12th International Conference on Information Fusion
Seattle, WA, USA, July 6-9, 2009

Integrating Distributed Bayesian Inference and Reinforcement Learning for Sensor Management

Corrado Grappiolo **Shimon Whiteson** **+Gregor Pavlin** **Bram Bakker**
ISLA,University of Amsterdam, +Thales Research and Technology, D-CIS lab,
Kruislaan 403, 1098SJ, Amsterdam, Postbus 90, 2600 AB Delft,
The Netherlands the Netherlands

Abstract − *This paper introduces a sensor management approach that integrates* distributed Bayesian inference *(DBI) and* reinforcement learning *(RL). DBI is implemented using* distributed perception networks *(DPNs), a multiagent approach to performing efficient inference, while RL is used to automatically discover a mapping from the beliefs generated by the DPNs to the actions that enable active sensors to gather the most useful observations. The resulting method is evaluated on a simulation of a chemical leak localization task and the results demonstrate 1) that the integrated approach can learn policies that perform effective sensor management, 2) that inference based on a correct observation model, which the DPNs make feasible, is critical to performance, and 3) that the system scales to larger versions of the task.*

Keywords: Sensor management, distributed Bayesian inference, reinforcement learning, POMDPs.

1 Introduction

Recent advances in sensing, communication and processing technology offer new opportunities for advanced situation assessment. Often the primary challenge is to perform efficient inference about phenomena that cannot be directly observed, given observations from sensors that may be heterogeneous and unreliable. For example, in industrial areas where chemical leaks can occur, rapid and accurate detection and localization of such leaks is critical to minimizing human casualties. In principle, this is a search problem [14] whose solution requires efficient fusion of information stemming from various, spatially distributed sensors. In addition, some of the sensors can be actively controlled. For example, the system may be capable of adjusting sensor settings (e.g., determining sensitivity vs. specificity) or turning certain sensors on or off. Or, as in the chemical leak scenario, some types of sensors may be too expensive to install in fixed locations throughout the area. Instead, it may be more practical to mount such sensors on a mobile device like an autonomous helicopter, which can sweep the area in search of leaks.

In other words, in the presented search challenge, the problem of sensor fusion is augmented by the problem of *sensor management* [8], i.e. we must determine the most efficient way for such a helicopter to navigate its environment so as to gather the most useful observations for rapid and accurate detection of the hidden phenomena. In this paper, we introduce a novel solution to this problem that integrates *distributed Bayesian inference* (DBI) and *reinforcement learning* (RL) [17].

DBI is implemented using *distributed perception networks* (DPNs) [2, 9], a multiagent approach to modeling the probabilistic relationships between the sensors and the hidden phenomena. Such models make it possible to efficiently fuse observations from all the sensors and compute updated beliefs about the hidden phenomena. Unlike centralized approaches [10], which require one agent to bear the entire burden of inference, DPNs enable sensors to individually perform inference using local models and the data they collect, and then share results with other components of the system. Hence, they distribute knowledge and computational requirements among the sensors and adapt flexibly to changes in network structure.

RL, on the other hand, is a way for an autonomous agent to learn, through interaction with the environment, a control policy that maximizes its long-term expected reward. Using RL, an active sensor can automatically discover a mapping from the beliefs generated by the DPNs to the actions that enable it to gather the observations most useful for identifying the hidden phenomena. In particular, we treat this problem as a *partially observable Markov decision process* (POMDP) [4] and train a linear function approximator on-line to estimate the mapping from beliefs to actions.

Often approaches to sensor management strive to compute at runtime the expected reduction in uncertainty that particular action sequences will create [14]. However, given the complexity of the probabilistic models involved and the vast number of action sequences to be considered, this approach can be too computationally expensive for practical use. By contrast, our approach generates complete control policies specifying

what action to take for every belief. Consulting the policy at runtime has trivial cost.

Furthermore, approaches that merely strive to reduce uncertainty about the hidden phenomena do not allow the agent to directly reason about the *value* of uncertainty reduction. In the RL approach, potential policies are evaluated with respect to the long-term reward they accrue, thus seeking an optimal balance between uncertainty reduction and other factors such as speed. For example, an RL agent can automatically determine when obtaining additional observations is no longer worth the delay, which methods that merely reduce uncertainty cannot do.

Previous work using the POMDP framework for sensor management [16] employs centralized inference to update beliefs and off-line planning to find control policies, both of which are computationally expensive. By contrast, we use DPNs to perform distributed inference and on-line linear function approximation to learn the policy, so both belief and policy updates are computationally efficient.

Hence, the primary contribution of this work is a principled, efficient approach to sensor management. From the perspective of DBI, this approach is an improvement because it adds a way to intelligently control mobile sensors that contribute observations for inference. From the perspective of POMDPs and RL, this approach is also an improvement because it offers a way to use DPNs to efficiently maintain, in a distributed manner, the beliefs necessary for reasoning about POMDPs.

