Blind Decision Making: Reinforcement Learning with Delayed Observations

Mridul Agarwal[1], **Vaneet Aggarwal**[1, 2]
[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]School of Industrial Engineering, Purdue University, West Lafayette, IN, USA
agarw180@purdue.edu, vaneet@purdue.edu

Abstract

In Reinforcement Learning (RL) the current state of the environment may not always be available. One approach to fix this could be to include the actions after the last-known state as a part of the state information, however, that leads to an increased state-space making the problem complex and slower in convergence. We propose an approach, where the delay in the knowledge of the state can be used, and the decisions are made to maximize the expected state-action value function. The proposed algorithm is an alternate approach where the state space is not enlarged, as compared to the case when there is no delay in the state update. Evaluations on the basic RL environments further illustrate the improved performance of the proposed algorithm.

Introduction and Related Work

Applications of Reinforcement Learning (RL) are continuously increasing in domains that can be formulated using state, action, and rewards. For example cloud scheduling (Arabnejad et al. 2017), robot manipulation (Clavera et al. 2019), and microgrid management (Kuznetsova et al. 2013). However, in many such applications the state update information is not available instantaneously. As an example, micro-grid control may have stochastic delays because of the communication link and these delays may have adverse impact on the system (Liu, Wang, and Liu 2015). The issue of such delay in the availability of state information limits the use of RL for practical applications (Mahmood et al. 2018). This paper proposes an algorithm for environments where the state updates are not immediately available.

As studied by (Katsikopoulos and Engelbrecht 2003), delays may be of three types, 1) delays in observations, 2) delays in executing actions, and 3) delays in obtaining cost/reward delays. Katsikopoulos and Engelbrecht (2003) show that action delays are equivalent to observation delays. If observations are delayed then the agent plays action with knowledge of the last observed state and the actions happened in the meantime. Similarly, if actions are delayed, the agent schedules actions for the future with the same information. In case of delays in rewards updates, algorithms train using mini-batches where a mini-batch consists of state observations, actions taken, and available rewards received for some duration. In this paper, we assume only observation or action delays. We further assume that rewards are provided to the agent along with state updates.

Altman and Nain (1992) show that for Markov Decision Process (MDP) where each observation is delayed by d steps, constructing an equivalent MDP with an augmented state space where previous d actions are appended to the currently known state restores the problem structure back to an MDP. However, this approach does not scale to stochastic delays. In order to resolve this, (Katsikopoulos and Engelbrecht 2003) proposed a new solution by assuming that the maximum delay is bounded. They assume that if the delays are more than a threshold n, then the algorithm freezes, and would not take any action. However, many real-time systems may not allow freezing of execution.

Further, the expansion of the state space might not be efficient for implementation because of the increased storage complexity. In order to alleviate this, the authors of (Walsh et al. 2009) proposed an algorithm to play action a_t that is optimal for the most likely state. It was assumed that the probability of not being in the most likely state is bounded by δ, where δ is small enough. For MDPs where state distributions are not concentrated, this assumption might not hold. Schuitema et al. (2010) presented memoryless algorithms dQ and dSARSA which work by updating Q-value corresponding to the delayed state directly. However, this approach cannot work with stochastic delays. Lastly, the regret analysis of the MDP shows that the regret for the augmented MDP scales as $A^{d/2}$, where d is the delay in the availability of the state information and A is the number of actions (Jin et al. 2018). For large d, the gap may be large enough for the approach to have significantly decreased performance.

We propose a solution that aims to alleviate these limitations. The proposed algorithm, called *Expectated-Q Maximization* (EQM), takes an action that maximizes the expected gain of true MDP across all possible states conditioned over the last known state s_{t-d}, and actions taken till time t. EQM for delayed reinforcement learning is, **(1) Space efficient**: The proposed algorithm does not use an augmented MDP to determine the action. It, however, uses the fact the current true state comes from the probability distribution generated by augmented MDP. **(2) Robust under deviation from most likely state**: The algorithm selects the

action that maximizes the expected value Q function. Thus, even though the distribution is not concentrated around a single state, the distribution is efficiently utilized. **(3) Handles stochastic delays:** The algorithm works well with stochastic delays, as well as missed information.

