Multi-Agent Systems on Sensor Networks: A Distributed Reinforcement Learning Approach

Chen-Khong Tham [1], Jean-Christophe Renaud [2]
Dept of Electrical & Computer Engineering, National University of Singapore
[1] *eletck@nus.edu.sg* [2] *jeanchristophe.renaud@nus.edu.sg*

Abstract

Implementing a multi-agent system (MAS) on a wireless sensor network comprising sensor-actuator nodes with processing capability enables these nodes to perform tasks in a coordinated manner to achieve some desired system-wide objective. In this paper, several distributed reinforcement learning (DRL) algorithms used in MAS are described. Next, we present our experience and results from the implementation of these DRL algorithms on actual Berkeley motes in terms of communication, computation and energy costs, and speed of convergence to optimal policies. We investigate whether globally optimal or merely locally optimal policies are achieved. Finally, we discuss the trade-offs that are necessary when employing DRL algorithms for coordinated decision-making tasks in resource-constrained wireless sensor networks.

1. INTRODUCTION

Reinforcement Learning (RL) ([1], [2]) is usually defined as the problem faced by a learner of how to determine correct behavior through trial-and-error interactions with a dynamic environment in order to achieve a goal. In the standard RL model, the learner and decision-maker is called an *agent* and is connected to its environment via perception or sensing, and action, as shown in Figure 1.

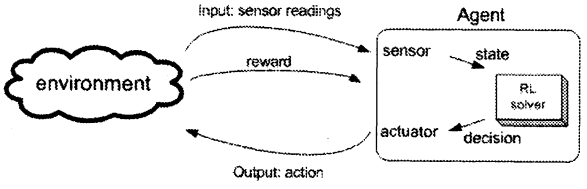

Fig. 1: Abstract view of an agent in its environment in RL

More specifically, the agent and environment interact at each of a sequence of discrete time steps t. At each step of the interaction, the agent senses some information about its environment (input), determines the world state and then, on that basis chooses and takes an action (output). The action changes the state of the environment and of the agent. One time step later, the value of the state transition following that action is given to the agent by the environment as a scalar called *reward*. The agent should behave so as to maximize the received rewards, or more particularly, a long-term sum of rewards.

Let s_t be the state of the system at time t and assume that the learning agent chooses action a_t, leading to two consequences. First, the agent receives a reward r_{t+1} from the environment at time $t+1$. Second, the system state changes to a new state s_{t+1}.

There are several ways to define the objective of the learning agent, but all of them attempt to maximize the amount of reward the agent receives over time. In this paper, we consider the case of the agent learning how to determine the actions maximizing the *discounted expected return* which is a discounted sum of rewards over time: $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where γ is a *discount factor* in [0..1] used to weight near term rewards more heavily than distant future rewards. We chose the discounted reward since it is appropriate for continuing tasks, in which the interaction with the environment continues without limit in time.

A. Distributed Reinforcement Learning (DRL) and Multi-Agent Systems (MAS)

We now turn our attention to multi-agent systems (MAS) made up of a number of agents. Following the taxonomy of MAS presented by Stone and Veloso [3], the MAS domain can be divided along two dimensions: (i) degree of heterogeneity of the agents, and (ii) degree of communication involved. This paper considers two main combinations of heterogeneity and communication: homogeneous non-communicating agents (Section 2-B) and homogeneous communicating agents (Sections 2-C and 2-D). We focus on cooperative MAS where agents share the same common goal.

From a particular agent's point of view, multi-agent systems differ from single-agent ones most significantly in that the environment dynamics can be influenced by other agents. In addition to the uncertainty (i.e. stochasticity) that may be inherent in the environment, other agents can affect the environment in unpredictable ways due to their actions. However, the full power and advantage of a MAS can be realized when the ability for agents to communicate with one another is added, enabling learning to be accelerated, more information to be gathered about the world state, and experiences of other agents to be shared. Different methods can be designed depending on the kind of information that is communicated, e.g. sensory input, local states, choice of action etc.

0-7803-9399-6/05/$20.00 © 2005 IEEE

B. Wireless Sensors Networks (WSNs) and MAS

Wireless sensor networks is a recent development arising from advances in wireless communications, microelectronics and miniaturized sensors. The most general concept is to have a large number of wireless sensor nodes spread out in an environment for monitoring or tracking purposes. Most work on sensor networks focus on methods to relay sensed information in an energy-efficient manner to a command center. In addition, methods for collaborative signal and information processing (CSIP) [4] which attempt to perform processing in a distributed and collaborative manner among several sensor nodes have also been proposed.

