Coverage Analysis of Multi-Environment Q-Learning Algorithms for Wireless Network Optimization

Talha Bozkus
Department of Electrical and Computer Engineering
University of Southern California, USA
bozkus@usc.edu

Urbashi Mitra
Department of Electrical and Computer Engineering
University of Southern California, USA
ubli@usc.edu

***Abstract*—Q-learning is widely used to optimize wireless networks with unknown system dynamics. Recent advancements include ensemble multi-environment hybrid Q-learning algorithms, which utilize multiple Q-learning algorithms across structurally related but distinct Markovian environments and outperform existing Q-learning algorithms in terms of accuracy and complexity in large-scale wireless networks. We herein conduct a comprehensive coverage analysis to ensure optimal data coverage conditions for these algorithms. Initially, we establish upper bounds on the expectation and variance of different coverage coefficients. Leveraging these bounds, we present an algorithm for efficient initialization of these algorithms. We test our algorithm on two distinct real-world wireless networks. Numerical simulations show that our algorithm can achieve %50 less policy error and %40 less runtime complexity than state-of-the-art reinforcement learning algorithms. Furthermore, our algorithm exhibits robustness to changes in network settings and parameters. We also numerically validate our theoretical results.**

***Index Terms*—Reinforcement learning, Q-learning, coverage coefficient, wireless networks**

I. Introduction

Markov Decision Processes (MDPs) are useful tools for modeling large-scale wireless networks and solving diverse wireless network optimization problems such as throughput maximization, collision avoidance, and route optimization [1]–[4]. When transition dynamics and cost functions are unknown, traditional methods such as dynamic programming [5] are inapplicable. Instead, *model-free* reinforcement learning (RL) algorithms such as Q-learning [5] can be employed. However, traditional Q-learning faces several challenges, including high computation complexity, high estimation bias and variance, and slow convergence. Consequently, several variants have been proposed to address these issues. These algorithms can be classified based on their estimation strategies, objectives, and implementations.

Similar to the original Q-learning, a single Q-function estimator on a single Markovian environment is employed in [5]–[7]. On the other hand, multiple Q-function estimators on a single Markovian environment are used in [8]–[11], in which each estimator is initialized independently, and their outputs are fused into a single estimate via some weighting mechanism. Recently, novel ensemble Q-learning algorithms that utilize multiple Q-function estimators on multiple distinct but structurally related Markovian environments have been proposed [3], [12]. These algorithms run multiple Q-learning algorithms on multiple Markovian environments. These environments are inherently different but share similar characteristics and dynamics, enabling improved performance with respect to exploration and training [12], [3]. The work of [12] addresses MDPs with arbitrary graphs and presents probabilistic convergence analysis by assuming distributional assumptions on Q-functions. In contrast, [3] leverages MDP structural properties to reduce training complexity with efficient sampling and state-aggregation strategies and provides deterministic stability and convergence analysis. Q-learning algorithms can also be categorized based on their objective. Estimation bias is considered in [8], and the estimation variance and training stability are examined in [9], [11]. The convergence rate is improved in [6], and training data efficiency is considered in [7]. The work of [3], [12] improves the overall exploration across large state-spaces and reduces the training runtime and sample complexity – the number of interactions with the environment to learn optimal policies.

Q-learning algorithms also differ in terms of their implementation and data collection strategies. In online Q-learning [13], [14], the RL agent learns and updates its Q-functions by interacting with the environment in real-time. This approach is suitable for dynamic environments. In contrast, offline Q-learning [15], [16] relies on pre-collected training data and is preferred when direct interaction with the environment is expensive. Recent hybrid versions [3], [12], [17], [18] combine online interaction with offline data for improved performance.

Despite their distinct characteristics, reducing sample complexity remains a crucial consideration across different types of Q-learning algorithms. To this end, the notion of *coverage conditions* plays a fundamental role in determining the sample complexity of RL algorithms [19], [20]. These conditions ensure that the data collection distribution adequately covers the state space. For instance, Fitted Q-Iteration [7] attains an ϵ-optimal policy under the condition that the data distribution uniformly covers all possible induced state distributions.