We evaluate this method on a simulation of the chemical leak localization task. Our results demonstrate 1) that the integrated approach can learn policies that perform effective sensor management, 2) that inference based on a correct observation model, which the DPNs make feasible, is critical to performance, and 3) that the system scales to larger versions of the task.

The remainder of this paper is organized as follows. Section 2 provides background about DPNs and RL and Section 3 details the chemical leak localization problem. Section 4 describes the integration of DPNs and RL for sensor management. Section 5 presents and discusses experimental results and Section 6 concludes and outlines directions for future work.

2 Background

The approach to sensor management presented in this paper is a combination of two components: DPNs to perform distributed Bayesian inference and RL to find control policies for mobile sensors. In this section, we provide brief background about these components.

2.1 Distributed Perception Networks

One way of performing distributed reasoning about causal stochastic processes is using a multiagent approach, such as distributed perception networks (DPNs) [9]. DPNs are modular, self-configurable inference systems that can efficiently interpret large amounts of heterogeneous information through cooperation among various DPN agents. Each such agent implements an inference module with limited domain expertise (e.g., about one particular sensor), which is represented by a local Bayesian network. The network is a directed acyclic graph that uses conditional probability tables to describe the relationships between variables [10].

To perform inference at runtime, special fusion modules integrate the expertise provided by these agents into complete distributed models that correctly capture probabilistic relationships between the sensors and the hidden phenomena. This integration is performed by applying simple design and assembly rules based on the locality of causal relations explicitly represented by Bayesian networks [10]. Contrary to other approaches to distributed inference [18, 6], the DPN framework does not rely on the computation of secondary inference structures such as junction trees, which span multiple modules. Compilation of such structures can be computationally expensive; it requires recursion through a system of inference modules (i.e. agents), which can be a serious obstacle if the sensor constellations change frequently at runtime. In DPNs, such compilation is avoided through systematic instantiation of variables in Markov boundaries [9]. The resulting inference is equivalent to exact belief propagation in a monolithic causal Bayesian network. Figure 1 shows a simple DPN used in the leak localization task. Note, however, that DPNs support inference for arbitrarily complex Bayesian networks (see [9] for more complex examples).

However, centralized control is no longer required. As a result, the computational load of inference can be distributed throughout the system. Furthermore, it is no longer necessary to gather knowledge about all the sensors in the system into one place. Instead, each sensor can be wrapped in a DPN agent that contributes a model of only that sensor, without needing knowledge of disparate sensors in other parts of the system. In other words, each agent contributes a local model which relates the sensor's observations with the rest of the distributed causal model. As a result, DPNs can efficiently cope with changing constellations of sensors. This is an important feature in applications like chemical leak localization, where sensor availability may not be known in advance and mobile sensors are constantly joining and leaving different parts of the environment.

2.2 Reinforcement Learning

Reinforcement learning (RL) [17] is a family of methods for solving sequential decision problems given only an immediate reward signal as feedback, in the absence of examples or correct or incorrect behavior. The traditional model for such problems is the *Markov decision*