Although there has been related work which adopt a multi-agent perspective to sensor networks [5] and reinforcement learning is a common technique employed in multi-agent systems [6], [7], [8], our work is novel since it is the first study and implementation of cooperative and coordinated reinforcement learning algorithms in an actual sensor network. The processing and communication capabilities of sensor nodes enable them to make decisions and perform tasks in a coordinated manner to achieve some desired system-wide or global objective. We also also consider the more general case of sensor nodes which can actuate and cause changes to the environment they operate in, i.e. *sensor-actuator nodes.*

A distributed approach to decision-making using WSNs is attractive for several reasons. First, sensing entities are usually spatially distributed, thus forming distributed systems for which a decentralized approach is more natural. Second, sensor networks can be very large, i.e. containing hundreds or thousands of nodes; thus, a distributed approach would always be more scalable than a centralized one. Finally, a distributed approach is compatible with the resource-constrained nature of sensor nodes. Actual commercialized motes such as Crossbow Mica2 motes [9] are small devices with limited memory and computational capabilities and are energy constrained since they are battery-powered. Therefore, a distributed approach to performing computation, i.e. using distributed algorithms, and limiting the amount and distance of communication are necessary design parameters in order to achieve an efficient, energy-aware and scalable solution. Furthermore, the restricted communication bandwidth and range in WSNs would exclude a centralized approach.

In a distributed learning and decision-making system, the system behavior is influenced by the whole team of simultaneously and independently acting agents. Thus, the dynamics of the environment are likely to change more frequently than in the single agent case. As a learning method that does not need any prior model of the environment and can perform online learning, RL is well-suited for cooperative MAS, where agents have little information about each other. RL is also a robust and natural method for agents to learn how to coordinate their action choices.

In this paper, we focus on distributed RL algorithms for cooperative decision-making implemented on actual wireless sensor nodes ('motes') and study how these algorithms behave in real environmental conditions.

2. Distributed Reinforcement Learning (DRL) algorithms

In [6], Schneider et al proposed an algorithm for DRL based on distributing the representation of the value function among the agents. Cooperative decision-making in the MAS is achieved by the exchange of the value of the states each agent lands in with its neighbors. Lauer and Riedmiller [7] also studied a distributed approach to RL in a cooperative MAS of independent and autonomous agents. Their main idea is to reduce the action space information needed by the agents to make a decision by projecting it to smaller action spaces. These two techniques appear promising for our distributed RL in WSN problem, and we study them further in this paper (see Sections 2-C and 2-D).

In the next few sections, we describe the background theory and algorithms for three DRL approaches that are studied in detail in this paper.

A. Background concepts

1) General framework: Markov Decision Processes (MDP): A RL problem that satisfies the Markov property, i.e. future outcomes are based only on the current state, is called a *Markov Decision Process*, or MDP. In this paper, we focus on finite state discrete MDPs for which the state and action spaces are both discrete and finite.

Definition 1: A Markov Decision Process (MDP) (as defined in [1]) is a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where:

- $\mathcal{S}$ is a discrete set of states;
- $\mathcal{A}$ is a discrete set of actions available in each state;
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a mapping from the state-action space to a probability distribution over the state space. A function of $\mathcal{P}$ is called a transition probability and is denoted $P^a_{ss'} = Prob(s_{t+1} = s' | s_t = s, a_t = a)$;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a mapping of the state-action space which returns the reward of taking a particular action in a given state. $R^a_{ss'} = E[r_{t+1} | s_{t+1} = s', s_t = s, a_t = a]$.

