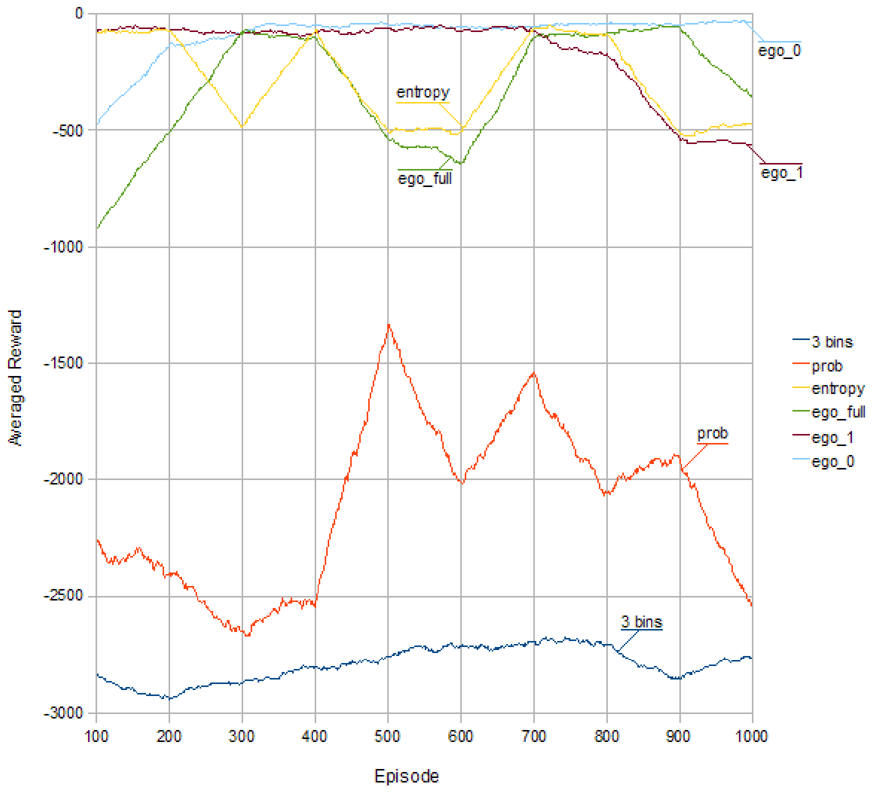

Figure 24: Discretized and continuous belief representation comparisons for a 3×3 sized grid.

sometimes it reaches the centre and, when it happens, it rarely senses; most of the time it reports straight ahead. This is because during the time it switches among cells, the fixed sensors keep working for him, and even if they have a lower performance, they are still better than random guessing; if the result of the inference is such that the uncertainty gets very low, the helicopter will not perform any sensing action and vice versa.

In conclusion, the egocentric representation is the best representation for the leak belief; the reduced view seems to be more flexible than the full representation, and seems to be able to let the helicopter learn a good control policy. A relationship between number of cells of the environment and number of features of the leak belief which leads to good control policy seems to exist.

7.3 Set B: Using Inaccurate Models

This set of experiments aim to determine how much the sensor model structure influences the policy learnt by the helicopter. If a sensor model is inaccurate then its causal representation of gas presence-sensor readings is misleading, that is, the result of its inference is more subject to false positives and negatives. The inaccuracy can affect both the network topology and its CPTs. The set of experiments proposed are based on five possible scenarios: two of them consider a correct network model but with wrong CPTs, the other three consider the case in which the network model is wrong and the CPTs vary according to the three possible CPTs used in the correct causal model scenario. The wrong causal model structure is shown in Figure 26.

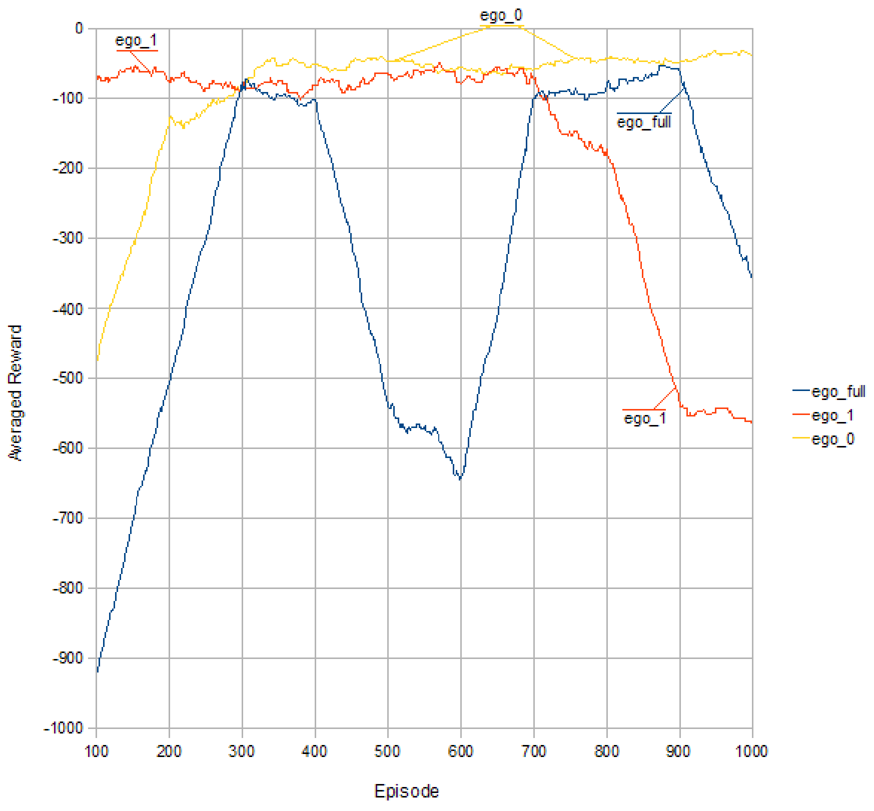

Figure 25: Magnified comparison for ego_full, ego_1 and ego_0 on a 3 × 3 grid.

This wrong causal model scenario can be seen as the fact that the helicopter is not integrated with the DPNs but instead has to perform inference by itself; it does not know the sensor causal model structure, so the least dangerous assumption it can make is the direct dependency between gas presence and sensor readings.

