Reinforcement Learning for Energy-Reducing Start-Up Schemes

Kristof Van Moffaert Yann-Michaël De Hauwere Peter Vrancx
Ann Nowé

Computational Modeling Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels
{kvmoffae, ydehauwe, pvrancx, anowe} @vub.ac.be
http://como.vub.ac.be

Abstract

In this paper, we present a learning technique for determining energy-reducing schedules for general devices and equipment. The proposed learning algorithm is based on Fitted-Q Iteration (FQI) and analyzes the usage and the users of a particular device to decide upon the appropriate profile of start-up and shut-down times. We experimentally evaluated our algorithm on a mixture of real-life and simulated data to discover that close-to-optimal control policies can be learned on a short timespan of a only few iterations. Our results show that the algorithm is capable of proposing intelligent schedules that minimize energy consumption and at the same time maximize user satisfaction.

1 Introduction

Automatic control policies received a great deal of attention in recent years. Combining such control policies with a low resource consumption and minimal costs is a goal that researchers from various disciplines are attempting to achieve. Traditional self learning or manually designed control strategies lack predictive capabilities to ensure a certain quality of service in systems that are characterized by diverse usage patterns and user preferences. As a result, such systems do not provide effective solutions for achieving the desired resource efficiency. Moreover, such traditional approaches typically also result in a significant risk of temporary discomfort as part of the learning phase or due to ill-configured systems.

In this paper we describe an approach that aims to automatically configure product systems to user demand patterns and their preferences. This means tailoring the performance of devices to the specific circumstances imposed on them by their everyday users. By taking into account patterns in user behavior and expectations, the system usage optimization is twofold. On the one hand side, the quality of service provided by the system to the end user, and on the other hand the resources needed to keep the system running. Such tailoring can be influenced by time dependent usage patterns as well as personal or group determined performance preferences. Both these aspects are brought together in the term *usage profile*.

Consider for instance a coffee machine. A coffee machine has different operational modes: on (making coffee), idle (temporarily heat water) and off. By default, the machine is idle. Every couple of minutes, the machine will re-heat it's water supply, to always be in a state of readiness when a user wants coffee. After office hours, the machine should be turned off manually, to bring down power consumption even further. Bringing the coffee machine from off to idle again in the morning mode requires a warming up phase, which implies that the machine is not immediately usable. On a typical day, the coffee machine used in an office environment will be turned on in the morning and remain on during the day, being used only sporadically. During long periods of time, the machine will be idling. Consistently turning it off after usage is a hindrance because the machine will need to warm up each time it is switched on again. Finding a correct control policy which optimizes energy consumption, without sacrificing human comfort will be the scope of the experiments described later on.

We propose a batch Reinforcement Learning (RL) approach that outputs a control policy based on historic data of usage and user preferences. This approach avoids the overhead and discomfort typically as-

sociated with a learning phase in reinforcement learning while still having the benefit of being adaptive to changing patterns and preferences. In Section 2, we elaborate on related concepts that allow automatic extracting of user patterns. Furthermore, we present our experimental setting in Section 3 and results in Section 4 which are discussed in the subsequent Section 5. To conclude the paper, we form conclusions on the results obtained in Section 6.

2 Background and preliminaries

In this section, we focus on related work of techniques concerning automatic retrieval of user patterns and profiles.

2.1 MDPs and Reinforcement Learning

A Markov Decision Process (MDP) can be described as follows. Let $S = \{s_1, \ldots, s_N\}$ be the state space of a finite Markov chain $\{x_l\}_{l \geq 0}$ and $A = \{a_1, \ldots, a_r\}$ the action set available to the agent. Each combination of starting state s_i, action choice $a_i \in A_i$ and next state s_j has an associated transition probability $T(s_j|, s_i, a_i)$ and immediate reward $R(s_i, a_i)$. The goal is to learn a policy π, which maps each state to an action so that the the expected discounted reward J^π is maximized:

$$J^\pi \equiv E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor and expectations are taken over stochastic rewards and transitions. This goal can also be expressed using Q-values which explicitly store the expected discounted reward for every state-action pair:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} Q(s', a') \tag{2}$$