2) Value functions: A *policy* π is defined as a rule by which the agent selects its action as a function of states. It is therefore a mapping from each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ to the probability of taking action a when in state s, i.e: $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0..1]$. Moreover, the value of a state s under a policy π, denoted $V^\pi(s)$, is the expected return the agent can receive when in state s and following policy π thereafter. This function, which is a mapping from the state space is called the *state-value function* for policy π. It is formally defined by:

$$V^\pi(s, a) = E^\pi\{R_t | s_t = s\}$$
$$= E^\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

Similarly, the value of taking action a in state s under a policy π, denoted $Q^\pi(s, a)$, is the return that the agent

can expect while starting in state s, taking the action a, and following policy π thereafter:

$$Q^{\pi}(s,a) = E^{\pi}\{R_t | s_t = s, a_t = a\}$$
$$= E^{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

The quantity Q^{π} is called the *action-value function* for policy π. The objective of the learning task can be expressed with Q^{π} in the following terms: find a policy π^* such that the expected value of the return is maximized, i.e find π^* such that:

$$Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \ \forall \ (s,a) \in \mathcal{S} \times \mathcal{A}$$

3) Q-Learning algorithm: Q-learning [10] is an algorithm developed from the theory of dynamic programming for delayed RL that does not need a model of the environment and can be used on-line. In Q-learning, the action values, Q^{π}, are represented by a two-dimensional lookup table indexed by the state-action pairs.

The update rule at time step $t+1$ of the Q-learning algorithm is given by:

$$Q_{t+1}(s_t,a_t) = (1-\alpha)Q_t(s_t,a_t) + \alpha\Big(r_{t+1}(s_{t+1}) + \gamma \max_{a \in A} Q_t(s_{t+1},a)\Big) \quad (1)$$

where γ is the discount factor mentioned earlier, where $\gamma \in [0..1]$, and $\alpha \in [0..1]$ is a learning rate parameter.

With this background, we now proceed to consider three DRL approaches for performing cooperative decision-making in wireless sensor networks.

B. Fully distributed Q-Learning (IndLearners)

This scenario is the simplest multi-agent case which involves homogeneous non-communicating agents: all the agents have the same internal structure such as states, available actions, goals and reward functions. Agents rely on their own sensor readings to perceive the environment they operate in, and decide on their own actions using the Q-learning algorithm shown in equation 1 above. We shall refer to this scheme as the *IndLearners* scheme.

In recent years, several extensions to RL and Q-learning for distributed systems have been proposed. The next two subsections present two of them: Distributed Value Function DRL [6] which we shall refer to as *DVF* in short, and another DRL algorithm proposed by Lauer and Riedmiller in [7], which we shall denote as *OptDRL*.

C. Distributed Value Function (DVF) DRL

1) Distributed approaches to RL: In [6], Schneider et al tackled the problem of finding a distributed solution for RL in a MAS. They proposed several methods according to the type of information that is distributed. These are:

- *Global Reward DRL*: For this method, agents sense, act, and learn locally. However, their decisions are based on a global reward function. This implies communication between the agents to spread the global reward among the system, or a broadcast of global reward;
- *Local DRL*: This method is fully distributed since each agent acts independently. The MAS is therefore the concatenation of several single agent systems. This case corresponds the *IndLearners* case described above. The major drawback of this method is that the composite action of all the independent agents is not guaranteed to be optimal;
- *Distributed Reward DRL*: Nodes communicate and exchange information about the immediate rewards they receive from the environment. Then, each node considers a weighted average of its local reward and those of its neighbors. In this scheme, the nodes try to optimize their behavior and those of their direct neighbors;
- *Distributed Value Function (DVF) DRL*: In this method, nodes communicate and exchange information about their value functions. In this case, nodes cooperate not only with their direct neighbors but with all the nodes of the MAS since the value function captures information about other nodes which are not direct neighbors as well. Moreover, by choosing the weighting functions well, the sum of the value functions over all the nodes can be equal to an expected future weighted sum of rewards over all the nodes. This is similar to the global reward DRL approach but agents achieve this behavior with access to only local rewards and communication of value function information only between neighbors.

2) Distributed Value Function (DVF) algorithm: Usually, in MAS, agents only have local state information since the global state of the system is not fully observable from each agent's point of view. Hence, Schneider at al proposed a Q-learning based algorithm for the DVF DRL case which allows each node to compute its local value function based only on available local information.

According to this version of the DVF algorithm, agents only need to transmit the estimated value of the current state they land in, i.e. $V^i(s_t^i)$ for agent i at time t at each iteration.