Before investigating the effects that inaccurate models have on the policy learnt, the reference experiment that assumes perfect sensor models is done. It is based on a 5 × 5 grid, in order to have a problem of increasing complexity. 10 runs of this experiments were done and the results, averaged over a window of 100 episodes, are given in Figure 27. ego_full represents the performance of the full egocentric representation (145 features); ego_2, ego_1 and ego_0 are the reduced egocentric representation with reduced view 2 (33 features), 1 (17 features) and 0 (9 features).

Once again, the results validate the hypothesis that there might be a relationship between number of features used to represent the leak belief and grid size which leads to better policies: ego_2 is the best representation, ego_0 the worst. The policy learnt by ego_2 makes the helicopter move toward the centre of the grid and performing sensing actions during its path; once the centre is reached, the helicopter performs further sensing actions and then reports. With ego_1 the helicopter moves toward the centre too but, as what seen for ego_1 in the 3 × 3 case, it performs many steps forward and backward; it still reports correctly because during the time it takes to reach the centre the fixed sensors keep working and the DPNs keep providing reliable soft evidence; approximately half of the times the helicopter reports without performing any sensing action, making it more dependent on the fixed sensors than ego_2. Other times, instead, it reaches the centre, performs many sensing actions and then reports. The number of sensing actions it performs are bigger than those for ego_2, this is due to the high level of generalisation it has for suburbs: they aggregate many cells, and before their

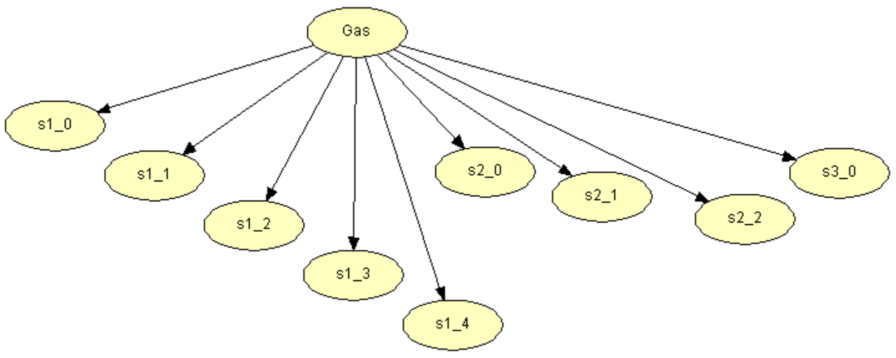

Figure 26: Wrong model used to analyze the impact of inaccurate sensor models have on the policy learning: it is assumed a direct dependency between sensor readings and gas presence.

mean entropy is reduced, since the helicopter does not move from the centre, many timesteps must pass. ego_full, instead, learns almost the same policy that was learnt by entropy in the 2×2 and 3×3 case: it moves chaotically and few times it performs few sensing actions before reporting; this behaviour can be justified by the fact that many of its features are usually not used, and it takes a lot to learn the correct weights, leading to the high dependency on the sensor readings. In fact, the policy learnt has no defined pattern as those for ego_2 and ego_1. ego_0, to conclude, is the worst possible: most of its episodes are aborted, its level of generalization is very high and this does not let it understand the global situation.

In conclusion, ego_2 and ego_1 are the representations which lead to the best policies; these are more flexible than ego_0 and ego_full, and let the helicopter either to rely on fixed sensors (more in case of ego_1) and on its own independent behaviour of moving toward the centre of the grid and sense (in case of ego_2). Their performance is now compared with the 5 different scenarios for inaccurate model.

Some notation is now given. The six scenarios (the five inaccurate ones and the perfect one) are named using composites of two identifying feature: the first one represents the causal model type, the second one the CTPs types. This means that the notations are corr_corr for the perfect model (correct causal model, correct CPTs), corr_soft and corr_hard for the two scenarios with correct causal models but wrong CPTs, specifically softer (that is closer to the uniform distribution) and harder (closer to the deterministic distribution); wrong_corr is used to represent the incorrect causal model shown in Figure 26 and with its CPTs that are computed from corr_corr in order to have the same conditional probability distributions $P(sensor\_readings|Gas)$; analogously, wrong_soft and wrong_hard are based on the wrong causal model structure and with CPTs that are calculated from the corresponding correct model - soft/hard CPTs in the same fashion as it is done for wrong_corr. The CPTs used are shown in Tables 7 and 8.

As it can be noticed, the CPTs for the helicopter sensor, $MEMS$, are not changed. This is based on the assumption that the helicopter should know the sensor model it is carrying. The results about the performance of ego_2 and ego_1, on a 5×5 grid, with respect to the five possible scenarios of incorrect model and the perfect model are given in Figure 28 and 29.

The most striking result is given by the fact that with a wrong causal model, no matter what the CPTs specify, the policy learnt is very bad. Most of the episodes are aborted, that is why the performance is close to -3,000. This is due to the fact that the result of the inference in the wrong model scenario is highly subject to false positives and negatives, and the threshold according to which perform report action cannot be learnt correctly. Both ego_2 and ego_1 do not learn to perform sensing action, they instead keep performing the same moving action or chaotically moving around until,

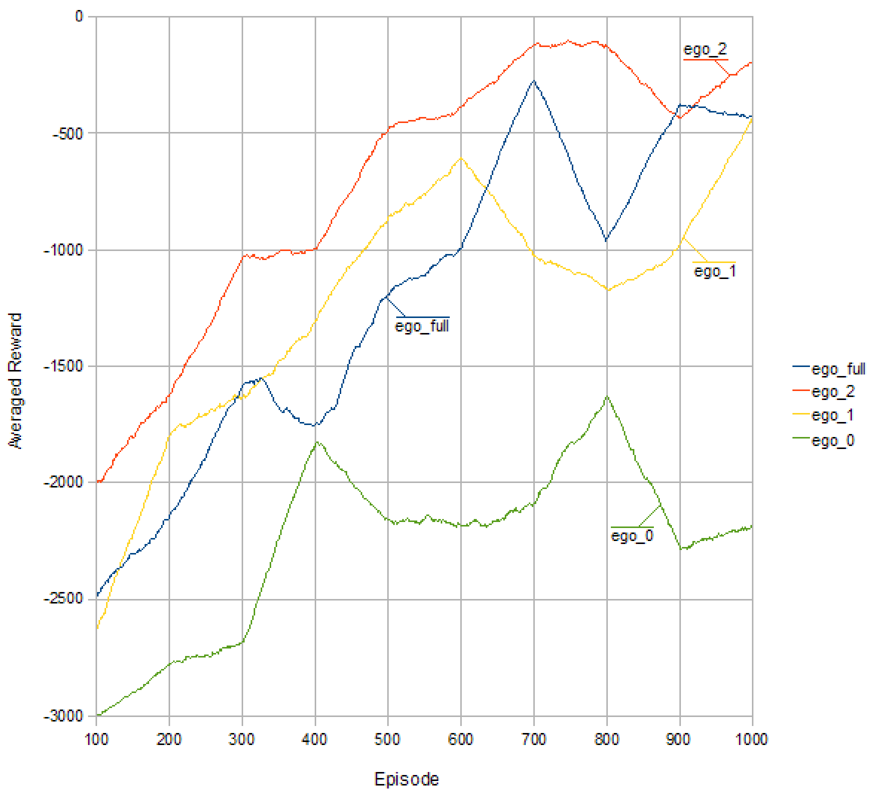

Figure 27: Performance of ego_full, ego_2, ego_1 and ego_0 in a 5×5 grid with perfect sensor models.