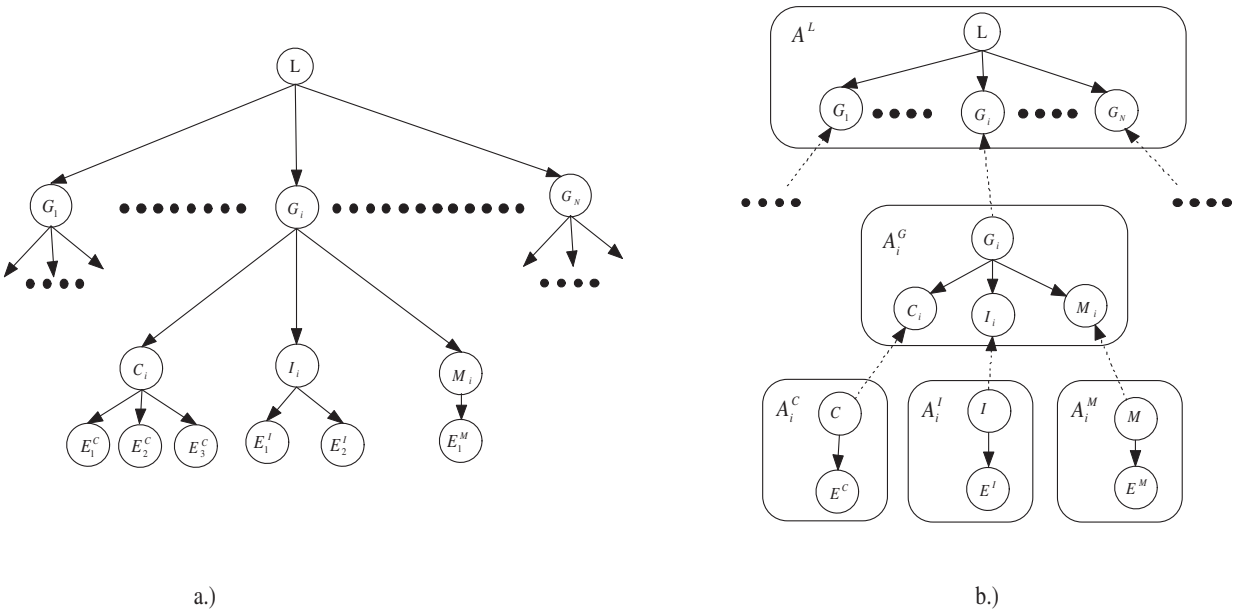

Figure 1: a) a monolithic causal Bayesian network describing the relationships between various senor observations (leaf nodes) and the location of a chemical leak (root node). b) an equivalent distributed network consisting of DPN agents (solid boxes). Dashed arrows show the flow of inter-agent messages containing partial fusion results.

process (MDP). In each timestep of an MDP, the agent perceives the environment's current state, s and selects an action a. The environment responds with a reward signal r and a new state s'. Since MDPs satisfy the *Markov property*, the transition and reward dynamics depend only on the current state, not the whole state history. If a model of the transition and reward dynamics is known, MDPs can be solved off-line using planning techniques. If a model is not known, an agent can still learn an optimal policy by interacting with the environment and using *temporal-difference* methods such as Q-learning [17].

Cases where the Markov property does not hold can be modeled using a *partially observable MDP* (POMDP) [4], where the state signal is replaced by an observation correlated with the state. These observations are not Markov but the agent can maintain a belief b, a distribution over the current state, which is Markov. As a result, if the transition, reward, and observation models are known, POMDPs can also be solved off-line using planning techniques.

The chemical leak localization, and other sensor management tasks, can be modeled as POMDPs, where the state describes the hidden phenomena and the observations consist of readings from fixed and mobile sensors. In Section 4 we describe how DPNs can be used to efficiently maintain updated beliefs given observations from distributed sensors and how RL can be used to tackle the resulting POMDP given such beliefs.

3 Chemical Leak Localization

In many industrial areas, the threat of chemical leaks is a serious concern. Discovering such leaks rapidly is critical to minimizing human casualties and property damage. Some types of sensors are inexpensive enough to distribute throughout the environment. Unfortunately, they are not very reliable. Reliable sensors exist, but are too expensive to put in each region of the environment. However, they can be mounted on a mobile device like an autonomous helicopter, which can sweep the area in search of leaks. In the chemical leak localization (search) problem, we seek a policy for an agent controlling such a mobile sensor.

The environment is simulated using an $m \times m$ square grid consisting of $n = m^2$ locations $l_1, \ldots, l_n$. Each location l_i contains one ionization sensor I_i and one conductivity sensor C_i. In addition, there is one mobile sensor M.

The complete state consists of the location of the mobile sensor S, which the mobile agent knows, and the location of the leak L, which it strives to discover. Both the mobile sensor and leak locations are assigned randomly at the beginning of each episode; only the former can change within an episode.

At each timestep, the mobile agent can take several types of actions. There are four *move* actions, one corresponding to each of the cardinal directions, that cause the mobile agent to move to an adjacent location unless the edge of the grid lies in that direction, in which case it remains stationary. There is one *sense* action, which employs the mobile sensor in the agent's current

location. There are also n *report* actions, one for each location, which end the episode and cause the mobile agent to issue a report stating that the leak is in that location. If the mobile agent does not report within the first 1000 timesteps, a report action is forced, in order to prevent episodes from continuing indefinitely.

Regardless of the action it selects, the agent observes at each location l_i sensor readings $\epsilon_i = \langle \epsilon_i^C, \epsilon_i^I \rangle$. If at location l_i the mobile agent performs a sense action, its observations at that location are $\epsilon_i = \langle \epsilon_i^C, \epsilon_i^I, \epsilon^M \rangle$. The observations are outcomes of the stochastic process described by the network shown in Figure 1a.

At each timestep that the mobile agent does not issue a report action, it receives a reward of -1, which represents the accumulating human casualties and property damage resulting from the undetected leak. When the mobile agent does report, it receives no reward if the leak is in the reported location. Otherwise, it receives a penalty $p = -1000$. The value of p determines an important trade-off in the optimal behavior of the mobile agent. If the magnitude of p is large, the mobile agent should wait until it is extremely confident about the leak location before reporting. If the magnitude of p is small, it should risk reporting early to avoid the cost of further reducing its uncertainty.

4 Integrating DPNs and RL

This section describes how DPNs can be integrated with RL to perform efficient sensor management in tasks like the chemical leak localization problem. Figure 2 shows an overview of the complete integrated system.