So in order to find the optimal policy, one can learn this Q-function and then use greedy action selection over these values in every state. Watkins described an algorithm to iteratively approximate Q^*. In the Q-learning algorithm [13] a large table consisting of state-action pairs is stored. Each entry contains the value for $\hat{Q}(s, a)$ which is the learner's current hypothesis about the actual value of $Q(s, a)$. The $\hat{Q}$-values are updated accordingly to following update rule:

$$\hat{Q}(s, a) \leftarrow (1 - \alpha_t)\hat{Q}(s, a) + \alpha_t[r + \gamma \max_{a'} \hat{Q}(s', a')] \tag{3}$$

where α_t is the learning rate at time step t and r is the reward received for performing action a in state s.

Provided that all state-action pairs are visited infinitely often and a suitable evolution for the learning rate is chosen, the estimates, $\hat{Q}$, will converge to the optimal values Q^* [12].

2.1.1 Fitted-Q Iteration

Fitted Q-iteration (FQI) is a model-free, batch-mode reinforcement learning algorithm that learns an approximation of the optimal Q-function [2]. The algorithm requires a set of input MDP transition samples (s, a, s', r), where s is the transition start state, a is the selected action and s', r are the state and immediate reward resulting from the transition, respectively. Given these samples, fitted Q-iteration trains a number of successive approximations to the optimal Q-function in an off-line fashion. The complete algorithm is listed in Algorithm 1. Each iteration of the algorithm consists of a single application of the standard Q-learning update from Equation 3 for each input sample, followed by the execution of a supervised learning method in order to train the next Q-function approximation. In the literature, the fitted Q-iteration framework is most commonly used with tree-based regression methods or with multi-layer perceptrons, resulting in algorithms known as Tree-based Fitted Q-iteration [6] and Neural Fitted-Q iteration [11]. The FQI algorithm is particularly suited for problems with large input spaces and large amounts of data, but where direct experimentation with the system is difficult or costly.

Algorithm 1 Fitted Q-iteration

$\hat{Q}(s,a) \leftarrow 0 \ \ \forall s,a$ ▷ Initialize approximations
repeat
  T,I $\leftarrow \emptyset$
  for all samples i **do** ▷ Build training set
    I $\leftarrow$ I $\cup (s_i, a_i)$ ▷ Input values
    T $\leftarrow$ T $\cup \ r_i + max_a \hat{Q}(s'_i, a)$ ▷ Target output value
  end for
  $\hat{Q} \leftarrow$ Regress(I,T) ▷ Train supervised learning method
until Termination
return $\hat{Q}$ ▷ Return final Q-values

3 Problem setting

In our experiments, we had to model several aspects of the users of a particular household device and the properties of the device itself, e.g. the startup and idle costs. For the relevance of our results, it is important that these models and distributions are as close as possible to the real world. In the sections below, we elaborate into detail how this data was generated.

3.1 Presence and usage probabilities

In the experiments we conducted below, we simulated the presence and usage of six individual users of a household machine. Their presence probability and the average usage of each individual is depicted in Figure 1 and 2, respectively. For the moment, the days we simulate are considered general working days with some deviation in the arrival and departure times of the users. Currently, the devices we are controlling consist of beverage machines, such as a coffee maker or a water dispenser.

For each of the six users, days are generated using these graphs as a probability distribution together with noise added from a Normal distribution. Notice that we also added a user that is present a lot of the time, but does not consume any beverages. It is interesting to notice that for both distribution graphs that during lunch break, both probabilities are diminishing as most people then spend time elsewhere, whereas before and after this timeslot most users tend to be present and drink beverages.

Similar to the distribution concerning presences, we use this distribution as input for a Poisson [7] distribution to generate different usages for different people for different days. The Poisson distribution is especially tailored for expressing the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event. The combination of the two graphs allow us to generate a variety days with simulated presences and usages up to the level of minutes, i.e. for each simulated day 1440 data points are collected.

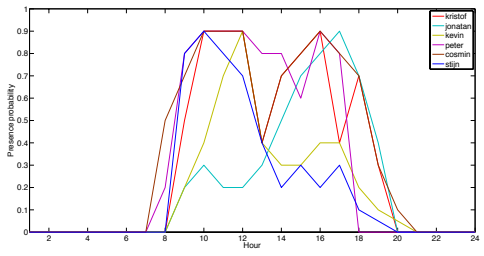

Figure 1: The probability distribution for each of the six individual users on their presence.