eventually, report action is performed.

With correct causal model, instead, the situation gets better. As expected, the performance with correct CPTs is never reached by soft and hard CPTs; ego_1 with soft CPTs has few episodes in which the performance beats the correct CPT version, but whilst the performance for corr_soft does not improve anymore, the performance for corr_corr keeps improving. The difference in performance between correct CPTs and the other two is more marked for ego_2; this is due to the fact that with ego_2 the helicopter performs a correct policy described previously, and since its own sensor model is not changed, the difference is given by the number of wrong reports: in the soft and hard CPT versions it has been noticed that when the helicopter reports a wrong leak position, the distance between the actual and the declared one is usually in the order of 1 or 2 cells. With corr_soft and corr_hard, the tradeoff between reporting fast and reporting correct is affecting the performance: in episodes with an overall good inference the helicopter reports correct leak positions and as a consequence it learns to report quickly; when the inference is overall bad, instead, the helicopter should perform more sensing action, but since it learns to report quickly, wrong locations are declared.

Nonetheless, no matter what kind of sensor model is used, both representations have an increase of their performance; it is very slow when wrong causal model is used, and it is very quick in case of correct causal model. Again, the segmented trend is due to the fact that, among 10 runs, the policy sometimes changes dramatically between intervals, leading to many abort actions resulting into a reward which influences significantly the trend.

The results bring to the conclusion that the integration between DPN and helicopter (RL) is a

probability	corr_corr	corr_soft	corr_hard
$P(Cond\|Gas)$	0.8	0.8	0.8
$P(Cond\|\overline{Gas})$	0.4	0.4	0.4
$P(Ion\|Gas)$	0.8	0.8	0.8
$P(Ion\|\overline{Gas})$	0.4	0.4	0.4
$P(MEMS\|Gas)$	0.8	0.8	0.8
$P(MEMS\|\overline{Gas})$	0.4	0.4	0.4
$P(S1\|Ion)$	0.6	0.51	0.7
$P(S1\|\overline{Cond})$	0.4	0.49	0.3
$P(S2\|Cond)$	0.75	0.65	0.85
$P(S2\|\overline{Cond})$	0.25	0.35	0.15
$P(S3\|MEMS)$	0.9	0.9	0.9
$P(S3\|\overline{MEMS})$	0.1	0.1	0.1

Table 7: CPTs for the DPNs with correct causal sensor models.

probability	wrong_corr	wrong_soft	wrong_hard
$P(S1\|Gas)$	0.56	0.506	0.62
$P(S1\|\overline{Gas})$	0.37	0.504	0.42
$P(S2\|Gas)$	0.65	0.59	0.71
$P(S2\|\overline{Gas})$	0.39	0.44	0.38
$P(S3\|Gas)$	0.74	0.74	0.74
$P(S3\|\overline{Gas})$	0.33	0.33	0.33

Table 8: CPTs for the DPNs with wrong causal sensor model.

necessary condition for good mobile sensor control learning.

7.4 Set C: Scaling to Large Grids

In this final scenario, only the perfect sensor model of the DPNs is considered and the environment is increased to a 10×10 grid. The egocentric full representation is abandoned, given the bad policy learnt in the 5×5 case, despite its acceptable performance. Here, only the egocentric representation with reduced view 5 (129 features to represent leak belief), 2 (33 features) and 1 (17 features) are investigated. Again, 10 runs per experiment were done and their results, averaged over a window of 100 episodes, are given in Figure 30. ego_5, ego_2 and ego_1 represent the corresponding reduced view performances.

Even if the performance is still bad, all the representations improve their averaged reward during the experiment; this means that with appropriate parameters α, number of learning episodes and so on, increase in performance can be easily achieved. The best performance is given by ego_5, which leads to the final hypothesis that it is sufficient to have a number of features the closest possible to the grid size to perform well: too many features are too sparse and make the policy too much dependent on the sensor model; too few features perform a huge aggregation of cells especially in the suburbs at the top-left, top-right, bottom-left and bottom-right: the information they carry is too vague. A reason according to which the helicopter tends to move to the centre of the grid is given by the fact that it is the best way to represent information about all cells of the environment; furthermore, from a central position, the aggregation performed at the suburbs is balanced with the precise representation of the centre area at best. Furthermore, even if the experiments shown an early stage of results, a general pattern for control policy was noticed: the helicopter tends to move to the centre and perform sensing actions; sometimes it also leaves the centre and moves toward the leak location; other times, instead, the helicopter goes toward the leak location without reaching the centre. Intuitively, the centre of the environment is a strategic place also for subsequent movement to cells far away from the initial position of the helicopter at timestep 0.

The final conclusion is that by keeping a leak belief representation with a number of features, speed of improvement, robustness with respect to incorrect models and to grid sizes.