In this work, we will analyze the coverage conditions of recent multi-environment Q-learning algorithms [3], [12],

This work was funded by the following grants: NSF CCF-1817200, ARO W911NF1910269, DOE DE-SC0021417, Swedish Research Council 2018-04359, NSF CCF-2008927, NSF CCF-2200221, ONR 503400-78050, ONR N00014-15-1-2550, NSF CCF 2200221 and USC + Amazon Center on Secure and Trusted Machine Learning

which have not been well-studied. We aim to improve the accuracy and complexity of these algorithms by leveraging their theoretical coverage limits. Our motivation is to handle complex scenarios in optimizing real-world wireless networks, such as multiple packet arrivals to energy-harvesting transmitters, significant channel quality variations in MIMO networks, simultaneous packet collisions, or buffer overflows due to multiple data arrivals in MISO networks.

The main contributions of the paper are as follows: (i) We adopt a probabilistic approach to derive upper bounds on the expectation and variance of different coverage coefficients (CC) for multi-environment ensemble Q-learning algorithms. To the best of our knowledge, there is no prior work on CC that takes this approach. (ii) We present an algorithm to assess and compare the utilities of different Markovian environments within these algorithms. Unlike prior work [12], our approach offers more accurate, low-complexity, and robust initialization for these algorithms. (iii) We conduct simulations on two real-world large state-space wireless networks, showing that our algorithm reduces policy error by 50% and increases speed by 40% compared to state-of-the-art RL algorithms. We also validate our theoretical findings numerically.

In this paper, vectors are bold lower case (**x**), matrices are bold upper case (**A**), and sets are in calligraphic font ($\mathcal{S}$).

II. System Model and Tools

A. Markov Decision Processes

MDPs are characterized by 4-tuples $\{\mathcal{S}, \mathcal{A}, p, c\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the finite state and action spaces, respectively. We denote s_t as the *state* and a_t as the *action* taken at discrete time t. The transition from s to s' under action a occurs with the probability $p_a(s, s')$, which is stored in the $(s, s', a)^{th}$ element of the probability transition tensor (PTT) $\mathbf{P}$, and a bounded cost $c_a(s)$ is incurred, which is stored in the s^{th} element of the cost vector $\mathbf{c}_a$. We focus on an infinite-horizon discounted cost MDP, where $t = \mathbb{Z}^+ \cup \{0\}$. Our goal is to solve the following optimization problem:

$$\mathbf{v}^*(s) = \min_{\pi} \mathbf{v}_\pi(s) = \min_{\pi} \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t \mathbf{c}_{a_t}(s_t) | s_0 = s \Big], \quad (1)$$

for all $s \in \mathcal{S}, a_t \in \mathcal{A}$, where $\mathbf{v}_\pi$ is the *value function* [5] – the expected cumulative discounted cost incurred starting from state s_0 under the *policy* π, $\mathbf{v}^*$ is the *optimal value function*, and $\gamma \in (0, 1)$ is the discount factor. The *deterministic* policy defines a specific action per state, and the *stationary* policy does not change over time. We herein consider deterministic and stationary policies as they always exist given a finite state and action space [5].

B. Q-Learning

Q-learning seeks to solve the optimization problem and find the optimal policy π^* by learning the Q functions $Q(s, a)$ – the expected cumulative discounted cost of taking action a in state s, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ using the following update rule:

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha (c_a(s) + \gamma \min_{a' \in \mathcal{A}} Q(s', a')), \quad (2)$$

where $\alpha \in (0, 1)$ is the learning rate. The *ϵ-greedy* policies are used to handle the *exploration-exploitation* trade-off: with probability ϵ, a random action is taken (exploration), and with probability 1-ϵ, a greedy action, which minimizes the Q-function of the next state, is chosen (exploitation). This balance is crucial to ensure that sufficient information about the system is captured by visiting each state-action pair sufficiently many times. The agent interacts with the environment and collects samples $\{s, a, s', c\}$ to update Q-functions using (2). The learning strategy must specify the trajectory length (l) (the number of states in a single trajectory) and the minimum number of visits to each state-action pair (v), which is generally used as a termination condition for the sampling operation. Q-functions converge to their optimal values with probability one, *i.e.,* $Q \xrightarrow{w.p.1} Q^*$ if a set of necessary conditions are satisfied [21]. The policy can then be inferred as $\pi^*(s) = \operatorname{argmin}_{a \in \mathcal{A}} Q^*(s, a)$.