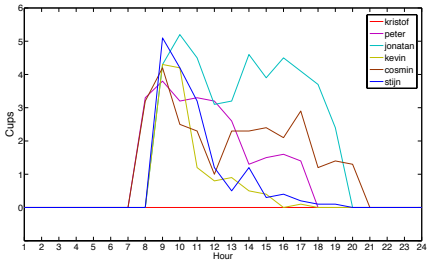

Figure 2: Every user also has a distribution on their usage and drinking behavior.

3.2 A general device model

Another important part of our experimental setting is the model used to represent the device, being controlled. This model should be both general and specific enough to capture all aspects of common household

devices. A Markov Decision Process, as introduced in Section 2.1, is specifically tailored for representing the behavior of a particular household device. In total, two possible actions and three states are presented in an MDP that would cover most, if not all household equipment. The three states or modes of the MDP are '*on*', '*off*' or '*booting*', where the latter represents the time needed before the actual operational mode is reached. The action space A of our MDP is limited to two distinct, deterministic actions, i.e. the agent can either decide to press a switch or relay that alters the mode of the machine or it can decide do leave the mode of the machine unchanged and do nothing. The former action is a simplification to two separate actions '*turn on*' and '*turn off*'.

An aspect of the MDP that we did not cover yet is the immediate reward $R_a(s, s')$ received after transitioning to state s' from state s by action a. These rewards are a combination of two objectives, e.g. an energy consumption penalty or cost and a reward given by the user. The latter is a predefined constant for different situations that can occur. For instance, when the machine is turned off but at the same time a user wanted coffee, then, the current policy does not meet that specific user's profile and the policy is manually overruled. In such a case, the system is provided with a negative feedback signal indicating the user's inconvenience. On the other hand, when the device is turned on at the same time that a user requested a beverage, then the policy actually suits the current user and the system anticipated well on his usage. In those cases, positive reward is provided to the system.

The former reward signal is a measure indicating quality of a certain action a on the level of power consumption. These rewards are device-dependent and allow the learning algorithm to learn over time whether it is beneficial to have the device in idle mode or if a shutdown is needed. By specifying a certain cost for cold-starting the device, in according to the real-life cost, the algorithm could also learn to power the device on x minutes before a timeslot with a lot of consumption is expected. In general, the learning algorithm will have to deduct which future timeslots are expected to have a positive difference between the consumption reward signal and the user satisfaction feedback signal. For the moment, these two reward signals are combined into one by a summation.

To conclude, our MDP is graphically represented in Figure 3 and is mathematically formalized as follows: $M = < S, A, P, R >$, where S = {On, Off, Booting} and A = {Do nothing, press switch}. The transitions between the different states are deterministic, resulting in a probability function P that is shown in Figure 3. The reward function R is device-specific and we will elaborate this function in the sections below.

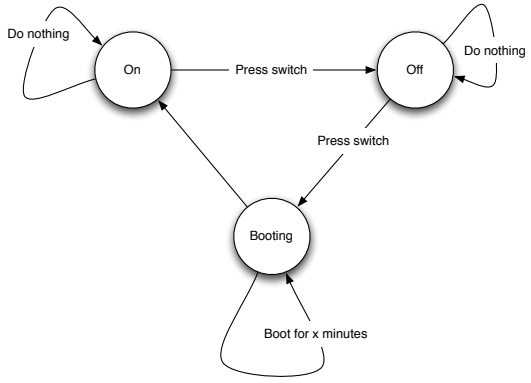

Figure 3: A general model for almost every household device.