Each location in the environment grid (bottom) contains fixed sensors (open circles) and possibly a mobile sensor (filled circle), which contribute observations to the DPNs (dotted box). The DPNs describe the probabilistic relationships between these observations and intermediate phenomena such as ionization and conductivity and, in turn, the relationship between those intermediate phenomena and the location of the chemical leak. These DPNs produce an updated belief which is used by the RL system to update its policy π. The resulting new policy π' is used to generate an action applied to the mobile sensor. The entire mobile agent (solid box) consists of the RL system and two DPN agents: one describing the mobile sensor and one integrating all the observations into a new belief.

4.1 Implementing DPN Component

The main task of the DPNs is to compute a belief vector $b = P(L|\epsilon) = [P(l_1|\epsilon), \ldots, P(l_n|\epsilon)]$, where $P(l_i|\epsilon)$ denotes the posterior probability of a leak at location l_i, given all available observations ϵ. $P(L|\epsilon)$ must correctly capture correlations between the observations and the leak location. This can be achieved through belief propagation in the causal Bayesian network that describes the processes producing a particular observation set ϵ (see Figure 1a).

However, because this network explicitly represents every sensor and its readings, at each time step we need a different monolithic network to reflect the current sensor constellations and reports. This is clearly impractical. Therefore, we use the DPN approach, where DPN agents collaboratively compute $b = P(L|\epsilon)$ (see Figure 1b). Each DPN agent uses a Bayesian network that captures a subset of variables and relations from the monolithic network, i.e., each DPN agent has partial knowledge of the process depicted in Figure 1a. The DPN agents autonomously configure distributed inference systems by using service discovery. For each location l_i we assume a system of DPN fusion agents processing all available observations ϵ_i obtained within l_i. The DPN agents collaboratively estimate the probability distribution $P(G_i|\epsilon_i)$, where G_i is indicates the presence or absence of the leak at location l_i.

In particular, there are five types of DPN agents. For each location l_i we use agent A_i^G which computes $P(G_i|\epsilon_i)$. In addition, agents A_i^C and A_i^I are used to process raw sensor reports and compute $P(C_i|\epsilon_i^C)$ and $P(I_i|\epsilon_i^I)$, respectively. For example, Figure 2 depicts sets of agents dedicated to locations l_1 and l_2.

Moreover, two DPN agents inside the mobile agent implement separate local inference processes (shown within the solid rectangle in Figure 2). Agent A^M computes $P(M|\epsilon^M)$ using observations from the mobile sensor at location l_i. Also, for all locations A^M stores $P(M|\epsilon^M)$ obtained up to a certain point in time. In this way A^M correctly combines observations collected during different visits to a certain location. Agent A^L computes $P(L|\epsilon)$ by integrating information from all locations:

$$P(L|\epsilon) = cP(L)\prod_{i=1}^{n}\sum_{G_i} P(G_i|L)P(G_i|\epsilon_i),$$

where c is a normalization constant. $P(L)$ describes the prior probability of a leak at different locations while $P(G_i|L)$ is a simple leak dispersion model. Both, $P(L)$ and $P(G_i|L)$ are local domain knowledge of agent A^L. Each distribution $P(G_i|\epsilon_i)$ is provided to A^L after being computed by the set of DPN agents assigned to location l_i and the DPN agents in the mobile agent. This approach supports correct computation of $P(L|\epsilon)$, since each agent A_i^G assumes a uniform prior distribution over G_i and the ratio $P(G_i|\epsilon_i)/P(\epsilon_i|G_i)$ is constant for all states of G_i.

When the mobile agent moves, it updates A^M with its current position. Using this information, A^M joins the DPN agents dedicated to that location. For example, in Figure 2 the helicopter is located in l_2 so A^M joins the network consisting of agents dedicated to l_2.