We evaluate EQM on Frozen Lake (8×8) grid, and Cart Pole environments of OpenAI Gym platform (Brockman et al. 2016). The results for delayed settings are compared with Extended MDP formulation of (Altman and Nain 1992), MBS algorithm given by (Walsh et al. 2009), and dQ algorithm proposed by (Schuitema et al. 2010) respectively. The metric of comparison is the total reward, collected in each episode.

Formulation

We consider a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, P, R)$, with set of states denoted by $\mathcal{S}$, and set of actions denoted by $\mathcal{A}$. At time t, the environment is in state $s_t \in \mathcal{S}$. The definitions are mostly consistent with those in (Sutton and Barto 2018; Puterman 2014). At any time t, the agent chooses action $a_t \in \mathcal{A}$ based on its knowledge about the current state. On playing the action a_t, environment rewards the agent with R_t, which is random variable conditioned on environment state s_t, and action chosen by agent at time t. The maximum reward the agent can receive at any time step is R_{max}. The goal of the agent is to maximize the discounted cumulative rewards it receives. The discount factor $\gamma \in [0, 1)$ denotes the importance of future rewards.

The probability distribution of next state s_{t+1} conditioned on current state s_t and action a_t is denoted by $p(s_t, a_t, s_{t+1})$. Shorthand notation by dropping the subscripts is denoted as,

$$p(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a) \tag{1}$$

Agent selects an action according to a stationary policy π as $a_t = \pi(s_t)$. The value function $V^\pi(s)$ of a state s is defined as the expected value of sum of discounted rewards which agent can receive over time starting from state s and choosing actions according to the policy π.

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=t_0}^{\infty} \gamma^{t-t_0} R_t \Big| s_{t_0} = s \right] \tag{2}$$

$$= \mathbb{E}_\pi \left[\sum_{t=t_0}^{\infty} \gamma^{t-t_0} r(s_t, a_t) | s_{t_0} = s \right] \tag{3}$$

where $r(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a]$ is the expected reward when action a is taken in a state s is defined as $r(s, a)$

This makes the maximum possible value of $V^\pi(s)$ as $\frac{R_{max}}{1-\gamma}$. Similarly action-value function $Q^\pi(s, a)$ is defined as the expected cumulative rewards which agent receives in state s on taking action a and then following policy π,

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}[V^\pi(s_{t+1}) | s_t = s, a_t = a] \tag{4}$$

For both value function $V^\pi(s)$ and action-value function $Q^\pi(s, a)$ the expectation is taken over the states which are distributed according to the transition dynamics of the MDP $\mathcal{M}$ and actions which are distributed according to the policy π. We use only π in the subscript for expectation as we can only control the policy. Optimal policy π^* is defined as the policy which maximizes the value function for all states.

$$V^*(s) = V^{\pi^*}(s) = \sup_\pi V^\pi(s) \quad \forall\, s \in \mathcal{S} \tag{5}$$

$$= \max_{a \in \mathcal{A}} Q^*(s, a) \quad \forall\, s \in \mathcal{S} \tag{6}$$

We also assume that all realizations of delay d to be a non-negative integer. At time t, the last known state for the agent is s_{t-d}. The actions played in d time steps are $a_{t-d}, a_{t-(d-1)}, \cdots, a_{t-1}$. We assume that at the beginning of any episode, all the delayed observations of previous episodes are available. This also means that the observations from any of the previous episodes are not corrupting the observations received in current episode.

Proposed Policy and Bounds

For a system with d delays, we construct an extended MDP $\widetilde{\mathcal{M}}$ which has state $\tilde{s}_t$ as $(s_{t-d}, a_{t-d}, a_{t-(d-1)}, \cdots, a_{t-1})$. The two MDPs $\widetilde{\mathcal{M}}$, and $\mathcal{M}$ share the same action space $\mathcal{A}$, so we will not change the notation for actions. For everything else we will put a tilde over the variables for the augmented MDP. The policy $\tilde{\pi}$ now selects an action based on $\tilde{s}$ or the tuple $(s_{t-d}, a_{t-d}, a_{t-(d-1)}, \cdots, a_{t-1})$. The corresponding Q-function for a policy $\tilde{\pi}$ over $\widetilde{\mathcal{M}}$ becomes,

$$\widetilde{Q}^{\tilde{\pi}}(\tilde{s}, a) = \mathbb{E}[R_t | \tilde{s}_t = \tilde{s}, a_t = a] + \gamma \sum_{\tilde{s}_{t+1} \in \widetilde{\mathcal{S}}} \widetilde{V}^{\tilde{\pi}}(\tilde{s}_{t+1}) \mathbb{P}[\tilde{s}_{t+1} | \tilde{s}_t = \tilde{s}, a_t = a] \tag{7}$$

Using this construction we present the key lemma based on which we construct our policy.