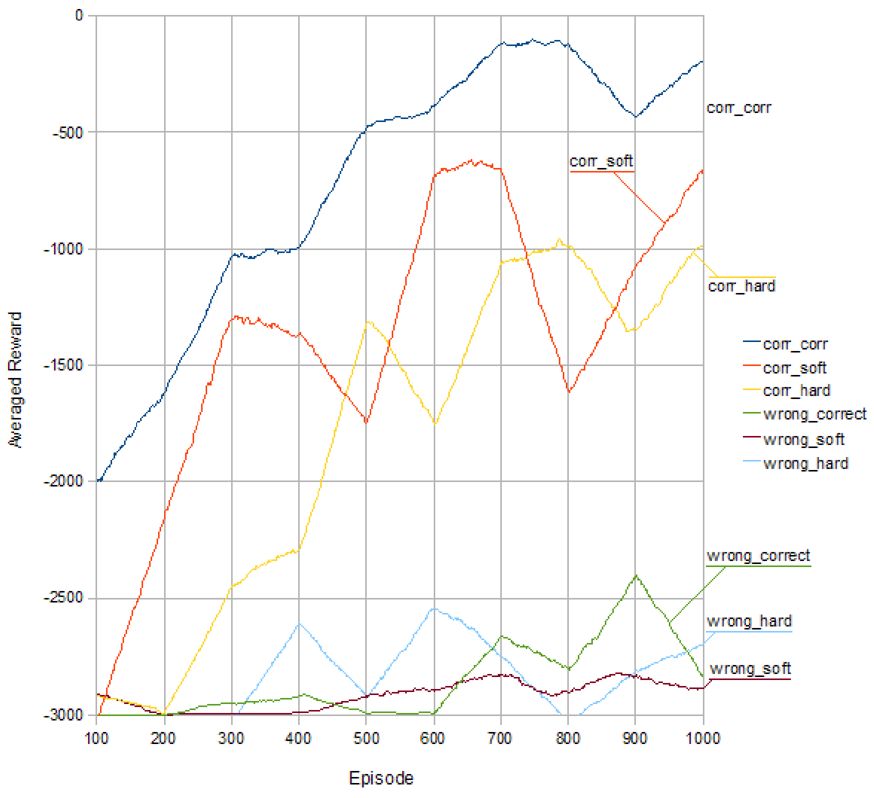

Figure 28: Reduced egocentric representation performance with respect to different sensor models in a 5×5 grid and $v = 2$.

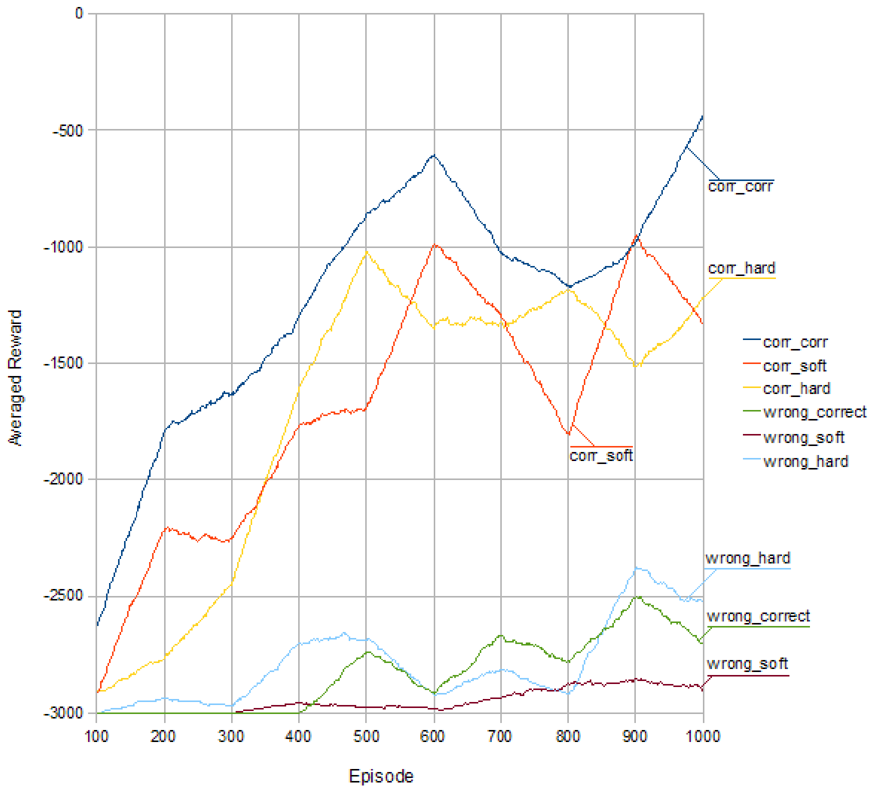

Figure 29: Reduced egocentric representation performance with respect to different sensor models in a 5×5 grid and $v = 1$.

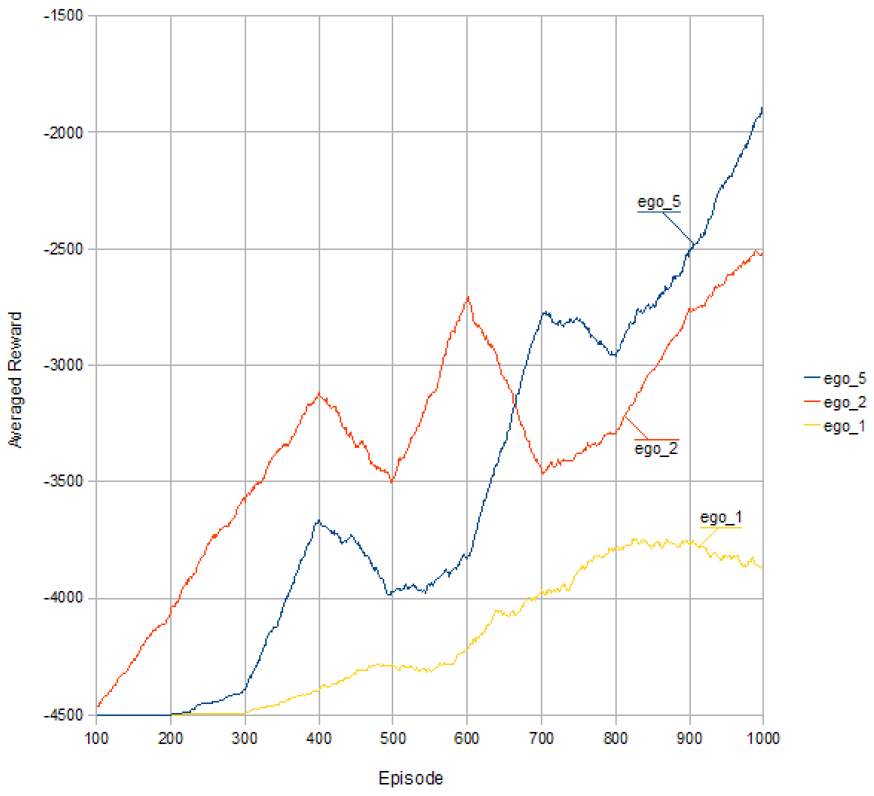

Figure 30: Reduced egocentric representation comparisons in a 10×10 grid.