C. Coverage Coefficient

Sample complexity is a crucial consideration for different RL algorithms. In online RL, sample-efficient algorithms require exploration conditions to learn a near-optimal policy when interacting with the unknown environment, such as epsilon-greedy, upper confidence level-based exploration, or Thompson sampling [22]. In contrast, sample-efficient algorithms for offline RL require data coverage conditions over the offline dataset for statistical guarantees. The work in [19] shows that the data coverage condition can ensure sample efficiency even in an online setting. The multi-environment Q-learning algorithms [3], [12] combine features of online and offline RL methods; hence, our focus will be to ensure good data coverage conditions.

The *coverage coefficient* quantifies data coverage. There exist different definitions for the coverage coefficient (CC) [20], [22]; we herein use the traditional definition of CC, which is applicable to discrete and finite state-action spaces:

$$C^\pi(s, a) = \frac{d^\pi(s, a)}{v(s, a)}, \quad C^* = \max_{s,a} C^\pi(s, a), \quad (3)$$

where $C^\pi(s, a)$ is the CC for the state-action pair (s, a) under the policy π, $d^\pi(s, a)$ is the likelihood of visiting (s, a) when following the policy π, $v(s, a)$ is the likelihood of visiting (s, a) under some exploratory behavior and C^* is the CC of the overall state-action space. Herein, C^* measures the deviation between the exploration distribution v and the distribution induced by the policy π. If v is close to d^π, it indicates that the policy π spends a comparable amount of time to the exploration of the state-action pair (s, a) by v. Achieving perfect coverage ($C^* = 1$) may not be feasible in practice; however, choosing a good exploration distribution v and optimizing the policy π can make C^* close to 1. In this work, we will leverage the distance between the simulated CC from its optimal value across different environments and policies to optimize multi-environment ensemble Q-learning algorithms.

III. Theoretical Analysis

We use Algorithm 2 of [12] (n-hop ensemble Q-learning = nEQL) as a generic example for our derivations, yet our analysis can be generalized to the algorithm of [3].

A. Assumptions and Preliminaries

Let K be the number of environments in the nEQL algorithm, and n denote the order of the environment ($n = 1, 2, ..., K$). We employ the distributional assumption on the Q-function errors of the n^{th} environment:

$$\mathcal{X}_t^{(n)}(s,a) = Q_t^{(n)}(s,a) - Q^*(s,a) \sim D_n\Big(0, \frac{\lambda_n^2}{3}\Big), \quad (4)$$

where $\mathcal{X}_t^{(n)}$ is the Q-function error of the n^{th} environment at time t, Q^* is the optimal Q-function of the original environment, and $Q_t^{(n)}$ is the Q-function of the n^{th} environment at time t for (s,a), and D_n is some random distribution (with zero-mean and finite variance). This assumption has been commonly employed in [11], [23] for $n = 1$ case with D_n being uniform, non-uniform, or normal. In [3], [12], the assumption is generalized to $n > 1$ case with no assumptions on D_n. We herein present our results for $|\mathcal{A}| = 2$ (see [24] for the generalization to the arbitrary $|\mathcal{A}|$). We assume that the policy π always chooses the action a in (3). (Otherwise, it is the trivial case: $d^\pi = C^\pi = 0$); thus, $C^\pi \in [1, nfty)$. Furthermore, we employ a linear action selection strategy for each different environment:

$$v_t^{(n)}(s,a) = \frac{Q_t^{(n)}(s,a)}{\sum_{k=1}^{|\mathcal{A}|} Q_t^{(n)}(s,a_k)}, \quad (5)$$

where $v_t^{(n)}(s,a)$ is the exploration distribution of (s,a) of the n^{th} environment at time t. (Softmax-action selection can also be employed, although the analysis is more involved [24]) We also assume the following relationship between the Q-functions of the same state under different actions k_1 and k_2:

$$\frac{1}{\theta} \leqslant \frac{Q^*(s,a_{k_1})}{Q^*(s,a_{k_2})} \leqslant \theta, \quad (6)$$

where $\theta > 1$. This parameter incorporates prior knowledge of the Q-functions (if any) or can be estimated numerically.

B. Bounds on the coverage coefficient

Please see [24] for the complete proofs of all propositions.

