Multi-Timescale Ensemble Q-learning for Markov Decision Process Policy Optimization

Talha Bozkus and Urbashi Mitra

***Abstract*—Reinforcement learning is a classical tool to solve network control or policy optimization problems in unknown environments. The original Q-learning suffers from performance and complexity challenges across very large networks. Herein, a novel model-free ensemble reinforcement learning algorithm which adapts the classical Q-learning is proposed to handle these challenges for networks which admit Markov decision process (MDP) models. Multiple Q-learning algorithms are run on multiple, distinct, synthetically created and structurally related Markovian environments in parallel; the outputs are fused using an adaptive weighting mechanism based on the Jensen-Shannon divergence (JSD) to obtain an approximately optimal policy with low complexity. The theoretical justification of the algorithm, including the convergence of key statistics and Q-functions are provided. Numerical results across several network models show that the proposed algorithm can achieve up to 55% less average policy error with up to 40% less runtime complexity than the state-of-the-art Q-learning algorithms. Numerical results validate assumptions made in the theoretical analysis.**

***Index Terms*—Markov decision process (MDP), network optimization, ensemble learning, reinforcement learning, Q-learning**

I. Introduction

Markov Decision Processes (MDPs) are natural mathematical tools for modeling sequential decision-making problems in many large real-world networks [1]–[3]. When the underlying system dynamics are observable, the optimization problem of MDPs can be solved by dynamic programming [4]. However, these algorithms are not directly applicable to problems where the underlying MDP is unknown (or non-observable), in which case *model-free* Reinforcement Learning (RL) algorithms such as Q-learning can be employed to simulate the system dynamics and learn the policies and value functions [5].

Q-learning can be employed to solve a variety of optimization and control problems in unknown environments [6]–[9]. However, it suffers from several performance and complexity challenges across large MDPs, including high estimation bias and estimation variance, training instability, slow convergence, and high sample complexity. To this end, several variants of Q-learning have been developed to handle these challenges. Estimation bias is considered in [10], [11], and the estimation variance and training stability are examined in [12], [13]. The convergence rate is improved in [14], and training data efficiency is considered in [15]. In [16], [17], strategies specific to wireless networks are considered. As these algorithms have different objectives, their strategies and implementations also differ. Similar to the original Q-learning, a single Q-function estimator is employed in [14], [15]. On the other hand, multiple Q-function estimators are used in [10], [12], [13], [18], [19], in which each estimator is initialized independently, and their outputs are fused into a single estimate via a weighting mechanism. These algorithms directly operate on the original Markovian environment; however, Q-learning algorithms using multiple Q-function estimators on **multiple** Markovian environments have not been well-studied. We will see that the performance and complexity of Q-learning can be further improved by employing multiple Q-function estimators on multiple structurally related synthetic Markovian environments operating at different time-scales.

Despite extensive prior work [20], [21], achieving efficient and scalable exploration in large Markovian environments remains a major challenge in reinforcement learning. *Too little exploration* may cause the agent to behave greedily. Consequently, some parts of the environment may never be visited, and the agent may keep selecting sub-optimal actions and get stuck in a local optimum. On the other hand, *too much exploration* may yield a high accumulated cost, preventing the utilization of previous experiences, and be computationally expensive for very large environments. While the previous work generally employs a *single* efficient exploration strategy [20], [21], we herein propose a **two-level exploration** strategy, *i.e.* there are two different sources of exploration: at the *algorithm-level*, we use one of the existing exploration techniques (such as epsilon-greedy Q-learning [5]), and at the *environment-level*, we utilize multiple, distinct, synthetically created and structurally related Markovian environments, which provide different orders of relationships between states. As we consider products of the probability transition matrix of the original system to construct our synthetic systems, we deem our approach as having **multiple time scales**, corresponding to the n-hop transition matrices of the Markov chain. Our goal is to improve the exploration capabilities of the agent and accelerate the exploration stage of Q-learning.