4 Empirical results

As already mentioned in Section 3, the devices we are focusing on to optimize, are beverage machines such as a coffee maker or a water dispenser. For each of these devices, we use the same MDP M, depicted in Section 3.2, but we use different reward functions to cover the specific behavior of these machines in real-life. At each of those data points, representing a point in time, the FQI algorithm, described in Section 2.1.1, will decide which action to take from the action space given the current hour, minute and presence set, with 24, 60 and 2^6 possible values, respectively. These figures results in a large state space of 92,160 possible combinations. In our setting, the FQI algorithm was first trained with data of one single simulated day and

the control policy was tested for one new day after every training step, whereafter this test sample was also added to the list of training samples to increase the training set's size. Thus, an on-line learning setting was created. In our experiments, we opted for the Tree-Based FQI algorithm with a classification and regression tree or CART [1] and we averaged our results over 10 individual trials.

4.1 Coffee maker experiment

For the first experiment, we mimicked the properties of a real-life coffee maker into our simulation framework. Using appliance monitoring equipment, we have tried to capture the real-life power consumption of the device under different circumstances. After measuring the power consumption of the machine for a few weeks, we came to the following conclusions:

- We noticed that, for our industrial coffee maker, the start-up time was very fast. In just over one minute, the device heated the water up to the boiling temperature and the beverage could be served. The power consumption of actually making coffee is 950 Watts per minute.
- When the machine was running in idle mode, the device is only using around 2 Watts most of the time. However, every ten minutes, the coffee maker re-heated its water automatically. On average, this results in an energy consumption of 50 watts per minute in idle mode.
- The device does not consume any power when turned off.

We normalized these values and incorporated them as rewards and costs into our simulation framework. The results are depicted below. In Figure 4, the average collected reward over 100 simulation days and ten trials is denoted. Starting from an inferior initial policy, the FQI algorithm is able to generalize the large state space and gradually refine its policy. This measure is an indication that indeed energy reducing policies can be learned over time. On the other hand, Figure 5 depicts the number of manual overrides of the proposed policy over each day. We see that in initial runs up to 60 overrides were recorded, where in final runs of the algorithm this is reduced to just under five manual interventions of the user. The combination of both graphs, i.e. the graphs where the collected reward is shown and the one with the number of manual, human interventions, conclude that appropriate schedule can be learned that focus on both energy reduction and user convenience. Figure 6 concludes the experiment on the coffee maker by showing the best policy found by the system. A little before 8 in the morning, before most of the users arrive at the site, the algorithm suggest to turn the machine and leave it on until 14h. Based on the generated days, it was beneficial to have the device turned off for 15 minutes. This could be the case because of the fact that that in the usage data from Figure 2, we have a significant drop in usage at 14h for some heavy drinkers, such as Peter, Kevin and Stijn. Although some people tend the be present until 20h to 22h on some occasions, the algorithm has learned that evening hours are not very beneficial to have the device turned on, because of the low consumption of beverages. We also notice that the proposed policy does not turn the machine on the entire time between 8h and 18h, i.e. there are some minimal moments where the device is turned off for only a few minutes. The reason behind this behavior is two-fold. First, this could be the case because the 100 simulation days did not provide enough information on those particular moments and further learning is needed. Secondly, because the device is up and running in little over one minute, the algorithm can in fact afford to save energy on such a short term as it has observed that the start-up time is minimal.

4.2 Water dispenser experiment

In a second experiment, we tried to perform a similar test where we focused on a different device with different characteristics, i.e. a water dispenser machine. This time, we want to investigate the outcome when different rewards or costs are provided for both the start-up and idle state. More precisely, the start-up costs was divided by four compared to the previous experiment and the idle cost was doubled. Additionally, we extended the start-up time required for the device to be in operational mode from one to ten minutes. We also limited our set of people being monitored to four (i.e. Stijn, Cosmin, Kristof and Kevin).

Similar results as in Section 4.1 were achieved for the collected reward and number of manual overrides of the policy in Figure 7 and 8, respectively. What we do notice is that as a result of the fact that the start-up and the idle cost are approaching each other, the algorithm starts to suggest to shut the device down for particular moments in time. Where in the previous experiment, the device was left on during the lunch break, the algorithm suggested a more conservative schedule with the device turned of for one hour between

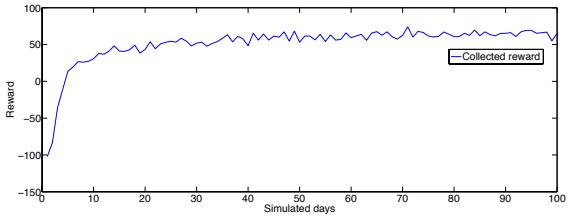

Figure 4: By observing the feedback on the users' convenience and energy consumption and acting in correspondence of these signals, the policies are being tailored to the users and more reward is being received.