Proposition 1. Let $\pi = \pi^{(n)}$ (estimated policy of the n^{th} environment in nEQL algorithm) in (3). Then:

$$\mathbb{E}[\log C^{\pi^{(n)}}(s,a)] \leqslant \log(1+\theta) + \frac{\lambda_n^2}{3Q^*(s,a)^2}\Big[\frac{1}{2} - \frac{1}{(1+\theta)^2}\Big].$$
$$\mathbb{V}[\log C^{\pi^{(n)}}(s,a)] \leqslant \frac{\lambda_n^2}{3Q^*(s,a)^2}\Big[1 + \frac{2\theta^2}{(1+\theta)^2} + \frac{2\sqrt{2}\theta}{1+\theta}\Big] \quad (7)$$

These bounds are governed by the estimation error variance λ_n^2, which varies across environments and will be key to optimizing the nEQL algorithm. As λ_n^2 decreases, the expectation bound tightens, and the variance bound approaches 0. The parameter θ provides a trade-off between the tightness of the bounds and the flexibility of the assumptions on the Q-functions (6). For example, the constant term $\log(1+\theta)$ can be made smaller, and bounds can be further tightened by employing a more strict assumption than (6) or employing a different exploration strategy than (5) (see [24] for an example).

Proposition 2. Let $\pi = \hat{\pi}$ (estimated policy of the nEQL algorithm) in (3). Under the assumptions of Section III-A:

$$\mathbb{E}[\log C^{\hat{\pi}}(s,a)] \leqslant \log(1+\theta) + \frac{\lambda^2}{3Q^*(s,a)^2}\Big[\frac{1}{2} - \frac{1}{(1+\theta)^2}\Big]\frac{1-u}{1+u}.$$
$$\mathbb{V}[\log C^{\hat{\pi}}(s,a)] \leqslant \frac{\lambda^2}{3Q^*(s,a)^2}\Big[1 + \frac{2\theta^2}{(1+\theta)^2} + \frac{2\sqrt{2}\theta}{1+\theta}\Big]\frac{1-u}{1+u}. \quad (8)$$

where $u \in (0,1)$ is the update ratio of nEQL and $\lambda = \max_n \lambda_n$ (The parameters of the nEQL algorithm are outlined in [12]). These bounds, akin to those in Proposition 1, depend on the hyper-parameter u, allowing for optimization through fine-tuning. Following a strategy as in [12] (choosing a large u or time-dependent u_t such that $u \xrightarrow{t\to\infty} 1$) tightens both bounds. The environment with the largest estimation error variance (*the worst environment*) governs both bounds; thus, optimizing only the Q-learning of the worst environment is an efficient way of improving the performance of the nEQL algorithm.

Proposition 3. The bounds in Proposition 2 can be refined to depend on K as follows:

$$\mathbb{E}[\log C^{\hat{\pi}}(s,a)] \leqslant \log(1+\theta) + \frac{1}{Q^*(s,a)^2}\Big[\frac{1}{2} - \frac{1}{(1+\theta)^2}\Big]\frac{f(\lambda,u)}{K}.$$
$$\mathbb{V}[\log C^{\hat{\pi}}(s,a)] \leqslant \frac{1}{Q^*(s,a)^2}\Big[1 + \frac{2\theta^2}{(1+\theta)^2} + \frac{2\sqrt{2}\theta}{1+\theta}\Big]\frac{f(\lambda,u)}{K} \quad (9)$$

where f is some function of λ and u. As K increases, the expectation bound tightens, and the variance bound approaches 0, which aligns with the variance-reduction goal of traditional ensemble algorithms. These bounds can also be optimized by fine-tuning u, function f, or parameter θ.

C. Comparing the bounds of different environments

The bounds presented in the previous section all rely on the distributional parameters λ_n (and λ). Thus, understanding the relationship between these parameters can be used to improve the accuracy and complexity of nEQL. Proposition 4 of [12] provides a partial ordering that helps compare estimation error variances in certain environments such as $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_4$, $\lambda_1 \geqslant \lambda_3$ or $\lambda_1 \geqslant \lambda_5$. However, it provides no information regarding λ_2 vs λ_3 or λ_3 vs λ_5. Hence, identifying the optimal set of environments may require exhaustive search, which can be computationally expensive for large K. Herein, we present several results and an algorithm to compare estimation error variances between any two environments that will be used to determine the optimal set of environments in the nEQL algorithm.