To this end, we propose a novel ensemble Q-learning algorithm, where *multiple* Q-learning algorithms are run in parallel on *multiple* distinct, synthetically created and structurally related Markovian environments. Their outputs are fused into a single Q-function estimate using an adaptive weighting mechanism based on a Jensen–Shannon divergence between the distributions corresponding to the Q-functions of different environments. In the end, an approximately optimal deterministic policy is obtained with low complexity. In our

Talha Bozkus and Urbashi Mitra are with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, USA. Email: {bozkus, ubli}@usc.edu.

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 and USC + Amazon Center on Secure and Trusted Machine Learning

prior work [22], [23], we introduced similar algorithms and presented preliminary findings. That initial analysis yielded significant insights into the advantages of employing multiple Markovian environments to improve the accuracy and complexity of the original Q-learning. Herein, we provide significant improvements over that preliminary work: (i) We propose a more interpretable and computationally cheaper way to construct multiple synthetic Markovian environments. (ii) We remove the constraints and approximations on the structure of the network models; hence, our current design is applicable to a wider range of networks. (iii) We provide a more complete theoretical analysis of the proposed algorithm. (iv) We utilize a new distance metric based on the JSD to improve the accuracy of the adaptive weighting mechanism.

The main contributions of the paper are as follows: (i) We systematically construct the multiple synthetic Markovian environments to enable an efficient and scalable exploration in Q-learning. (ii) We propose a novel Q-learning algorithm based on an ensemble of the Q-functions from multiple Markovian environments. (iii) We provide theoretical analyses on the convergence and error variance of the proposed algorithm. (iv) We simulate the algorithm on several large real-world network classes. Numerical results show that the proposed method outperforms the state-of-the-art Q-learning algorithms on all networks, achieving up to 55% less policy error with up to 40% less runtime complexity. In addition, simulations confirm the theoretical analyses.

We use the following notation: the vectors are bold lower case ($\mathbf{x}$), matrices and tensors are bold upper case ($\mathbf{A}$), and sets are in calligraphic font ($\mathcal{S}$).

II. System Model and Tools

A. Infinite Horizon Discounted Cost MDP model

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 period t. The transition from state s to s' under action a occurs with probability $p_a(s,s') = p(s' = s_{t+1} \mid s = s_t, a = a_t)$, which is stored in the $(s,s',a)^{th}$ element of the three-dimensional probability transition tensor (PTT) $\mathbf{P}$, and a bounded average cost $c_a(s) = \sum_{s' \in \mathcal{S}} p_a(s,s')\hat{c}_a(s,s')$ is incurred, which is stored in the $(s,a)^{th}$ element of the cost matrix $\mathbf{C}$, where $\hat{c}_a(s,s')$ is the instantaneous transition cost from state s to s' under action a. We denote the probability transition matrix (PTM) and cost vector under the action a by $\mathbf{P}_a$ and $\mathbf{c}_a$, respectively. We focus on infinite horizon discounted cost MDPs, where $t = \mathbb{Z}^+ \cup \{0\}$. Our goal is to solve *Bellman's optimality* equation:

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

$$\pi^*(s) = \operatorname*{argmin}_{\pi} \mathbf{v}_\pi(s), \quad (2)$$

for all $s \in \mathcal{S}$, where $\mathbf{v}_\pi$ is the *value function* [4] under the *policy* π, $\mathbf{v}^*$ is the *optimal value function*, π^* is the *optimal policy*, and $\gamma \in (0,1)$ is the discount factor. The policy π can define either a specific action per state (*deterministic*) or a distribution over the action space per state (*stochastic*) for each time period. If the policy does not change over time, *i.e.,* $\pi_t = \pi$, $\forall t$, then it is deemed *stationary*. There always exists a deterministic stationary policy that is optimal given a finite state and action spaces [4]. Hence, we, herein, consider deterministic and stationary policies.