8 Validating Software Implementation

In order to gather results validating the theories developed in this thesis, an applicative simulation of the environment and the helicopter agent were implemented in C/C++. The DPNs and *Mother Nature*, which are part of the environment, represented a simulation of a bigger fusion system, the distributed perception network toolkit (DPNT), a Java-based software package developed by Universiteit van Amsterdam which can be used to define a multi-agent Bayesian reasoning system for information fusion based on the theory of distributed perception networks [22].

The simulation of the DPNs and *Mother Nature* had to be done because of technical programming issues: the two applications, that is the RL and DPNT, were written with different codes, and the fastest way to let them communicate was through a file messaging synchronization protocol. The communication was done from the environment to the DPNT in order to perform sampling and inference, and the result of it was passed back to the environment; then, the environment was passing the inferences as observation to the helicopter as thoroughly explained previously. Furthermore, since the DPNT is a multi-threaded application, some initial synchronization was needed. A schema representing the communication between helicopter, environment and DPNT is given in Figure 31.

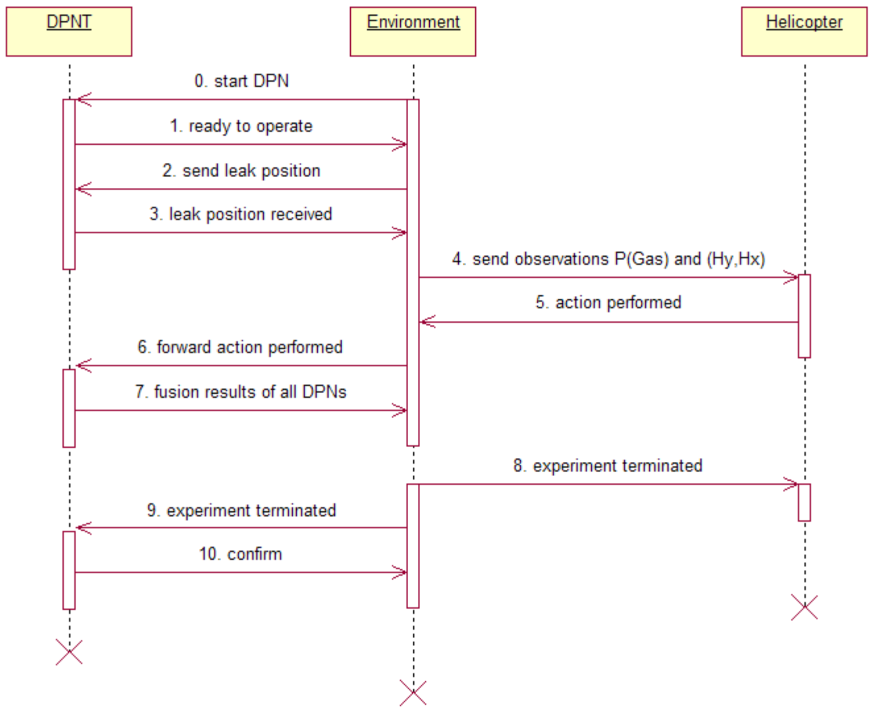

Figure 31: Message passing protocol between helicopter, environment and DPNT.

Apart from the two initial and the last three synchronization messages, which were done just once per experiment, the main bottleneck of this communication was given by the other messages. Messages 2 and 3 were sent at each episode (this means 51,000 times for experiments of set A and B and 101,000 for set C), messages from 4 to 7 were done in loop, for each timestep, and this represented the biggest obstacle. After many tweaks in the DPNT code, the time elapsed from message 4 to the next message 4 was still in the order of 6 seconds. Clearly, this time was not efficient at all.

The problem was given by the fact that the DPNT was created with the idea of having only fixed sensor agents that only occasionally disappear from the network of agents; with the implementation of a mobile agent, instead, the fusion process and the removal of such sensor was occurring at each timestep, in order to keep the system coherent. The tweaks were done ad-hoc for the purpose of having an integrated system and this caused an increase of instability in the whole integration.

Because of the simulation of the DPNT was used, in order to make possible to have valuable results, given the shortage of time left to the thesis, the simplistic assumption (A4), of having the same DPN structure in each cell of the environment, had been done.

However, some very small experiments using the actual DPNT were performed, in order to validate the possibility of having a working integrated system. These experiments were based on a 1×2 grid, and even if the tweaks made the whole integrated system very unstable and no experiments could have been completed, the soft evidence computed by the DPNT could have been successfully used by the helicopter to perform inference about leak location. Based on these results, the simulation of the DPNT was built inside the C/C++, in order to have an exact representation of the DPNT.

Studies for the application of DPNs through the use of the DPNT for crisis management have been done, for example, in [23]; the fact that an intelligent and autonomous agent for mobile sensor control can be successfully integrated to the DPNT, then, represents an improvement to the DPNT, which becomes more flexible (the sensor agents can move independently) and can perform more powerful inference.

9 Discussion

This chapter brings valuable discussions about the results achieved. The main result is that mobile sensor control can be successfully achieved by integrating RL techniques with DPN fusion systems, under the necessary condition that the DPNs keep at least a correct causal model that represents the sensor structures.

The chemical leak detection task is a large POMDP problem, and even if the helicopter has a complete knowledge of the model of the environment, planning cannot be applied. However, due to the fact that it knows the model of the environment, the helicopter can maintain its belief state and consider the problem as a continuous space MDP. This is why model-free methods can be used. This represents the most innovative contribution that this thesis provides to the techniques for POMDP solving.

The first set of experiments were mainly focused on the definition of good belief state and related value function representations. The results gathered lead to the following statements.

The use of discretised representation of the belief state and a tabular approach for q-value learning is infeasible: the closer the discretisation is to the continuous representation, the better the performance; however, the number of entries of the table are exponentially proportional to the environment size and level of discretisation, and even if the performance is increasing, it remains very low, since the problems related to the exploration and exploitation of the belief state space occur.

By representing the belief state in a continuous manner and learning a linear mapping between belief state-action pairs into q-values, significant results can be achieved. The minimum requirement is the representation of the leak belief through entropies and not probabilities: the probabilities represent the uncertainty in a non-linear way: the closer they are to 0 or 1, the lower the uncertainty about the gas presence. Entropies, on the other hand, solve this issue.