Proposition 4. The previous bounds on the expectation and variance are tightest for $n = 1$ and $\lambda_1 = \min_n \lambda_n$. Additionally, λ_n is non-monotonic across n for $n > 1$.

Algorithm 1 Coverage-based ensemble Q-learning (CCQ)

Inputs: MDP model $(\mathcal{S}, \mathcal{A}, \hat{p}, \hat{c})$, K, K_{total}, γ, u, α
Outputs: $\hat{Q}$, $\hat{\pi}$

1: Estimate c_{min}, c_{max} using estimated cost functions $(\hat{c})$
2: Use Proposition 5 to sort λ_n for $n = 1, 2, ..., K_{total}$ in increasing order.
3: Pick the first K environments with the tightest bounds on the coverage coefficient using Proposition 1 ($K \ll K_{total}$)
4: Run nEQL [12] with the K environments (the other hyper-parameters are optimized as in [12]).
5: Output estimated Q-functions $\hat{Q}$ and estimated policy $\hat{\pi}$

This proposition suggests that the original environment, with the smallest estimation error variance (*the best environment*), should always be included in the nEQL algorithm as it is most likely to achieve CC close to 1. For $n > 1$, the non-monotonicity of λ_n poses challenges in optimizing nEQL, necessitating a strategy to determine the relationship between the λ_n. This result aligns with Proposition 4 of [12].

Proposition 5. Let λ_n and λ_m be the estimation error variance of two distinct environments. Let the cost function in the underlying MDP be bounded as $c_a(s) \in [c_{min}, c_{max}]$ with $c_{min} > 0$ and $c_{max} < \infty$. Then, the following rule can be employed to compare λ_n and λ_m:

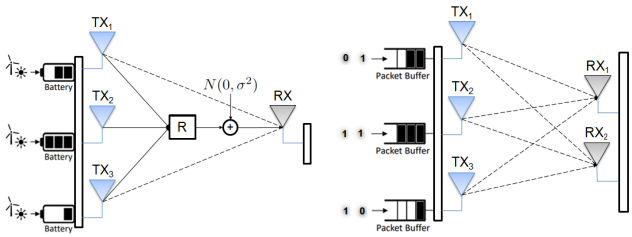
$$\begin{aligned} &\lambda_n < \lambda_m \quad \text{if} \quad f(\gamma, n, m) > \alpha \frac{c_{\max}}{c_{\min}} + (1-\alpha)\frac{c_{\min}}{c_{\max}}. \\ &\lambda_n > \lambda_m \quad \text{otherwise}, \end{aligned} \tag{10}$$

where $f(\gamma, n, m) = \frac{(1-\gamma^n)(1-\gamma^{m-1})}{(1-\gamma^m)(1-\gamma^{n-1})}$, γ is the common discount factor and $\alpha \in (0, 1)$ is the weight factor. This proposition gives complete ordering between all environments (as opposed to the partial ordering of [12]). This strategy needs at most $\binom{K}{2}$ comparisons to order all K environments. Herein, α is used to determine the decision boundary (see [24] for details). It can be randomly chosen between (0,1) or can be fine-tuned to minimize the misordering error.

After establishing the bounds on CC and a strategy to order the estimation error variances of different environments, we give the main algorithm of this work in Algorithm 1. The inputs are the MDP model with $\mathcal{S}, \mathcal{A}$ and estimated transition probabilities $(\hat{p})$ and cost functions $(\hat{c})$ (see [12] for estimation details), the number of environments (K) to initialize the nEQL algorithm out of K_{total} environments, the update ratio of nEQL (u), the discount factor (γ) and a weight factor (α). The outputs are the ensemble Q-function output of nEQL $(\hat{Q})$ and corresponding estimated policy $\hat{\boldsymbol{\pi}}$. Please refer to [12] for the details of nEQL. The parameter K_{total} is chosen moderately large and increases as a function of the network size as explained in [12] and $K \ll K_{total}$. While exact bounds in Propositions 1, 2, 3 require the optimal Q-functions (Q^*), we can order the environments without knowing optimal Q-functions as Q^* is a constant with respect to n.