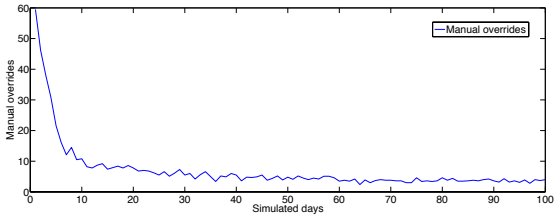

Figure 5: The number of manual interventions and overrides decrease over time as the policy becomes more and more refined to the users and their profiles. In the end, only four manual interventions occur on average.

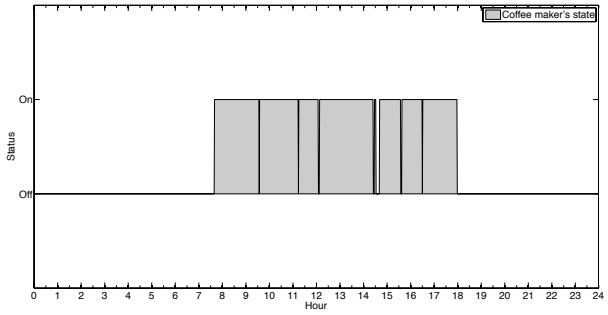

Figure 6: For every point in time, the suggested action of the best policy is graphically represented. The learned policy is specifically suited for a general weekday. where people tend to be present from 9h to 17h with some slight deviations.

12h30 and 13h30. Subsequently, the device is also turned off for a shorter timespan around 11h. Another consequence of these cost functions is the fact that the number of manual overrides (Figure 8) remains quite high because the proposed schedule is now more conservative and meets the user demands less. We also notice from the same figures that the policy is achieved noticeably faster than in the previous experiment, where the number of users being taken into account was larger.

5 Discussion and Related Work

In the results, presented in Section 4, we have seen the FQI algorithm in action on a large state space with features corresponding to time-based information and a set of presences. Provided with only a few training points, the algorithm, in combination with its classification and regression tree, managed to obtain acceptable policies after only a few iterations and was able to increase its collected reward while reducing the number of manual interventions needed. Throughout the experiments, we have seen that when providing the model with different reward and cost structures the policies can be heavily influenced. For instance, when an relatively high cost corresponds to leaving the machine running in idle mode, we notice that the algorithm proposed a much more conservative schedule than before and suggests to have the device turned off for longer periods of time.

Previous research has applied the Fitted-Q algorithm mainly in single-objective optimal control problems (e.g. [2, 11, 6]). More recently, [3] also introduced a multi-objective FQI version, which is capable of approximating the Pareto frontier in learning problems with multiple objectives, and applied this algorithm to learn operation policies for water reservoir management. None of these works, however, consider the problem of user interactions and taking into account end-user preferences. To the best of our knowledge this paper presents the first application of FQI in a setting which includes both a cost function and direct user feedback.

Several authors have considered other reinforcement learning algorithms in problem settings related to those presented in this paper. In Dalamagkidis et al. [4], an online temporal difference RL controller is developed to control a building heating system. The controller uses a reinforcement signal which is the weighted combination of 3 objectives: energy consumption, user comfort and air quality. Online RL

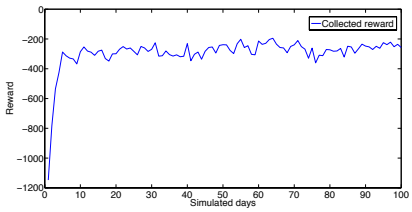

Figure 7: The collected rewards in the water dispenser experiment increased rapidly and the policy converged after 10 iterations on average.

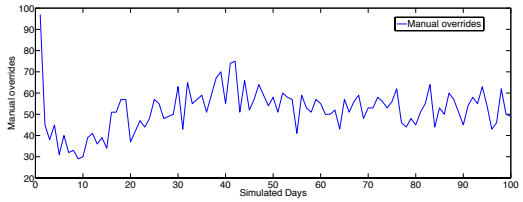

Figure 8: The number of human interventions quickly dropped as more days were being simulated.