Lemma 1. *Expected reward obtained by agent in augmented state $\tilde{s}_t$ by taking an action a, is related to the true state s_t of environment as*

$$\tilde{r}(\tilde{s}_t, a) = \sum_{s \in \mathcal{S}} r(s, a) p(s | \tilde{s}_t) \tag{8}$$

Proof. (Outline) The reward R_t generated by the environment is oblivious to the state maintained by the agent. Using the tower rule of expectation, we get the required result. A detailed proof is provided in our technical report (Agarwal and Aggarwal 2020). □

Lemma 1 states that the expected reward received on taking action a in state $\tilde{s}$ is the expected reward received by taking action a in the unobserved stated conditioned on $\tilde{s}$. Based on Lemma 1, a myopic policy, which maximizes immediate expected return for the agent, selects greedy action $a_t(\tilde{s})$ as

$$a_t = \arg\max_{a \in \mathcal{A}} (\mathbb{E}_s[R(s, a) | \tilde{s}_t]) \tag{9}$$

Inspired by the myopic policy, we now propose a policy for working with delayed state updates.

We propose a policy that maximizes expected Q value (over the states) of the optimal policy for $\mathcal{M}$, or

$$\tilde{\pi}(a_t | \tilde{s}_t) = \begin{cases} 1, & \text{if } a_t = \arg\max_{a \in \mathcal{A}} \mathbb{E}[Q^*(s, a) | \tilde{s}_t], \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The following theorem provides bounds on minimum value an augmented state $\tilde{s} \in \widetilde{\mathcal{S}}$ would fetch for the agent. That is, value function using the policy defined in Equation (10) ensures the minimum value given in Theorem 1.

Theorem 1. *If the agent follows policy as given in (10), for an augmented MDP $\widetilde{\mathcal{M}}$, then the value of each state $\tilde{s} \in \widetilde{\mathcal{S}}$ satisfies the following lower bound,*

$$\widetilde{V}^{\tilde{\pi}}(\tilde{s}) \geq \mathbb{E}_{s|\tilde{s}}\left[V^{\pi^*}(s)\right] - \frac{R_{max}}{(1-\gamma)^2}\left(1 - \frac{1}{|\mathcal{A}|}\right) \tag{11}$$

where $\pi^*(a_t|s_t)$ is oracle aided policy which gives the optimal action for true MDP $\mathcal{M}$.

Proof. We first mention and prove the lemmas required for the proof, and then continue to the final proof. Lemma 2 relates the value function for $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ under same policy $\tilde{\pi}$. Lemma 3 relates expected value using the oracle aided policy and the state-action value using oracle aided policy.

Lemma 2. *Value function of a policy $\tilde{\pi}(a|\tilde{s})$ for augmented MDP $\widetilde{\mathcal{M}}$ is related to value function for same policy under true MDP $\mathcal{M}$ is related as*

$$\tilde{V}^{\tilde{\pi}}(\tilde{s}) = \mathbb{E}_s\left[V^{\tilde{\pi}}(s)|\tilde{s}\right] \tag{12}$$

where $V^{\tilde{\pi}}(s)$ is value function for policy $\tilde{\pi}$ with MDP $\mathcal{M}$.