The update rule at time step t for agent i are given by:

$$Q_{t+1}^i(s_t^i,a_t^i) = (1-\alpha)Q_t^i(s_t^i,a_t^i) + \alpha\Big(r_{t+1}^i(s_{t+1}^i) + \gamma \sum_{j \in Neigh(i)} f^i(j) V_t^j(s_{t+1}^j)\Big) \quad (2)$$

$$V_{t+1}^i(s_t^i) = \max_{a \in A^i} Q_{t+1}^i(s_t^i,a) \quad (3)$$

where $f^i(j)$ are factors that weight the value functions of the neighbors of agent i in the computation of its own value function. Several weighting factors are possible. In our implementation, we defined them to be the simple case of:

$$f^i(j) = \begin{cases} \frac{1}{|Neigh(i)|}, & \text{if } Neigh(i) \neq 0; \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

where $j \in Neigh(i)$, the set of neighbor nodes.

D. Optimistic DRL (OptDRL)

1) General framework: Multi-agent MDP: For the analysis of this distributed decision-making method based on RL, a general framework which is close to the basic MDP case described above (Definition 1) is used [7]. This framework extends MDPs to the multi-agent case by taking into consideration the state and action of every agent. A state and an action therefore becomes a vector composed of all the elementary states and actions of the different agents of the system. In this paper, we assume that the system is a collection of m agents that all have the same state and action spaces. There is a close relationship between the single-agent MDP and the multi-agent extension, defined as follows:

Definition 2: A cooperative Multi-Agent Markov Decision Process (MAMDP) of m agents is a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where $\mathcal{S} = \prod_{i=1}^{m} S^i$ is the global state of the system, $\mathcal{A} = \prod_{i=1}^{m} A^i$ the global action space, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ a probability distribution function over the state space and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function.

2) Optimistic DRL (OptDRL) algorithm: The algorithm presented by Lauer and Riedmiller in [7] focuses on distributed RL for cooperative MAS. In this cooperative MAS formulation, agents are given the same reward function.

The challenge of incomplete information with respect to the choice of action was tackled in [7]. Every independent learning agent only knows its local part of the action-vector (its own local action) and cannot distinguish between several action-vectors which have the same local action for the agent. Therefore, the basic idea is to project the information of the whole system (considered as a single-agent MDP) into smaller tables available at each agent which depend only on local actions. However, projecting this information leads to some loss of information and there is no more guarantee that the policies computed based in the distributed manner are optimal. An additional mechanism of coordination between the agents is then designed to overcome this problem.

The update rule at time step $t+1$ for agent i is given by:

$$q^i_{t+1}(S_t, a^i_t) = \max\{q^i_t(S_t, a^i_t), (1-\alpha)q^i_t(S_t, a^i_t) + \alpha\big(r^i_{t+1}(S_{t+1}) + \gamma \max_{a \in A^i} q^i_t(S_{t+1}, a)\big)\} \quad (5)$$

$$\Pi^i_{t+1}(S_t) = a^i_t \text{ iff } \max_{a \in A^i} q^i_t(S_t, a) \neq \max_{a \in A^i} q^i_{t+1}(S_t, a) \quad (6)$$

The central idea of equation (5) is that each agent chooses its action assuming the other agents will act optimally. Equation (6) specifies that the policy is updated only when there is an improvement in the $q^i(\cdot)$-values, thus introducing coordination between the agents.

Lauer and Riedmiller prove that the *OptDRL* algorithm finds optimal policies in deterministic environments. However, these results are not applicable in stochastic environments, since, for such environments, it is difficult to differentiate between the behavior of other agents in the successor state from the random behavior of the environment itself.

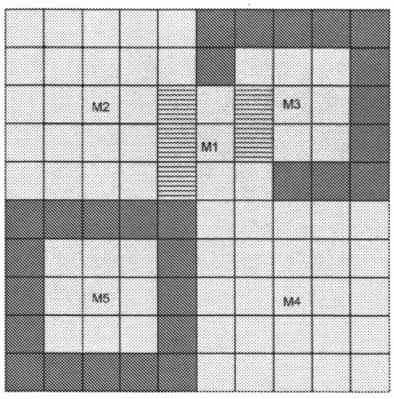

Fig. 2: Representation of the room environment using a 10x10 grid. Grey cells are not illuminated, yellow cells are illuminated by one mote and striped cells are illuminated by two motes.

3. Implementation on Berkeley motes and TOSSIM simulations

In order to study and compare the performance of the *IndLearners, DVF and OptDRL* algorithms, we implemented them on actual Crossbow Mica2 sensor motes [9]) and used the TOSSIM [11] simulation platform in order to evaluate parameters of interest such as energy consumption. TOSSIM is a discrete event simulator for TinyOS [12] sensor networks that builds directly from the same TinyOS code written for the actual motes. Energy measurements are obtained with PowerTOSSIM [13], which is an extension to TOSSIM that provides an accurate per node estimate of power consumption of several mote components such as the radio and CPU.