B. Q-Learning

When the system dynamics (p and c) are unknown or non-observable, Q-learning can be used to solve (1) and (2). Q-learning seeks to find the optimal policy π^* by learning the Q functions for all (s,a) pairs using the following update rule:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(c_a(s) + \gamma \min_{a' \in \mathcal{A}} Q(s',a')), \quad (3)$$

where $\alpha \in (0,1)$ is the learning rate. In practice, *ϵ-greedy* policies are used to tackle the *exploration-exploitation* trade-off to ensure that sufficient sampling of the system is captured by visiting each state-action pair sufficiently many times [5]. To this end, a random action is taken with probability ϵ (exploration), and a greedy action that minimizes the Q-function of the next state is taken with probability $1-\epsilon$ (exploitation). The agent interacts with the environment and collects samples $\{s,a,s',c\}$ to update Q-functions using (3). The learning strategy must specify the trajectory length (l) (the number of states in a 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.,* $Q(s,a) \xrightarrow{w.p.1} Q^*(s,a)$ for all (s,a) if necessary conditions are satisfied [24]. The optimal policy and value functions can be inferred from the Q-functions as follows:

$$\pi^*(s) = \operatorname*{argmin}_{a \in \mathcal{A}} Q^*(s,a), \quad \mathbf{v}^*(s) = \min_{a \in \mathcal{A}} Q^*(s,a). \quad (4)$$

Q-learning algorithms using multiple Q-function estimators on the original Markovian environment [10], [12] are shown to improve the performance and complexity of the original Q-learning. In contrast, Q-learning algorithms using *multiple* Q-function estimators on *multiple* Markovian environments have not been well-studied. We shall see that sampling and training can be accelerated and made data efficient; furthermore, more accurate and stable Q-functions can be obtained by combining the capabilities of multiple environments. We next explain how to create these multiple Markovian environments.

C. Creating Multiple Markovian Environments

There are several ways to create multiple environments (and the corresponding PTTs) based on the PTT of the original environment $\mathbf{P}$. A natural strategy is to employ some function of $\mathbf{P}$ (or $\mathbf{P}^T$). In particular, the probability $p_a(s,s')$ should be related across different environments, and the PTTs of different environments should be row-stochastic (*i.e.* each PTM corresponding to a different action is row-stochastic) or can be converted into row-stochastic environments by employing appropriate normalization without changing the original structure. We herein propose to use n-hop PTTs ($\mathbf{P}^n$) because (i) they describe the n-step transition probabilities between

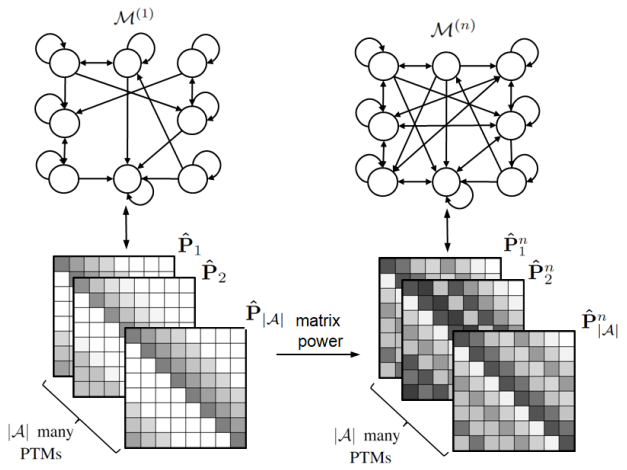


Fig. 1: The relationship between $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(n)}$.

states; hence, they are interpretable, corresponding to multiple time-scales, (ii) they are easily computable using efficient matrix multiplication methods, (iii) they are inherently row-stochastic, and (iv) they lead to a nice mathematical analysis (as will be shown later).