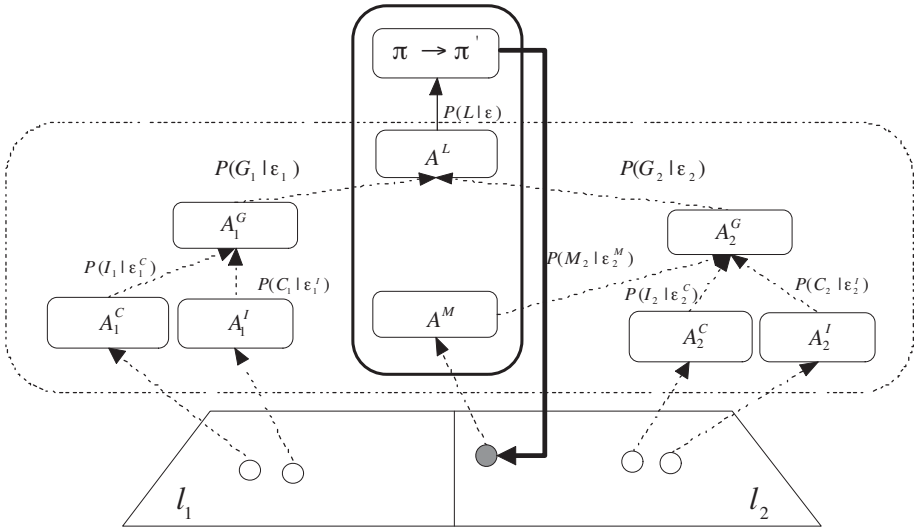

Figure 2: The integrated DPN+RL system. Each location in the environment grid (bottom) contain fixed sensors (open circles) and possibly a mobile sensor (filled circle), which contribute observations to the DPNs (dotted box). The DPNs produce an updated belief used by the RL system to produce a new policy that generates an action applied to the mobile sensor. The mobile agent (solid box) consists of the RL system and a subset of the DPNs.

4.2 Implementing RL Component

DPNs provide an efficient mechanism for maintaining an updated belief about the leak location as new sensor readings are observed. What remains is to determine how the mobile sensor can obtain the most useful observations. We apply RL to this part of the problem because it allows the mobile agent to directly reason about the value of uncertainty reduction.

In this section, we consider the best way to apply RL to a POMDP like the chemical leak localization problem. The traditional approach to solving POMDPs is to find a policy mapping beliefs to actions via off-line planning methods such as value iteration [15]. To perform such planning, complete models of the transition, reward, and observation functions are required. For the chemical leak localization problem, this information is available: the transition and reward functions are simple and deterministic and the observation function is described by the DPNs. However, the computational cost of exact POMDP planning is prohibitive for all but the smallest problems. Many approximate planning methods such as value-directed methods [12] and point-based value iteration [11] are faster but still too expensive for real-world tasks with large state spaces, such as chemical leak localization.

In lieu of planning, it is also possible to learn online in a model-free fashion. Various heuristics such as action voting [13], U-trees [7], and recurrent neural networks [1] use the current belief or portions of the agent's observation history to estimate the current state and then act as in an MDP: with a policy mapping states to actions. These approaches avoid the computational problems of POMDP planning but are only applicable when the current state can be accurately estimated from the history. In chemical leak search, this is not possible because ambiguity about the state is central to the problem itself: if the state could be accurately estimated, the leak location would be known. Thus, a good policy for efficiently finding the leak must be conditioned on beliefs, not on estimates of the current state.

Since accurate state estimation is not possible and off-line planning is intractable, we employ a different approach. We treat the POMDP as a *belief MDP*, i.e. an MDP in which the states are beliefs in the POMDP [3, 5]. In essence, this approach treats belief maintenance as part of the environment, such that the belief update occurs within the state transition function. Since the beliefs are Markov, a policy for the belief MDP can be learned using standard temporal-difference methods such as Q-learning [17]. Note that, although temporal-difference methods are a form of "model-free" RL, this approach still depends critically on the observation function (i.e. the DPNs) to perform the belief updates. The main difficulty is that the beliefs are real-valued so function approximation is required. For simplicity, we employ a linear function approximator that maps beliefs to value estimates.

First, we must select an appropriate representation for the mobile agent's current belief $B = (S, b)$ about the complete state, which consists of the mobile sensor's current location S (known with certainty) and the belief b about L, the leak location. To facilitate function approximation, the mobile sensor's location is represented by a binary vector $S = S_1, \ldots, S_n$, where S_i is one if

the sensor is in l_i and zero otherwise. To represent b, we apply two changes to the belief provided by the DPNs, to make it *egocentric* and to use *entropies* instead of raw probabilities.

In an egocentric representation, the belief is described relative to the mobile agent's current position, to make it easier for function approximation to generalize. This involves reordering the locations in the grid based first on their Manhattan distance from the mobile agent and then clockwise within the same distance. When there are $n = m^2$ locations, the mobile agent's view must include $m - 1$ locations in all directions to ensure it can always see every location in the grid (e.g., when the agent is in the southwest corner, the northern and eastern edges are each $m - 1$ locations away). Hence, this representation requires $(2m - 1)^2$ elements in the belief vector. When the mobile agent is near an edge, some of the locations in its view will not exist; the corresponding vector elements are set to zero.

In addition, the raw probabilities are translated into entropies, such that $b = b_1, \ldots, b_{(2m-1)^2}$, where

$$b_i = -[P(l_i|\epsilon) \log P(l_i|\epsilon) + P(\neg l_i|\epsilon) \log P(\neg l_i|\epsilon)].$$

Using entropies in place of probabilities facilitates linear function approximation because entropy has a monotonic relationship with uncertainty. Intuitively, the mobile agent should behave differently when uncertainty is low. This is difficult to represent linearly given only probabilities, since uncertainty is minimized both when the probability is low and when it is high.