Proof. (Outline) We get the required result by using linearity of expectation, Lemma 1 and the fact that the environment state transition probability does not change from MDP $\mathcal{M}$ to $\widetilde{\mathcal{M}}$. □

Lemma 3. *For the optimal policy π^*, and distribution over initial state s, μ,*

$$\max_a \mathbb{E}_{s\sim\mu}\left[Q^*(s,a)\right] \geq \frac{1}{|\mathcal{A}|}\mathbb{E}_{s\sim\mu}\left[V^*(s)\right] \tag{13}$$

Proof. Let initial state follow some distribution μ, or $\mathbb{P}\left[s_t = s\right] \sim \mu$. Then we have,

$$\begin{aligned}\mathbb{E}_{s\sim\mu}\left[V^*(s)\right] &= \mathbb{E}_{s\sim\mu}\left[\max_a Q^*(s,a)\right] \\ &\leq \mathbb{E}_{s\sim\mu}\left[\sum_{a\in\mathcal{A}} Q^*(s,a)\right] \leq |\mathcal{A}| \max_{a\in\mathcal{A}} \mathbb{E}_{s\sim\mu}\left[Q^*(s,a)\right]\end{aligned} \tag{14}$$

□

We can now use Lemma 2, Lemma 6.1 from (Kakade and Langford 2002) and Lemma 3 to find the minimum value of $\widetilde{V}^{\tilde{\pi}}(\tilde{s}) - \mathbb{E}_s\left[V^{\pi^*}(s)|\tilde{s}\right]$. A detailed proof is provided in (Agarwal and Aggarwal 2020). □

Theorem 1 states that the proposed policy can suffer a maximum degradation of $\left(1 - \frac{1}{|\mathcal{A}|}\right)\frac{R_{max}}{(1-\gamma)^2}$ only from the expected optimal value of unobserved state conditioned on the extended state.

Now, the task that remains is to find Q^* for true MDP. We assume that the delayed state observations can be identified using timestamp or index header. This is a common engineering principle in communication networks to deal with asynchronous packets (Walrand and Parekh 2010), and hence it is a valid assumption. This allows to find optimal Q-function for true MDP $\mathcal{M}$. We next provide a detailed algorithm for the policy described in this section.

Algorithm

We now utilize Equation (10) to construct *Expectated-Q Maximization* (EQM) algorithm (described in Algorithm 1) which is space efficient and which can handle stochastic delays. We keep track of the estimates of true transition probabilities to calculate the Q-value function for the optimal policy.

Algorithm Description and Complexity

We assume that the state value feedback is mapped with a time stamp to obtain a state-action pair and the next state into which the environment transitions. We maintain variables corresponding to number of times a state-action pair was visited $N(\cdot,\cdot)$ and the counter for next state from a state-action pair $P(\cdot,\cdot,\cdot)$ to calculate the estimates of probability transitions $(\hat{p}(\cdot,\cdot,\cdot))$ as

$$\hat{p}(s',a,s) = \frac{P(s',a,s)}{\max\{1, N(s,a)\}}. \tag{15}$$

We also calculate the Q-values of states and actions as:

$$Q(s,a) = \hat{r}(s,a) + \sum_{s'\in\mathcal{S}} \gamma\left(\max_{a'} Q(s',a')\right)\hat{p}(s,a,s') \tag{16}$$

The estimates of transition probabilities and Q-values are improved as the algorithm explores. *Get_EQM_Action* is presented in Algorithm 1. If the current state is available, the expected Q-function becomes the true Q function. Algorithm 1 computes the expected Q-value for Equation (10) from the probability estimates. At each time step t, Algorithm 1 computes the expected value of the Q-function whenever an action needs to be taken. This requires $\mathcal{O}\left(d|\mathcal{S}|^2 + \log\left(|\mathcal{A}|\right)\right)$ computations. For our algorithm, we calculate the probability vector $\bar{\mathbf{p}}$ which is the conditional probability distribution of the states given the last known state s_{t-d}, and the sequence of actions $a_{t-d}, \cdots, a_{t-1}$. For time $t-d$, the true state is known and conditional probability becomes

$$\bar{\mathbf{p}}_{t-d}(s) = \begin{cases} 1, & s = s_{t-d}, \\ 0, & otherwise \end{cases} \tag{17}$$

Then, for each next time step, the probability vector is updated using the following recursion equation.