There are various factors that suggest employing n-hop Markovian environments could improve the exploration capabilities of the overall system in several ways: (i) They enable the agent to traverse longer trajectories and uncover new state-action pairs beyond its immediate reach, potentially expediting the agent's understanding of the environment with fewer interactions. (ii) They enable the agent to learn from indirect experiences by simulating trajectories that are not directly observed. (iii) They can encourage the agent to consider longer-term rewards and take actions that may not have immediate rewards, leading to better long-term performance, particularly in environments with sparse rewards or long-term dependencies. (iv) They enable the agent to exploit environment patterns by exploring longer trajectories that uncover complex relationships between actions and outcomes, which can be particularly valuable in structured or repetitive environments such as mazes or puzzles, where the agent must learn to identify and leverage patterns to achieve its objectives. (v) They can help the agent to better handle environments with changing dynamics by enabling it to learn from past experiences that may no longer be directly relevant to the current state of the environment.

The underlying PTT, $\mathbf{P}$, is initially unknown; thus, it needs to be estimated to create the PTTs of multiple environments. If the number of one-step transitions between states is known (or can be counted), then the transition probabilities can be estimated via *sample averaging* [25]. Moreover, if there is a particular structure in the one-step transition probabilities, then more data-efficient estimation can be performed [22], [26]. We denote the estimated PTT by $\hat{\mathbf{P}}$. We then construct the n-hop PTTs using $\hat{\mathbf{P}}$. Each different $\hat{\mathbf{P}}^n$ $(n > 1)$ inherently corresponds to a different Markovian environment, called a *synthetic Markovian environment* and denoted by $\mathcal{M}^{(n)}$. The relationship between the original Markovian environment $\mathcal{M}^{(1)}$ and the synthetic Markovian environments (SME) $\mathcal{M}^{(n)}$ $(n > 1)$ is given in Fig.1, where $\hat{\mathbf{P}}_i$ denotes the PTM for the i^{th} action in $\hat{\mathbf{P}}$ for $i \in \{1, 2, ..., |\mathcal{A}|\}$. Thus, $\hat{\mathbf{P}}_i^n$ represents the n-hop probability transitions for action i – we take action i over n time steps.

Algorithm 1 n-hop Ensemble Q-Learning (nEQL)

Input: l, v, u_t, K, $\mathbf{Q}^{(n)}, n \in \{1, 2, ..., K\}$
Output: $\mathbf{Q}^{it}$, $\hat{\boldsymbol{\pi}}$

Initialize $\mathbf{w}_0$ randomly, $\mathbf{Q}_0^{it} \leftarrow \mathbf{0}$, $t \leftarrow 0$
while each (s, a) pair in $\mathcal{M}^{(1)}$ not visited v times **do**
  choose common initial state for all $\mathcal{M}^{(n)}$ randomly
  repeat l **times**
    for each $n \in \{1, ..., K\}$ **do**
      sample $\{s, a, s', c\}$ from $\mathcal{M}^{(n)}$ and update $\mathbf{Q}_t^{(n)}$ using (3)
      convert $\mathbf{Q}_t^{(n)}$ into probabilities $\hat{\mathbf{Q}}_t^{(n)}$ state-wise using the negative softmax
      $\mathbf{w}_t^{(n)} \leftarrow 1 - \mathrm{AJSD}(\hat{\mathbf{Q}}_t^{(1)} \| \hat{\mathbf{Q}}_t^{(n)})$
    end for
    $\mathbf{w}_t \leftarrow softmax(\mathbf{w}_t)$
    $\mathbf{Q}_{t+1}^{it} \leftarrow u_t \mathbf{Q}_t^{it} + (1 - u_t) \sum_{n=1}^{K} \mathbf{w}_t^{(n)} \mathbf{Q}_t^{(n)}$
    $t \leftarrow t + 1$
  end
end while
$\hat{\boldsymbol{\pi}}(s) \leftarrow \mathrm{argmin}_{a'} \mathbf{Q}^{it}(s, a')$