The problem is not only how to represent the uncertainty in a linear way: the control policy learnt should be such that the helicopter performs sensing actions because its own sensor is more reliable than the fixed sensors in the environment; it is important that it knows which entropy is related to which cell of the environment with respect to its point of view. This is why the representation of the leak belief according to the global ordering in coordinates of the environment leads to wrong policies where no sensing action is performed but instead the helicopter just waits for the right time to report. The explanation is given by the fact that every time the helicopter changes its location, so does the correspondent current cell's entropy in the leak belief. This means that the linear mapping should be able to represent at the same time the q-value when the entropy of a particular cell is the same as the helicopter's current cell and when it is not, and this is not possible.

Another important result achieved through this set of experiments was also the definition of the hypothesis that there might be some relationship between the size of the leak belief and the environment which can lead to good policies: with a full representation of the egocentric view, a lot of redundancy is introduced: most of the features of the leak belief are used only in few occasions, and this makes their parameter vector learning very slow. Furthermore, it has been noticed that when the extensive representation is used, the policy learnt is mostly passive: the helicopter relies almost completely on the sensor models and it only learns when to report. If the reduced view is too small, instead, the aggregation performed at the suburbs is too big, and the mean entropy value is no longer a good representation; the result is once again a policy highly dependent on fixed sensors. The helicopter does nothing but move chaotically, when in the meantime the DPNs keep "working for it", and then it reports. Another problem with the high level of aggregation is the fact that the policy learnt changes often during experiments, and many times it does not even learn when to report.

The best approach is in the middle: by keeping a leak belief with the closest size possible to the grid size, the policy learnt is the best. On a 2×2 grid, the best reduced view is the one that leads to a leak belief of 9 features only; the same number of features were required for the best policy found on a 3×3 grid. For a 5×5 grid, instead, the reduced view equal to 2 (33 features) gives the best results. A reduced view equal to 5 (129 features), instead, leads to the best representation for a 10×10 grid.

Over all the experiments, from those based on 3×3 up to 10×10 grid sizes, a common behaviour among best policies/reduced view representations was found: the helicopter tends to move to the centre of the environment, in order to have a complete reduced egocentric representation of the environment's uncertainty; with a leak belief of the same size as the environment, furthermore, only a few of the features will not represent cells of the environment.

Apart from a good belief state representation, the need for good sensor models is required. Experiments focused on the effect that different sensor models have on the control policy, leading to the conclusion that a good causal model for the sensors is necessary for good policy learning, since their inference results are less subject to false positives and negatives.

The need for a good inference model can be seen by the small example shown in Figures 32 and 33. They correspond to the models introduced in Section 7.3. The first one represents the perfect causal sensor model, the second one represents the wrong causal sensor model and its CPTs were defined such that they could generate the same probability of having sensor readings given *Gas*. In this example, at timestep 0, the helicopter performs a sensing action. Let assume that 5 readings for Ion are true, the 3 readings for $Cond$ are false, and the reading for $MEMS$ is true. For these observations inference in the two models will produce very different results: the correct model version has a belief about *Gas* of 53.17%, whilst the wrong version has a belief about *Gas* of 49.53%. Even if the beliefs are still very uncertain, and according to the best policy learnt it would lead to further sensing actions, the result of the inference of the wrong result will for sure influence the policy learnt. The importance of having good structure more than having good CPTs is throroughly discussed in [2], and the results of this thesis go accordingly.

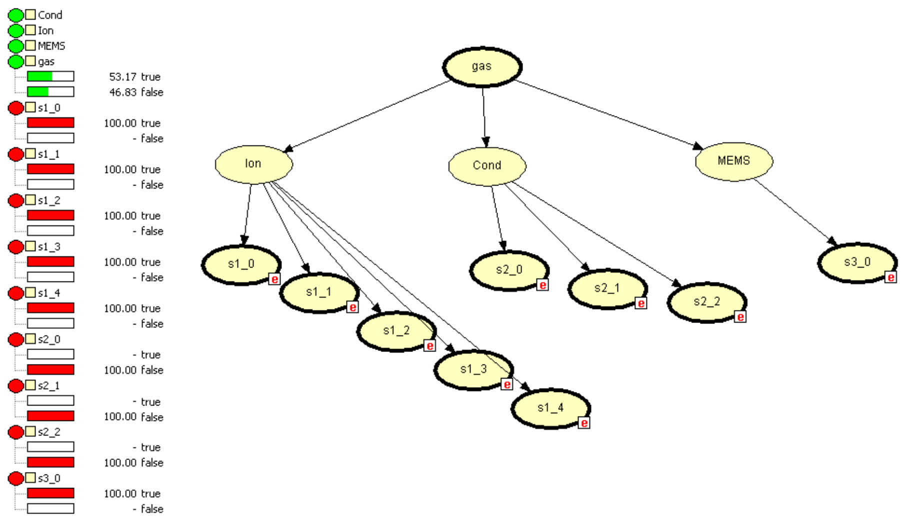

Figure 32: Results of the inference about *Gas* for the integrated correct model representation

The final conclusion, hence, is that under the necessary condition of correct causal sensor models, a reduced view representation of the leak belief using entropies can lead to good policies which are scalable to large grids and that lead to a general behaviour of moving to the centre of the environment

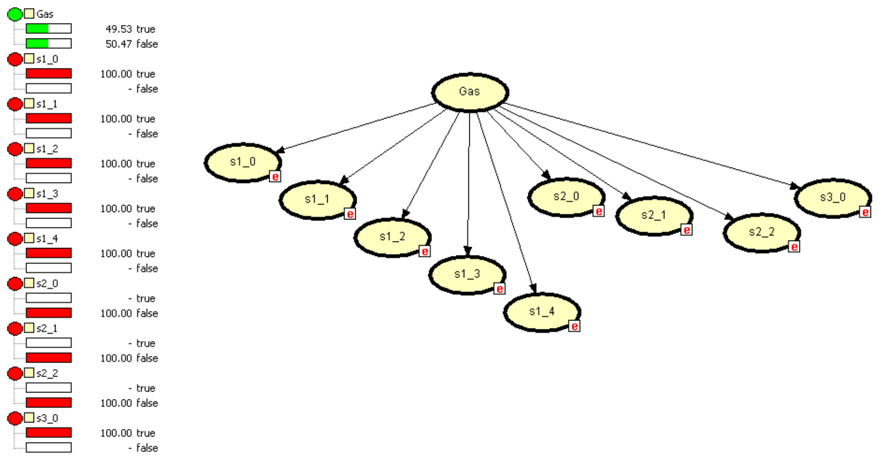

Figure 33: Results of the inference about *Gas* for the non integrated version

in order to have a complete view of the uncertainty about the leak position, perform sensing actions in appropriate cells and then report a correct leak position the quickest possible.