A. Operating environment

The algorithms have been implemented for a lighting control application of a room represented by a 10×10 grid as shown in Figure 2). This room contains a group of five agents on five motes with light sensing capabilities, labeled from M1 to M5. Each of them also has an embedded light source that it can actuate to illuminate the part of the room surrounding the agent. The objective is for the agents to learn to cooperate with one another in order to completely illuminate the room in an energy-efficient way, i.e. minimize the number of lights turned on. We considered a deterministic environment with deterministic state transitions and rewards.

B. State-action spaces

The area of an agent i, denoted as $\mathcal{A}^i$, refers to the 5×5 grid square centered on the agent i.

1) Local agent states: Each mote i senses the light level in its area. Its local state s^i is the concatenation of the light readings (bright or dim) of the 25 cells. Therefore, there are 2^{25} possible states for each agent (see Table 2).

2) Local agent actions: Each mote has the ability to take one of the following three actions in any state it lands in. The action space A^i is:

- Action 0: Turn OFF the light;

- Action 1: Turn on the light in LOW mode. This mode illuminates nine cells around the agent, as shown by M1, M3 and M5 in Figure 2;
- Action 2: turn on the light in HIGH mode. This mode illuminates the 25 cells around the agent, as shown by M2 and M4 in Figure 2.

3) Global state of MAS: The global state of the five-agent MAS, denoted by S, is the concatenation of the light level of all the cells in the room. There are 2^{100} possible global states.

C. Reward functions

1) IndLearners and DVF algorithms: For these algorithms, the agents use only information that is locally available to make their decisions. The reward for agent i, denoted as r^i, is a function of its state s^i and is defined by:

$$r^i(s^i) = G^i(s^i) - C^i \tag{7}$$

where:

- $G^i(s^i)$ is a function of the number of cells illuminated in the area of agent i. We used $G^i(s^i) = nb\_cells\_bright(\mathcal{A}^i) \times GAIN\_CELL\_BRIGHT$;
- C^i is a function of the energy consumption resulting from the previous action of agent i. We used:

$$C^i = \begin{cases} 0, & \text{if action=0;} \\ COST\_LIGHT\_LOW, & \text{if action=1;} \\ COST\_LIGHT\_HIGH, & \text{if action=2.} \end{cases}$$

2) OptDRL algorithm: This algorithm requires the global state of the environment as well as a global reward function in order to force cooperation between the agents. The global reward is defined as:

$$Glob\_rew(S) = G(S) - C(S) \tag{8}$$

where:

- $G(S)$ is a function of the number of cells illuminated in the whole room. $G(S) = nb\_cells\_bright\_room \times GAIN\_CELL\_BRIGHT$;
- $C(S)$ is a function of the energy consumption resulting from the previous actions of all the agents, i.e. modes of all the light sources.

The values of the parameters $GAIN\_CELL\_BRIGHT$, $COST\_LIGHT\_LOW$ and $COST\_LIGHT\_HIGH$ were chosen such that the optimal policy is as follows: agents M2, M3, M4 and M5 turn on their lights in HIGH mode, and M1 turns OFF its light.

D. Results

1) Convergence of the algorithms: Figures 3, 4, 5 show the convergence of the Q-values for the three actions in the state "All Bright" with respect to the number of learning iterations for agents running the *IndLearners*, *DVF* and *OptDRL* algorithms, respectively.

The figures show that this state is stable and that the MAS is able to learn the optimal policy in the cases of the *DVF* and the *OptDRL* algorithms. However, the final policy learned in the case of *IndLearners* where there is no cooperation between agents is for all agents to turn on their lights in the HIGH mode, thus incurring more energy than the case of the optimal policy.