IV. Numerical Results

We consider two distinct wireless network models that vary in their topology, scale, objective, and implementations:



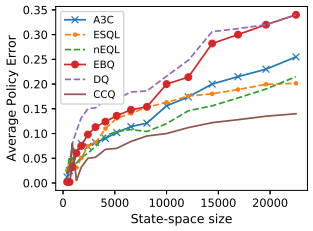
(a) MISO energy harvesting wireless network with multiple relays (b) MIMO wireless network

Fig. 1: Examples wireless network models.

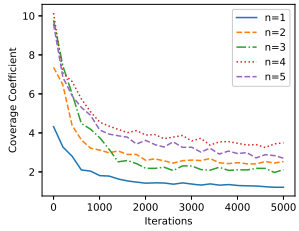
(i) MISO energy harvesting wireless network with multiple relays [2], (ii) MIMO wireless network [3]. Example wireless networks are shown in Fig. 1. We use the following numerical parameters: $|\mathcal{S}| = 10000$, $|\mathcal{A}| = 2$, $K = 5$, $K_{total} = 10$, $u = 0.5$, $\gamma = 0.95$, $\alpha \sim \text{unif}(0, 1)$, $(s, a) = (6, 1)$. The parameter θ is numerically estimated and ranges in $[1.04, 1.26]$ for different settings. In the wireless networks, the n^{th} environments consider n packet changes at a time – transmitting or receiving n energy/data packets, and the corresponding PTTs describe the n-step transition probabilities.

For MISO network, Algorithm 1 gives the following set in increasing order: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_6, \lambda_5, \lambda_4, \lambda_7, \lambda_9, \lambda_8, \lambda_{10}\}$. In contrast, the partial ordering approach of [12] gives us the following three set of orders: $\{\lambda_1, \lambda_2, \lambda_4, \lambda_8\}$, $\{\lambda_1, \lambda_3, \lambda_6\}$ and $\{\lambda_1, \lambda_5, \lambda_{10}\}$ each with no preference. Extensive simulations yield the following true ordering: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_5, \lambda_6, \lambda_4, \lambda_7, \lambda_9, \lambda_8, \lambda_{10}\}$. Observe that Algorithm 1 gives the correct set of environments for $K = 5$ $(1, 2, 3, 5, 6)$, yet the partial ordering is mostly inconclusive. We repeat the simulation using the following parameters: $|\mathcal{S}|$ $\in \{500, 1000, ..., 40000\}$, $\mathcal{A} \in$ [2,6], $K_{total} \in$ [10,20], $K \in$ [2,10] using both MISO and MIMO networks, the number of transmitters, receivers and relays $\in [2, 8]$. Overall, Algorithm 1 gives us 82% accuracy in finding the set of optimal environments, which shows the robustness of the algorithm to the changes in network parameters. Algorithm 1 is 45% faster than the nEQL algorithm with partial ordering [12] (with exhaustive search where the partial ordering is inconclusive).

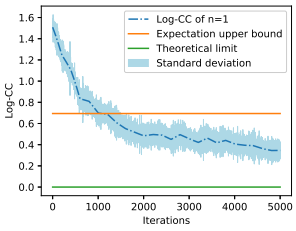
We also evaluate the performance of Algorithm 1 using MIMO network using Average Policy Error (APE), defined as $\frac{1}{|\mathcal{S}|}\sum_{s=1}^{|\mathcal{S}|} \mathbf{1}(\boldsymbol{\pi}^*(s) \neq \hat{\boldsymbol{\pi}}(s))$, where π^* is the optimal policy of the original environment as defined in Section II-B. For comparison, we employ the following Q-learning algorithms: (i) n-hop Ensemble Q-Learning (nEQL) [12], (ii) Ensemble Synthetic Q-Learning (ESQL) [3], (iii) Asynchronous Advantage Actor Critic (A3C) [25], (iv) Ensemble Bootstrapping Q (EBQ) [10], and (v) Double Q (DQ) [8]. These algorithms differ in their objective, estimation strategy, and data collection strategy and provide a fair comparison. The APE results vs state-space size is shown in Fig.2a. The proposed algorithm can reduce the APE of nEQL by %50 and also achieves %40 less APE than the other algorithms across large state-spaces, which shows the benefits of initializing the algorithm with the optimal set of environments.