$$\bar{\mathbf{p}}_{t-k} = \left(\hat{p}(:, a_{t-(d-k)}, :)\right)^T \bar{\mathbf{p}}_{t-k-1} \;\forall\, s \in \mathcal{S}, \forall\, 1 \leq k < d \tag{18}$$

where $\hat{p}(:, a_{t-k-1}, :)$ is the state transition matrix of MDP $\mathcal{M}$ induced by action a_{t-k-1}. Since we do this update d times, and each matrix multiplication costs $\mathcal{O}\left(|\mathcal{S}|^2\right)$, the total complexity to compute the state probability at each time step t becomes $\mathcal{O}\left(d|\mathcal{S}|^2\right)$. Fetching the maximum element cost extra $\mathcal{O}\left(\log\left(|\mathcal{A}|\right)\right)$. The overall complexity at any time step t thus becomes $\mathcal{O}\left(d|\mathcal{S}|^2 + \log\left(|\mathcal{A}|\right)\right)$. The complete algorithm is provided in (Agarwal and Aggarwal 2020).

Evaluation

We evaluate our algorithm EQM on the Cart Pole problem of OpenAI Gym (Brockman et al. 2016). The Cart Pole environment has continuous state space with discrete action space.

Algorithm 1 *Get_EQM_Action*

Input: $\mathcal{S}$, $\mathcal{A}$, $\hat{p}$, Q, $(s_{t-d}, a_{t-d}, \cdots, a_{t-1})$
Output: Estimated greedy action a_t
$\bar{\mathbf{p}} = [0, \cdots, 0]$, vector of length $|\mathcal{S}|$
$\bar{\mathbf{p}}[s_{t-d}] = 1$
for $0 \leq k < d$ **do**
  $\bar{\mathbf{p}} = \left(\hat{p}(:, a_{t-(d-k)}, :)\right)^T \bar{\mathbf{p}}$
end for
$\bar{\mathbf{Q}} = \bar{\mathbf{p}}^T Q$
Return $\arg\max_a \bar{\mathbf{Q}}$

We compare our algorithm with Extended MDP formulation by (Altman and Nain 1992), Model Based Simulation (MBS) algorithm of (Walsh et al. 2009), and dQ algorithm of (Schuitema et al. 2010) for constant delays. We also compare the proposed EQM algorithm with MBS algorithm for stochastic delays. The metric of comparison is total cumulative reward accumulated at the end of each episode averaged over last 50 episodes. Exploration factor is time dependent and is chosen as $\epsilon_t = \frac{\log\left(|\mathcal{S}| \sum_{s,a} N(s,a)+1\right)}{\sum_a N(s,a)+1} \ \forall s, a$. This choice of exploration factor is same across all simulations. For stochastic delays, creating an augmented MDP is not feasible as delays can be arbitrarily large. We run 20 independent iterations to obtain confidence intervals. Each iteration is trained over 1000 episodes.

For constant delays, we chose delays in the range of $d \in \{2, 4\}$. For stochastic delays, each observation was independently delayed by delays generated using a geometric distribution with parameter p. The expected delay for this distribution is $\frac{1}{1-p}$. We note that, this may create asynchronous observations as delay d_1 of observation at t_1 may be higher than delay d_2 of observation at t_2, where $t_1 + d_1 > t_2 + d_2$. The issue of asynchronous delays can be dealt by introducing time stamps in observations.

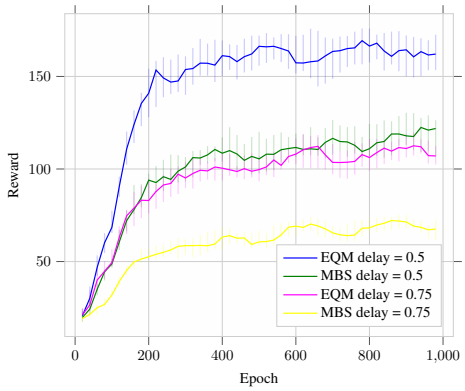
Results

Simulations results based for both constant delays and stochastic delays are presented in Figure 1. We plot median of rewards in each iteration along with the top and bottom quantiles.