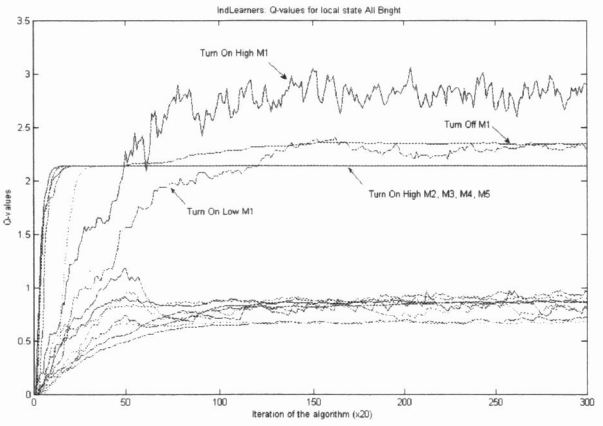

Fig. 3: Convergence of the Q-values for agents running the *IndLearners* algorithm

Fig. 4: Convergence of the Q-values for agents running the *DVF* algorithm

Finally, the *OptDRL* agents learn the optimal policy much faster than *DVF* agents, i.e. 1,200 iterations for *OptRL* compared to 4,400 for *DVF*). However, we note that the environment in this study is deterministic. It is expected that, for general stochastic environments, the *OptDRL* algorithm will perform worse than what is shown here because it is unable to distinguish between random noise and the behavior of other agents.

2) Energy and memory considerations: Table 1 compares the energy spent on communication and computation by the entire MAS comprising the five agents on motes during the learning phase (first 4,000 iterations) for the different algorithms described above. The application-level performance metrics, i.e. illuminating the entire room with the fewest number of lights and the lowest setting, are also shown, together with the cumulative rewards received.

Although the *OptDRL* algorithm converges quickly to the

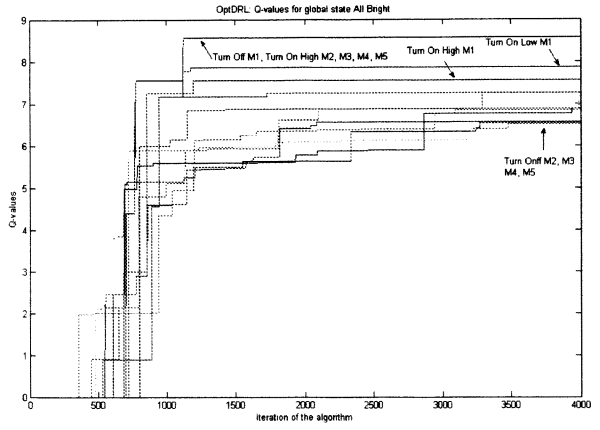

Fig. 5: Convergence of the Q-values for agents running the *OptDRL* algorithm

TABLE 1: ENERGY CONSUMPTION (J) AND APPLICATION-LEVEL PERFORMANCE OF THE MAS WITH 5 MOTES DURING THE FIRST 4000 ITERATIONS

	Ind Learners	*DVF*	*OptDRL*
Communication	0	1648.4	1681.0
Computation	954.9	971.7	1187.8
Lights LOW	2404	3014	1393
Lights HIGH	12721	12192	12123
Cells Bright	318323	317674	321766
Cumulative Reward	17429	17466	19078

optimal policy as shown above, and it performs the best in application-level performance metrics, i.e. highest degree of illumination at the lowest light settings, as well as receiving the largest cumulative reward, it also consumes the most energy for communication and computation since global state-related information needs to be exchanged. From Table 2, we can also see that *OptDRL* requires significantly more memory compared to the other algorithms. In our formulation, the size of the global state and memory requirements increase exponentially with the size of the environment under consideration, and linearly with the number of actions of each agent and the number of agents. In our experiments, we have assumed that the global state can be broadcast to all agents and all are within radio range. This means that the communication cost for *OptDRL* will only increase exponentially with the size of the environment under consideration. If the global state needs to be relayed in a hop-by-hop manner, the amount of communication is likely to increase further and be dependent also on the number of agents.

The *DVF* agents have more modest energy requirements for communication and since they only communicate their V-values with their neighbors. Furthermore, the memory requirement of the algorithm is significantly less than the case of *OptDRL* since only local state is constructed. Since the *DVF* algorithm can achieve coordinated behavior and find the optimal policy, it seems to be an appropriate choice for performing cooperative distributed decision-making on resource-constrained wireless sensor networks.