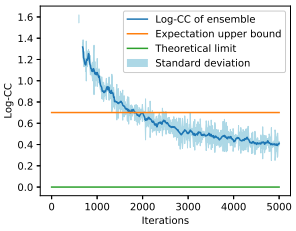
(a) APE of different algorithms across increasing state-space size



(b) $C^{\pi^{(n)}}(6,1)$ vs environment order n across iterations



(c) $\log C^{\pi^{(1)}}(6,1)$ and upper bound from Proposition 1



(d) $\log C^{\hat{\pi}}(4,0)$ and upper bound from Proposition 2

Fig. 2: Numerical simulations

We give the CC over time across different orders (n = 1,2,3,4,5) for $(s,a) = (6,1)$ in Fig.2b using the MIMO network with $\theta = 1.15$. The CC varies non-monotonically across n, with the original environment having the smallest CC, followed by the 3rd, 2nd, 5th, and 4th environments. The estimation error variances of the environments from Proposition 5 and the numerically estimated variances follow the same order as the CCs ($\lambda_1 \leqslant \lambda_3 \leqslant \lambda_2 \leqslant \lambda_5 \leqslant \lambda_4$). This suggests that estimation error variances can effectively order CCs. The log-CC under the estimated policy of the original environment ($\pi^{(1)}$) for $(6,1)$, along with expectation bound from Proposition 1, is shown in Fig. 2c. The upper bound is useful when the algorithm converges, and its tightness changes with (s,a) and n. Fig. 2d illustrates similar trends for $(s,a) = (4,0)$ under the policy of Algorithm 1 ($\hat{\pi}$) with the expectation bound from Proposition 2.

V. Conclusions

We provided a detailed analysis of the data coverage conditions of recently proposed multi-environment ensemble Q-learning algorithms [3], [12] for optimizing a variety of real-world wireless networks. We provided upper bounds on the expectation and variance of different coverage coefficients and explain how to interpret them. Exploiting these bounds, we presented an algorithm to provide an efficient and accurate way of initializing these algorithms by ordering the utilities of different environments. Our simulations on two different wireless networks showed that we could reduce the policy error by %50 and increase the speed by %40 of the state-of-the-art prior work. We also verified our theoretical assumptions.

References

[1] Jie Wang, Talha Bozkus, Yao Xie, and Urbashi Mitra. Reliable adaptive recoding for batched network coding with burst-noise channels. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pages 220–224. IEEE, 2023.

[2] Talha Bozkus and Urbashi Mitra. Link analysis for solving multiple-access mdps with large state spaces. *IEEE Transactions on Signal Processing*, 71:947–962, 2023.

[3] Talha Bozkus and Urbashi Mitra. Leveraging digital cousins for ensemble q-learning in large-scale wireless networks. *IEEE Transactions on Signal Processing*, 72:1114–1129, 2024.

[4] Talha Bozkus and Urbashi Mitra. A novel ensemble q-learning algorithm for policy optimization in large-scale networks. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pages 1381–1386. IEEE, 2023.

[5] Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[6] M. G. Azar, R. Munos, M. Ghavamzadaeh, and H. J Kappen. Speedy q-learning. 2011.

[7] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[8] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[9] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pages 176–185. PMLR, 2017.

[10] Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *International Conference on Machine Learning*, pages 8454–8463. PMLR, 2021.

[11] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. *CoRR*, abs/2101.05982, 2021.

[12] Talha Bozkus and Urbashi Mitra. Multi-timescale ensemble q-learning for markov decision process policy optimization. *IEEE Transactions on Signal Processing*, 72:1427–1442, 2024.

[13] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[14] Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34:7193–7206, 2021.

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[16] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[17] Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

[18] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[19] Tengyang Xie, Dylan J Foster, Yu Bai, Nan Jiang, and Sham M Kakade. The role of coverage in online reinforcement learning. *arXiv preprint arXiv:2210.04157*, 2022.

[20] Masatoshi Uehara and Wen Sun. Pessimistic model-based offline reinforcement learning under partial coverage. *arXiv preprint arXiv:2107.06226*, 2021.

[21] Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

[22] Fanghui Liu, Luca Viano, and Volkan Cevher. What can online reinforcement learning with function approximation benefit from general coverage conditions? In *International Conference on Machine Learning*, pages 22063–22091. PMLR, 2023.

[23] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, pages 1–9, 1993.

[24] Talha Bozkus and Urbashi Mitra. Supplementary appendix file. https://github.com/talhabozkus/spawc-24-supplementary-material, 2024.

[25] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.