Given such a belief representation, we need a practical approach to handling the action space. Since the mobile agent can report any location as the source of the leak, the action space is large and grows linearly with respect to n. However, it would never be advantageous to report the leak in any location other than that deemed most likely in the current belief. Hence, we define a *meta-report* action, which searches the current belief for the location with the highest probability and executes the corresponding low-level report action. Consequently, the mobile agent needs to reason about only six actions: four movement actions, one sense action, and one meta-report action.

Using this setup, we approximate the action-value function with a separate linear function for each action a. Each such function $Q(B, a)$ is a linear combination of the features in B, parameterized by a vector θ_a. These parameters are updated on-line using Q-learning each time the mobile agent takes an action,

$$\theta_a \leftarrow \theta_a + \alpha[r + \gamma max_{a'} Q(B', a') - Q(B, a)] \Delta_{\theta_a} Q(B, a)$$

where α is a learning rate parameter, γ is a discount factor, and $\Delta_{\theta_a} Q(B, a)$ is the gradient of the value function with respect to the parameters, which in this case is simply B. The mobile agent's policy is derived directly from the action-value function: $\pi(B) = max_a Q(B, a)$. At each timestep, it follows this policy with probability $1 - \beta$; with probability β, it takes a random action to ensure sufficient exploration.[1]

This method of estimating the action-value function is scalable to larger versions of the chemical leak localization problem because the linear function is concise, i.e., the size of each θ_a grows only linearly with respect to the number of locations in the grid. Because we use temporal-difference methods instead of off-line planning, the mobile agent incrementally improves its policy as it interacts with the environment. However, this approach does not *require* the mobile agent to learn on-line in the real world, which would be infeasibly risky in tasks like chemical leak localization. On the contrary, since a complete model is available, experience can be simulated in advance, and the mobile agent deployed only once a good policy is discovered.

5 Results and Discussion

In this section, we report on three sets of experiments designed to investigate and illustrate our approach. In all experiments, the following learning parameters were used: $\beta = 0.05$, $\gamma = 0.99$, and $\alpha = 0.01$. In each run, learning was paused every 1,000 episodes to test the quality of the policy learned so far. In these tests, exploration was turned off ($\beta = 0.0$), the current policy was evaluated for 100 episodes, and the average reward per episode was computed. All graphs below show this testing performance, averaged over 10 independent runs and smoothed using a uniform moving average with a window of 10 testing intervals (10,000 learning episodes) for the first two experiments and 50 testing intervals (50,000 learning episodes) for the third.

The first experiment investigates the feasibility of our approach by testing the integrated DPN+RL system in the chemical leak localization problem where $n = 25$ (a 5×5 grid). To confirm that learning is successful, performance is compared to a random baseline policy that chooses actions with equal probability. Despite an absence of learning, this baseline performs reasonably well. Since the fixed sensors provide useful, though noisy, information about the leak location, the random policy can simply wait for the DPNs to slowly reduce uncertainty about the leak location, even without intelligently using the mobile sensor.

Figure 3 shows that the DPN+RL system learns to substantially outperform this baseline, cutting the average negative reward in half. Since the system relies on linear function approximation, the final learned policies are not optimal. However, they display intuitively correct behavior, exploring the grid, sensing, and reporting when uncertainty becomes low. The DPN+RL system performs poorly in early episodes, which is not

[1] Typically named ϵ, e.g., ϵ-greedy exploration, but here named β to avoid confusion with the observations ϵ.

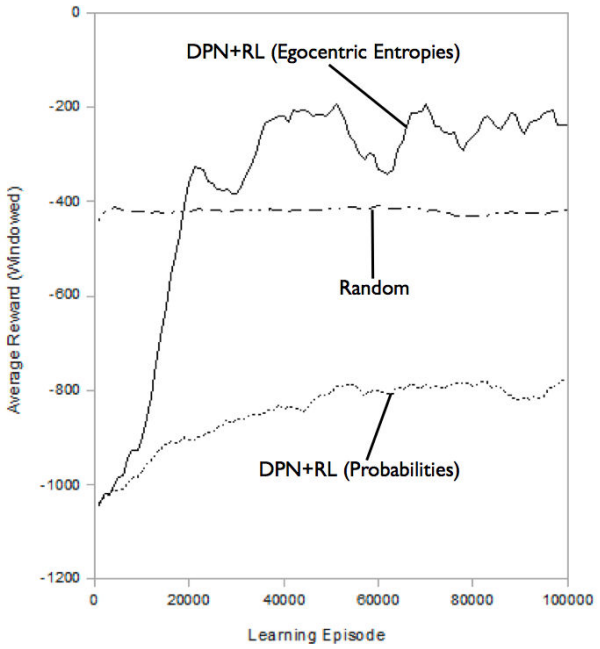

Figure 3: The average performance over time of the DPN+RL system using two different belief representations, compared to a random baseline policy.

surprising since initial learning can easily lead to policies that are worse than random, e.g., ones that report immediately. Since off-line training is feasible, these early episodes incur only simulation costs, not the actual costs of incorrect leak localization in the real world.