TABLE 2: MEMORY REQUIREMENTS OF THE ALGORITHMS

	Expression	Actual values
IndLearners	$\|s^i\| \times \|A^i\|$	$2^{25} \times 3$
DVF	$\|s^i\| \times \|A^i\| + \|s^i\|$	$2^{25} \times 4$
OptDRL	$\|S\| \times \|A^i\| + \|S\|$	$2^{100} \times 4$

4. CONCLUSION AND FUTURE WORK

A. Related work

In [8], Guestrin et al present several new algorithms for multi-agent RL. The common feature of these algorithms is a parameterized, structured representation of a policy or value function. Their approach is to approximate the joint value function as a linear combination of local value functions, each of which relates only to the parts of the system controlled by a smaller number of agents. Agents then use an efficient linear programming algorithm to derive a rule-based value function which is an approximation to the optimal joint value function. Given this value function, the agents then apply a *coordination graph* algorithm at each iteration of the process to determine a joint action. This coordination structure can change in different states and may provide a well-founded schema for other MAS coordination and communication approaches. However, this approach requires extensive message passing and may be incompatible with WSNs where we try to minimize communication between agents for energy savings.

Besides the theoretical studies, several applications of distributed RL have been proposed. In [14], Ferreira and Khosla used the Distributed Value Function algorithm [6] to reach collaboration in a MAS and applied it to two different distributed applications: a mobile robot planning and searching task, and an intelligent traffic system in an urban environment. However, no comparison with other existing distributed algorithms was provided. Packet routing is also a domain for which distributed RL algorithms have been designed. For example, in [15], Littman and Boyan described the routing task as a RL problem and proposed a self-adjusting RL-based algorithm based only on local information. Although their empirical studies and simulations are promising, they are not realistic from the point of view of actual computer networks.

B. Future work

Our work has highlighted several challenges that still need to be tackled. First, the choice of the weighting factors in the *DVF* algorithm influences its overall performance as the authors suggested themselves [6]. Additional weighting functions need to be studied. These should take into consideration the major constraints of WSNs. For example, the choice of the neighboring nodes an agent exchanges its V-value with can be decided based on communication cost parameters, i.e. an agent can decide to transmit its V-value to neighbors only when the communication cost incurred is less than the expected gain obtained by the exchange of the V-values. Second, in a MAS, a particular agent cannot observe directly the local state of other agents; instead, an agent can use communication in order to share this information and determine the global

state of the MAS needed in the *OptDRL* algorithm. However, communication incurs a cost in terms of bandwidth and energy consumption, which are two scarce resources in WSNs. Methods using partial observability can be applied to enhance the studied algorithms. An agent should also decide if it is worthwhile to perform a communication action at every iteration. Thus, an agent's policy may need to include agent communication decisions in addition to action decisions. We plan to study these aspects in greater detail in our future work.

Furthermore, we plan to deploy these studied algorithms on sensor networks which form part of larger global architecture which we term *SensorGrid* [16], [17]. This effort focusses on integrating sensor and actuator networks, in particular, to the Grid in order to perform real time decision-making on a large scale. It uses a hierarchical architecture relying on sensor nodes at the lower level, cluster heads and finally the Grid at the upper level, comprising Grid clients and Grid servers. All these entities are decision-makers at various levels, and both network and application aspects of quality of service (QoS) such as delay, accuracy of sensing, reliability of the response, optimality of the decisions etc. need to be taken into consideration.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[2] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[3] P. Stone and M. M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[4] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information-directed approach," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1199–1209, August 2003.

[5] V. Lesser, J. Charles L. Ortiz, and M. Tambe, *Distributed Sensor Networks: A Multiagent Perspective*. Netherlands: Kluwer Academic Publishers, 2003.

[6] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," in *Proc. 16th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1999, pp. 371–378.

[7] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 535–542.

[8] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 227–234.

[9] Crossbow Technology Inc.: Motes, smart dust sensors, wireless sensor networks. [Online]. Available: http://www.xbow.com/Products/productsdetails.aspx?sid=3

[10] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, UK, 1989.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 2003.

[12] TinyOS. [Online]. Available: http://www.tinyos.net/

[13] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM Press, 2004, pp. 188–200.

[14] E. Ferreira and P. Khosla, "Multi agent collaboration using distributed value functions," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2000)*, October 2000, pp. 404 – 409.

[15] M. Littman and J. Boyan, "A distributed reinforcement learning scheme for network routing," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-93-165, 1993.

[16] SensorGrid: Sensor networks integrated with grid computing. [Online]. Available: http://www.sensorgrid.org

[17] C.-K. Tham and R. Buyya, "SensorGrid: Integrating sensor networks and grid computing," *CSI Communications*, pp. 24–29, July 2005, invited paper.