9.1 Related Work

Most of the work on mobile sensor control is based on multi-agent systems organized in communicative networks. [24] describes an event-based motion control for mobile sensor networks; the sensors are in communication with each other and all moves according to the presence of an event, such as a chemical spill. [25] presents a stable control strategy for groups of vehicles to move and reconfigure cooperatively in response to a sensed, distributed environment. Each vehicle in the group serves as a mobile sensor and the vehicle network as a mobile and reconfigurable sensor array. Their paper is focused on gradient climbing missions in which the mobile sensor network seeks out local maxima or minima in order to optimize detection and measurement of features of a particular field. The network can adapt its configuration in response to the sensed environment in order to optimize its gradient climb. [26], instead, describes a swarm-based fuzzy logic control (FLC) mobile sensor network approach for collaboratively locating the hazardous contaminants in an unknown large-scale area. The mobile sensor network is composed of a collection of distributed robots, each of which has limited sensing, intelligence and communication capabilities. By gathering other nodes locations and measurement data, each nodes FLC can independently determine its next optimal deployment location. [27] adopts a filtering approach to fusing local sensor data into a global model of the environment. Their approach is based on the use of a communication network in which each mobile sensor diffuses its information. Their network of sensors has characteristics similar to the DPN: it is scalable and fully decentralized and allows changing network topologies and anonymous agents to be added and subtracted at any time. However, centralised control is done.

9.2 Future Work

The first ideas for future works come from the weakening of the assumptions made in Chapter 6 and, from the approaches defined in this thesis, bring modifications in order to have an initial starting research point.

The helicopter moves toward the centre of the environment for two reasons: from there it has the best representation of the overall uncertainty of the true state and from there it can move to other locations with the minimum number of actions. This was due to the belief representation and the fact that there is only only leak in the environment. The sensing actions performed by the helicopter aim to remove the false positives and negatives resulting from the inference obtained from less performing fixed sensors; by doing that, the leak can is found. Another way of seeing the path the helicopter performs is the following: it is dependent on the probability distribution $P(Gas|L)$: the false positives are more likely to happen next to the leak, this is why the helicopter moves along the direction which lead it toward the leak. The DPNs are reliable enough to keep the uncertainty for cells far from the leak low, and this is another reason why the reduced view is very effective: areas with low uncertainty are grouped together and the helicopter does not have to care about them. The path the helicopter does can be seen as a hill climbing, where the height is represented by the values of its leak belief.

In case of more than one leak, more hills must be climbed. The ideal behaviour does not change: the helicopter still has to remove the false positives and negatives; the only difference is that an episode should not end after one report. A new and unique action that terminates the episode should be introduced. Once the helicopter reports a leak, it will recieve a reward according to its correct declaration, then it will update its belief (after recieving the reward it can exactly know if the leak was there or not) and then either look after the next leak (in case of correct report) or focusing again on the old one (otherwise). If the number of leaks are not fixed then the problem becomes more complex, and the helicopter must know exactly when to end an episode. However, nothing forbids to start approaching this new problem from the results and theories of this thesis.

The thesis' scenario is based on the following assumption: the time needed to end an episode is such that the chemical gas contamination does not change. In case the leak changes its contamination area, for example because of diffusion and air flow, then the approach of this thesis should be improved, since in this thesis the main idea is that the leak is present somewhere in the contaminated area. Some few changes can be made in order to adapt the thesis achievements to this new scenario: since that the helicopter knows $P(Gas|L_{y,x})$, the physical laws that define the chemical spread in the environment, it is possible to assume that the helicopter has knowledge about the effects that the wind has on a chemical cloud and its diffusion properties. Then, the helicopter can "rewind the tape", that is, can more or less retrieve the original situation present at the moment where the it was still in the hangar, that is, no alarm was switched on. In order to "rewind the tape" it might be needed to add a history that builds the trail of sensor readings and this can be achieved with minor work, given the quasi-static behaviour of the sensor agents of the DPNs which keep the complete history of all sensor readings.

In the experimental setups made, the DPNT was simulated in the C/C++ code of the environment, and the simplistic assumption that there was only one type of DPN, composed of the same fixed sensor models was made. Future works can be focused on the definition of more complex scenarios, such as the definition of different DPN models, coexistence of both wrong and correct sensor models in the environment, not uniform distribution of DPNs in each cell of the environment and so on. Furthermore, with further work on the software integration between the DPNT and the RL part, even more complex scenarios can be formulated.

Studies about the relationship between the leak belief obtained from a certain reduced view and environment size could be done, in order to validate the hypothesis of this thesis, that is, that the sizes of those components should be similar. Other studies aiming to find different ways of defining suburban areas could be done.

The thesis was based on the assumption that the helicopter was always able to perform sensing actions; in real life it might happen that good sensors need time to be cleaned and reset; studies in which for some timesteps the sensing action is not available could be done, in order to see how the helicopter behaves when it cannot sense; for example, while the sensing action is disabled, it could decide to move somewhere else, it could find more convenient to stand still and wait for the sensor to be ready, and so on.

The true state transition function used in this thesis is deterministic; the next step can be its transformation into a stochastic one; further studies in which the time to travel from a location to another could be done; this would likely make the problem semi-Markovian as for example in [20], but it would be for sure a more realistic scenario.

The studies could move to a multi-agent scenario with more than one helicopter solving the leak detection task, by allowing cooperation between the agents (for example they can communicate about their inference, they can share rewards etc.); it could be interesting to see the combination of this scenario with the one that assumes more than one chemical leak.

The very final work, clearly, remains the realization of a real-world implementation of the theories of this thesis: fixed sensors can be organised in DPNs, an intelligent and autonomous helicopter based on the control policy found through the use of the theories of this thesis can be realized, controlled tests about chemical leak detection can be performed, in order to lead to a final robust and reliable installation of the system into real industrialised areas.