This figure also compares performance to a version of the DPN+RL system that does not use the belief representation based on egocentric entropies described in Section 4.2, but instead simply uses the raw probabilities supplied by the DPNs. The importance of using the egocentric entropy representation to facilitate linear function approximation is confirmed by these results, which show that the alternative representation performs dramatically worse, failing even to match the baseline method. Informal experiments run for more episodes suggest that the performance of this representation does not improve even given more time.

The second experiment investigates the consequences of using an incorrect observation model to perform inference. Instead of using a correct tree structure for the computation of $P(G_i|\epsilon_i)$, the system relied on a naïve Bayesian model that incorrectly assumes all observations ϵ_i are independent given G_i. In addition, the conditional probability tables relating G_i and ϵ_i assume lower noise than the true model. Figure 4 shows the results of this experiment, again on a 5×5 grid, which compares the performance of the system using a correct model (also shown in Figure 3) to one using an incorrect model.

This experiment clearly demonstrates that using a correct model is essential for learning a good control policy. It also gives some insight into how much performance benefit the DPNs can offer, as the system with an incorrect model can be said to approximate the case where no DPNs are used. Since DPNs allow distributed sensors to correctly combine their own partial models, they can increase the quality of the resulting inference. In the absence of DPNs, the mobile agent will have to use a monolithic network. Since the cost of maintaining and performing inference on such a network may be too great for a computationally constrained mobile agent, it will likely be forced to rely on approximations, the effects of which are underlined by this experiment.

The third experiment investigates scalability by testing the integrated DPN+RL system on a larger (10×10) grid with $n = 100$. We compared the performance of the DPN+RL system with a correct model and the egocentric entropy belief representation to the random baseline method. Figure 5 shows the results of this experiment. As before, the DPN+RL method learns to clearly outperform the baseline method. Learning is slower than in the smaller grids, but not by much considering that the grid is four times larger and the state space is 16 times larger. Hence, this result suggests that the generalization capabilities of the linear function approximator contribute to the system's scalability. Nonetheless, the improvement over the baseline method is smaller than in the 5×5 grid, which indicates that finding good linear approximations becomes more difficult in larger versions of the problem and that coupling our approach with more sophisticated, nonlinear function approximators could result in further performance improvements and additional scalability.

Overall these results validate the potential of DPN+RL as a tool for sensor management. The integration of these two technologies results in several important advantages. Because inference is distributed, the computational costs are not imposed on a single agent. Making inference tractable is important at two stages: when searching for a policy and when using a policy. Regardless of what approach is used to find a policy, doing so requires performing many belief updates. Once a policy is found, it can be used only if the mobile agent knows its current belief, which requires continual on-line belief updates. At both of these stages, the DPNs make the necessary inference tractable by distributing the work among many agents. The DPNs also enable the agents to distribute *knowledge*. When it is not feasible for the mobile sensor to maintain correct models of every sensor in the system, it can still perform correct inference since the DPNs allow each sensor to contribute correct partial models. The importance of using correct models for inference is demonstrated in our second experiment.

The use of RL to find a control policy for the mobile sensor allows the mobile agent to do more than just re-

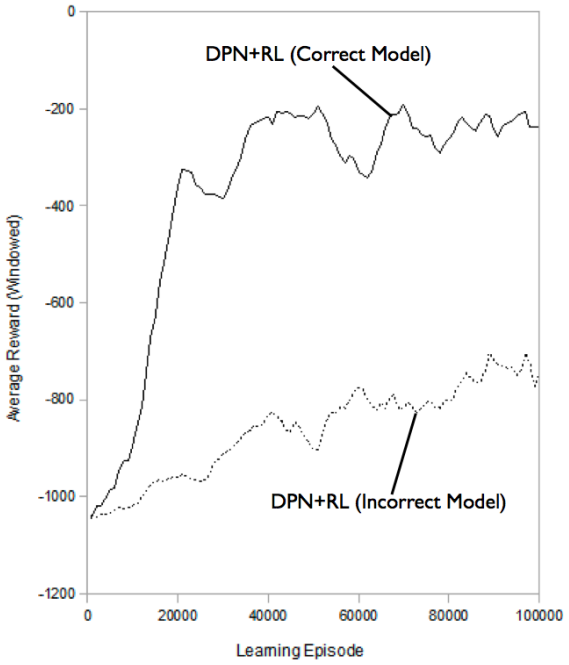

Figure 4: The average performance over time of the DPN+RL system using a correct observation model compared to one with incorrect structure and conditional probability tables.