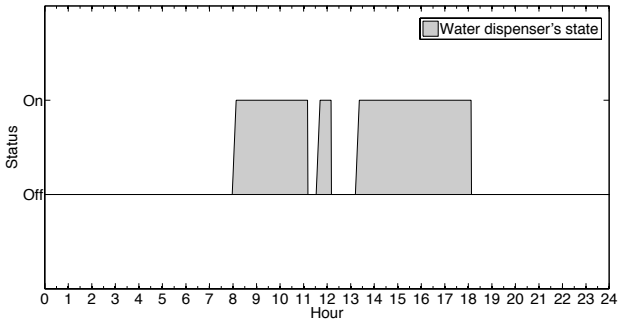

Figure 9: The proposed policy of the algorithm suggests to have the machine on before and after the lunch break. This schedule is more conservative and shuts the device down during noon.

algorithms have also been applied to the problem of energy conservation in wireless sensor networks, often in combination with other objectives such as satisfying certain routing criteria (see e.g. [9, 10, 5]). Finally, Khalili et al. [8] apply Q-learning to learn user preferences in a smart home application setting. Their system is able to adapt to (time-varying) user preferences regarding ambient light and music settings, but does not take into other criteria such as account energy consumption.

6 Conclusions

Throughout this paper, we have elaborated the need for algorithms that allow automatic control and optimization of common devices, that at the same time reduce the energy consumption and keep an acceptable level of user convenience. We have elaborated a learning method, based on reinforcement and supervised learning techniques to deduct appropriate start-up and shutdown schedules for common household equipment. A simple model is proposed that allows us to generalize every device and was instantiated to represent two different scheduling cases. For both experiments, the learning agent proposed logical schedules after only a few iterations that focus on both energy reduction and user satisfaction.

In future work, we will focus on taking this framework to the next level and build a real-life setting where the algorithm's policies can be used by a controller device and where they can be gradually refined by manual interventions by real-life users of the system. We will also look into details whether we can combine both reward signals, e.g. the consumption and the user satisfaction feedback signal, in a more intelligent way by, for instance, incorporating techniques from multi-objective optimization.

Acknowledgements

This research is supported by IWT-SBO project PERPETUAL. (grant nr. 110041).

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[2] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*, volume 39 of *Automation and Control Engineering Series*. CRC Press, 2010.

[3] A. Castelletti, F. Pianosi, and M Restelli. Tree-based fitted q-iteration for multi-objective markov decision problems. In *Proceedings International Joint Conference on Neural Networks (IJCNN 2012)*, 2012.

[4] K. Dalamagkidis, D. Kolokotsa, K. Kalaitzakis, and G.S. Stavrakakis. Reinforcement learning for energy conservation and comfort in buildings. *Building and Environment*, 42(7):2686 – 2698, 2007.

[5] M. Devillé, Y-A. Le Borgne, and A. Nowé. Reinforcement learning for energy efficient routing in wireless sensor networks. In P. De Causmaecker, J. Maervoet, T. Messelis, K. Verbeeck, and T. Vermeulen, editors, *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, pages 89–96, Ghent, Belgium, 2011.

[6] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[7] F.A. Haight. *Handbook of the Poisson distribution*. Publications in operations research. Wiley, 1967.

[8] A.H. Khalili, C. Wu, and H. Aghajan. Autonomous learning of users preference of music and light services in smart home applications. In *Proceedings Behavior Monitoring and Interpretation Workshop at German AI Conf.*, 2009.

[9] Zhenzhen Liu and Itamar Elhanany. A reinforcement learning based mac protocol for wireless sensor networks. *Int. J. Sen. Netw.*, 1(3/4):117–124, January 2006.

[10] M. Mihaylov, K. Tuyls, and A. Nowé. Decentralized learning in wireless sensor networks. *Lecture Notes in Computer Science*, 4865:60–73, 2010.

[11] Martin Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In *In 16th European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[12] J.N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Journal of Machine Learning*, 16(3):185–202, 1994.

[13] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989.