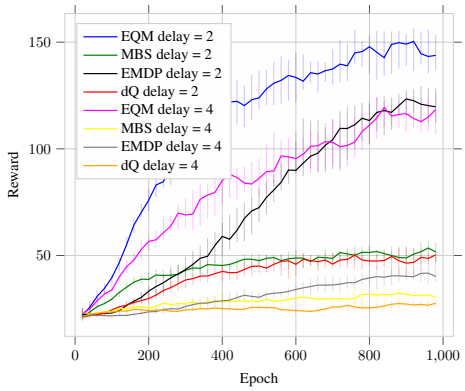
As observed in Figure 1, total reward per episode is higher for EQM algorithm compared to all other algorithms (MBS, dQ, EMDP) in the presence of constant delays. As delay increases, gap between the rewards also increase between the two algorithms. For MBS algorithm, this can be reasoned as - with large values of delays the state with largest likelihood might have lower probability of occurring.

For dQ algorithm, the reason for increasing difference in accumulated rewards can be credited to the memoryless algorithm. Also, Extended MDP algorithm is slow in convergence because of larger state space which grows exponentially. This reduction in convergence speed is visible in Figure 1 where delay of 4 time steps cause a significant drop in performance of Extended MDP algorithm.

For stochastic delays, the difference between the rewards accumulated by EQM, and MBS algorithm increases when the probability of delay increase. The EQM algorithm beats the MBS algorithm significantly even when the stochastic delays are geometric distributed with expected delay of 1 unit in Figure 1.



(a) Stochastic delays



(b) Constant delays

Figure 1: Reward accumulated by EQM, MBS, EMDP, and dQ-learning algorithm with constant delays and stochastic delays. EQM achieves higher average reward per episodes compared to other algorithms.

Conclusion

We considered the problem of delays in observation updates for a reinforcement learning agent where the current state of the environment is not immediately available to the agent. We proved that the expected immediate rewards generated for MDP with delays is same as expected immediate rewards generated for corresponding extended MDP without delays. We proposed a new policy which can handle stochastic delays by optimizing on optimal Q-function of the true MDP. We then provided a lower bound on the value function for all states following the proposed policy. Based on this policy, we proposed a new algorithm, *Expectated-Q Maximization* (EQM), which is robust under constant, and stochastic delays.Evaluations demonstrate the improvement over existing algorithms under constant, and stochastic delays. Studying the convergence rates of the Q-value functions in presence of delays and extending EQM to deep neural networks remains potential future work.

References

Agarwal, M.; and Aggarwal, V. 2020. Blind Decision Making: Reinforcement Learning with Delayed Observations. *arXiv preprint arXiv:2011.07715* .

Altman, E.; and Nain, P. 1992. *Closed-loop control with delayed information*, volume 20. ACM.

Arabnejad, H.; Pahl, C.; Jamshidi, P.; and Estrada, G. 2017. A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '17, 64–73. Piscataway, NJ, USA: IEEE Press. ISBN 978-1-5090-6610-0. doi: 10.1109/CCGRID.2017.15. URL https://doi.org/10.1109/CCGRID.2017.15.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR* abs/1606.01540.

Clavera, I.; Nagabandi, A.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2019. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=HyztsoC5Y7.

Jin, C.; Allen-Zhu, Z.; Bubeck, S.; and Jordan, M. I. 2018. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, 4868–4878.

Kakade, S.; and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, 267–274.

Katsikopoulos, K. V.; and Engelbrecht, S. E. 2003. Markov decision processes with delays and asynchronous cost collection. *IEEE transactions on automatic control* 48(4): 568–574.

Kuznetsova, E.; Li, Y.-F.; Ruiz, C.; Zio, E.; Ault, G.; and Bell, K. 2013. Reinforcement learning for microgrid energy management. *Energy* 59: 133–146.

Liu, S.; Wang, X.; and Liu, P. X. 2015. Impact of communication delays on secondary frequency control in an islanded microgrid. *IEEE Transactions on Industrial Electronics* 62(4): 2021–2031.

Mahmood, A. R.; Korenkevych, D.; Komer, B. J.; and Bergstra, J. 2018. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4635–4640. IEEE.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Schuitema, E.; Buşoniu, L.; Babuška, R.; and Jonker, P. 2010. Control delay in reinforcement learning for real-time dynamic systems: a memoryless approach. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3226–3231. IEEE.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Walrand, J.; and Parekh, S. 2010. Communication networks: a concise introduction. *Synthesis Lectures on Communication Networks* 3(1): 1–192.

Walsh, T. J.; Nouri, A.; Li, L.; and Littman, M. L. 2009. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems* 18(1): 83.