10 Conclusions

The chemical leak detection task performed by an autonomous and intelligent helicopter was the case study for the theories of this thesis. The fact that the helicopter has a partial view of the true state, that is it does not know where the leak is occurring, makes this problem a partially observable Markov decision process (POMDPs). It is actually a particular POMDP problem, for which the uncertainty that must be solved is the task to be done and not some kind of noise in the observation of the true state. The problem is a large one, since the helicopter must detect the leak even in large grids; this makes the usual planning approach for POMDP problems unfeasible. However, it turns out that the helicopter can have a perfect knowledge of the model of the environment: this let the reduction of the POMDP problem into a continuous space Markov decision process (MDP) one, where the state space is represented by the belief state space. The belief transition function is known, so model-free methods can be used in order to solve the problem.

According to how the belief state was representing the uncertainty about the leak position, two approaches were defined. The first one considered a discretised belief state representation and through a tabular approach the helicopter could learn belief-action values. The second approach, instead, maintains the continuous belief state and assumes a linear mapping of belief-actions into values; linear function approximation method's gradient descent was used in order to learn the weights of such linear mapping.

Results showed that the discretised belief representation is unfeasible: the number of entries of the table are exponentially proportional with respect to the environment size and the level of discretisation of the leak belief. The main conclusion obtained was that the more defined the discretisation is, the better the performance. A drawback was given by the size of the table, that could not been explored properly.

The continuous belief representation and the related linear function approximation was investigated. Four different types of state belief representation were defined. The results showed that probabilities represent the uncertainty in a way which is not linear with respect to the function aimed to be learned; a significant improvement is achieved by representing the uncertainty through information entropy of the leak beliefs: the entropy is a linear measure for uncertainty, hence, it is perfect for the linear function sought. It has also been noticed that by organizing the beliefs according to the helicopter's egocentric view, the results is the definition of correct policies; furthermore, by representing the belief through both aggregation of cells and precise representation of single cells, the system become very flexible with respect to the changes of its belief and the sensor models: the helicopter tends to move to the centre of the environment and perform sensing actions, then report the correct leak location. Different scenarios in which imperfect sensor model were also investigated; this led to the conclusion that correct causal sensor models are a necessary condition for good control policy learning.

The final result of the thesis, hence, is the proof that RL is an effective tool for doing mobile sensor control under the necessary condition that the sensor causal model are correct, so the integration of those with the mobile sensor into DPNs can lead to policies which are robust against false positives, false negatives and also with respect to large environments.

References

[1] Charles M. Grinstead, J. Laurie Snell - *Introduction to Probability*

[2] Judea Pearl - *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* - Morgan Kaufmann

[3] Gregor Pavlin, Patrick de Oude, Marinus Maris, Thomas Hood - *Distributed Perception Networks: An Architecture for Information Fusion Systems Based on Causal Probabilistic Models* - In International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany, 2006.

[4] Gregor Pavlin, Marinus Maris, Jan Nunnink - *An Agent-Based Approach to Distributed Data and Information Fusion* - In IEEE/WIC/ACM Joint Conference on Intelligent Agent Technology, pages 466-470, 2004.

[5] Patrick de Oude, Brammert Ottens, Gregor Pavlin - *Information Fusion With Distributed Probabilistic Networks* - In Artificial Intelligence and Applications, pages 195-201, Innsbruck, Austria, 2005.

[6] Gregor Pavlin, Jan Nunnink - *Probabilistic Resource Allocation In Distributed Fusion Systems*

[7] David Heckerman - *A Tutorial on Learning With Bayesian Networks* - March 1995 (Revised November 1996), Technical Report MSR-TR-95-06 - Microsoft Research Advanced Technology Division.

[8] Stuart Russel, Peter Norvig - *Artificial Intelligence - A Modern Approach* - Prentice Hall Series in Artificial Intelligence

[9] Tom Mitchell - *Machine Learning* - McGraw Hill

[10] Anthony R. Cassandra - *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes* - Ph.D. Thesis. Brown University, Department of Computer Science, Providence, RI, 1998

[11] Richard S. Sutton and Andrew G. Barto - *Reinforcement Learning: An Introduction* - The MIT Press

[12] Chip Reinhardt - *Taxi Cab Geometry: History and Applications* - Stevensville High School, Stevensville, Montana, Accepted with revisions: 27. July 2004

[13] Xavier Décoret - *Manhattan distance of a point and a line* - December 26, 2006

[14] Ningning Zhou, Alice Agogino, Kristofer S.J. Pister - *Automated Design Synthesis for Micro Electro Mechanical Systems (MEMS)* - Proceedings of DETC 2002: Design Automation Sep. 29-Oct. 2, 2002, Montreal, Canada

[15] Eugene Charniak - *Bayesian Networks without Tears* - The American Association for Artificial Intelligence

[16] Finn V. Jensen - *Bayesian Networks and Decision Graphs* - Springer

[17] Patrick de Oude, Gregor Pavlin - *A Modular Approach to Adaptive Bayesian Information Fusion*

[18] Kevin P. Murphy - *Dynamic Bayesian Networks* - 12 November 2002

[19] Nikos Vlassis - *A Concise Introduction to Multiagent Systems and Distributed AI* - Universiteit van Amsterdam

[20] Thomas G. Dietterich - *Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition* - Journal of Artificial Intelligence Research 13 (2000) 277 - 303 - submitted 11/99; published 11/00

[21] Richard S. Sutton, Andrew G. Barto - *Reinforcement Learning - an Introduction* - MIT Press, Cambridge, MA, 1998

[22] D-Cis Lab Distributed Perception Networks - *http://forge.decis.nl/projects/dpn*

[23] Marinus Maris, Gregor Pavlin - *Distributed Perception Networks for Crisis Management* - Proceedings of the 3rd International ISCRAM Conference (B. Van de Walle and M. Turoff, eds.), Newark, NJ (USA), May 2006

[24] Zack Butler and Daniela Rus - *Event-Based Motion Control for Mobile-Sensor Networks* - Dartmouth College

[25] Petter Ögren, Edward Fiorelli, Naomi Ehrich Leonard - *Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment* - IEEE Transactions on Automatic Control, vol. 49, no. 8, AUGUST 2004

[26] X. Cui, T. Hardin, R. K. Ragade, and A. S. Elmaghraby - *A Swarm-based Fuzzy Logic Control Mobile Sensor Network for Hazardous Contaminants Localization* - Computer Engineering and Computer Science Department University of Louisville, KY 40292

[27] Kevin M. Lynch, Ira B. Schwartz, Peng Yang, Randy A. Freeman - *Decentralized Environmental Modeling by Mobile Sensor Networks* - IEEE Transactions on Robotics, Vol. 24, NO. 3, JUNE 2008