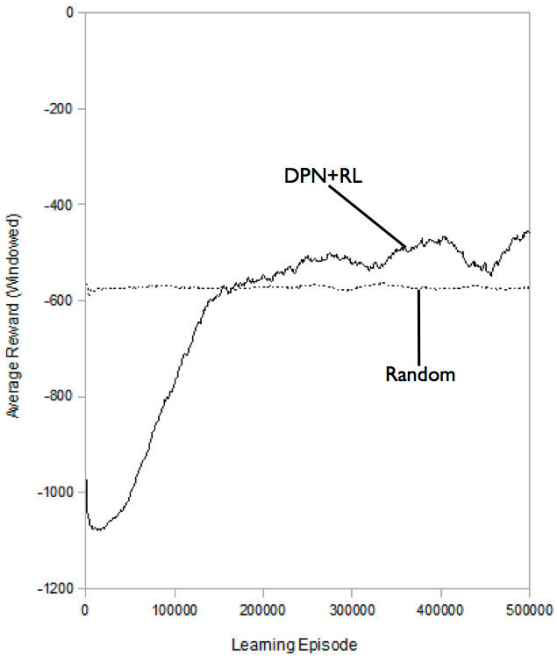

Figure 5: The average performance over time of the DPN+RL system on a large (10×10) grid, compared to a random baseline policy.

duce uncertainty. It allows it reason about the value of uncertainty reduction and discover an effective balance between it and other goals. Furthermore, since RL discovers a complete policy mapping beliefs to actions, it avoids the expense of computing at runtime the expected reduction in uncertainty of particular action sequences. By applying temporal-difference methods with linear function approximation, the intractability of POMDP planning is also avoided. As a result, the system is scalable to larger grid sizes, as confirmed by our third experiment.

We believe this method can be useful in a wide range of sensor management tasks. Many such tasks involve mobile sensors that require control policies. More broadly, any task containing sensors that are controllable, even if their location is fixed, could be appropriate. For example, some sensors may have multiple settings that trade-off sensitivity and specificity and in some tasks, it may be useful to determine when to turn some sensors off. In all these cases, sensor management could benefit from a principled approach to finding control policies for such sensors.

6 Conclusions

This paper presents an approach to sensor management that integrates DPNs and RL. DPNs are used to perform efficient, distributed inference and RL is used to automatically discover control policies for active sensors. The resulting method is evaluated on a simulation of a chemical leak localization task. The results demonstrate 1) that the integrated approach can learn policies that perform effective sensor management, 2) that inference based on a correct observation model, which the DPNs make feasible, is critical to performance, and 3) that the system scales well to larger versions of the task.

The proposed method could be extended to handle multiple mobile sensors which would require learning separate control policies for each sensor via a multiagent reinforcement learning method. Moreover, our function approximation scheme could be extended to nonlinear representations such as neural networks. This may further improve system performance and allow additional scalability to larger grids. We would also like to systematically investigate the impact various modeling errors have on the learner's performance. While our results show that correct models are critical, we have not yet determined the importance of errors in model structure relative to errors in the conditional probability tables.

7 Acknowledgments

The reported research is part of the Interactive Collaborative Information Systems (ICIS) project (www.icis.decis.nl), supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

References

[1] Bram Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems 14*, 2002.

[2] P. de Oude, G. Pavlin, and T. Hood. A modular approach to adaptive Bayesian information fusion. In *the 10th International Conference on Information Fusion*, 2007.

[3] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[4] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[5] T. Karadeniz and L. Akin. FDMS with Q-learning: A neuro-fuzzy approach to partially observable markov decision problems. *International Journal of Advanced Robotic Systems*, 1(3):251–262, 2004.

[6] Paskin Mark, Guestrin Carlos, and Jim McFadden. A robust architecture for inference in sensor networks. In *Fourth International Conference on Information Processing in Sensor Networks*, Los Angelos, California, 2005.

[7] Andrew R. McCallum. Instance-based utile distinctions for reinforcement learning. In *Proceedings of the Twelfth International Machine Learning Conference*, pages 387–395, 1995.

[8] H.M. Mitchel. *Sensor Management*. Springer, 2007.

[9] G. Pavlin, P. de Oude, M. Maris, J. Nunnink, and T. Hood. A multi agent systems approach to distributed Bayesian information fusion. *International Journal on Information Fusion*, 2008. To appear.

[10] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, 1988.

[11] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 2003.

[12] P. Poupart and C. Boutilier. Value-directed belief state approximation for POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 279–288, 2000.

[13] Reid Simmons and Sven Koenig. Probabilistic navigation in partially observable environments. In *Fourteenth International Joint Conference on Artifcial Intelligence*, pages 1080–1087, 1995.

[14] C. Simonin, J.-P. Le Cadre, and F. Dambreville. The cross-entropy method for solving a variety of hierarchical search problems. *10th International Conference on Information Fusion*, pages 1–8, 2007.

[15] R. W. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[16] Matthijs T. J. Spaan. Cooperative active perception using POMDPs. In *AAAI 2008 Workshop on Advancements in POMDP Solvers*, 2008.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachussets, 1998.

[18] Yang Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.