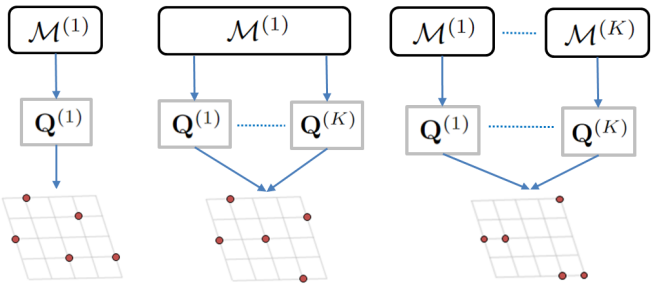
Other approaches to estimate $\mathbf{P}$ include function approximations and approximate maximum likelihood estimation techniques [27]–[29]. However, these approaches have drawbacks: (i) Non-linear function approximations such as neural networks lack interpretability. (ii) They do not appear to offer significant computational advantages over sample averaging in sparse Markovian environments. (iii) They generally do not exploit the structural properties of the system. (iv) Training and parameter optimization can be computationally challenging.

The estimation quality affects the accuracy of $\hat{\mathbf{P}}^n$ (and thus $\mathcal{M}^{(n)}$) differently for each n. As n increases, the error from imperfect sampling accumulates due to matrix multiplications. Consequently, higher-order environments may have low accuracy if sampling in the original environment is not done sufficiently. This suggests that n should not be chosen very large for practical purposes, as we shall see in the numerical results.

III. nEQL Algorithm and Analysis

In this section, we present the n-hop Ensemble Q-Learning (nEQL) algorithm (Algorithm 1). It is a model-free algorithm since the system dynamics, including transition probabilities and costs, are unknown. Our approach utilizes $K - 1$ SMEs ($\mathcal{M}^{(n)}$ for $n \in 2, ..., K$) in addition to the original Markovian environment ($\mathcal{M}^{(1)}$), resulting in a total of K Markovian environments. The high-level comparison between the original Q-learning algorithm, conventional ensemble Q-learning algorithms and proposed Q-learning algorithm is shown in Fig.2, where $\mathbf{Q}^{(n)}$ represents the Q-function estimator of the Q-learning run on $\mathcal{M}^{(n)}$ for $n \in [1, K]$.

The inputs to Algorithm 1 consist of the trajectory length (l), the minimum number of visit requirement to each state-action



(a) Original QL (b) Ensemble QL (c) Proposed QL

Fig. 2: Classification of Q-Learning (QL) algorithms based on their strategies and implementation.

pair (v), the update ratio at time t ($u_t \in [0,1]$), the total number of Markovian environments (K), and the empty Q-tables for K different environments ($\mathbf{Q}^{(n)}$ for $n \in \{1,2,...,K\}$). Let $\mathbf{w}_t$ be the weight vector of size K at time t (with $\mathbf{w}_t^{(n)}$ being the n^{th} element of $\mathbf{w}_t$). The weight vector at $t=0$ ($\mathbf{w}_0$) is initialized randomly to break the symmetry in line 1 (*i.e.* each element is chosen randomly from [0,1], and the vector is softmax-normalized so that $\sum_{n=1}^{K} \mathbf{w}_0^{(n)} = 1$). The iterations continue until each state-action pair in $\mathcal{M}^{(1)}$ is visited at least v times (in line 2) to ensure that different state-action dynamics are sufficiently represented.

At the end of each trajectory (*i.e.* every l time step), all K Markovian environments are reset, and a common initial state is assigned randomly from $\{1,2,...,|\mathcal{S}|\}$, as indicated in line 3. In line 6, independent samples are collected from each different Markovian environment, and corresponding Q-tables are updated independently. We emphasize that given the common initial state, different actions are taken following the epsilon-greedy policy of each different environment. As a result, different next state and cost pairs are observed for different environments. This procedure is repeated l times, after which a random but common initial state is set. In line 7, the Q-functions are converted into probabilities per state using the negative softmax function. For example, if the Q-functions for a given state are [1, 1.4, 0.8, 2] (assuming four actions), we input the negative of the Q-functions ([-1, -1.4, -0.8, -2]) to softmax function, and compute the corresponding probabilities as [0.31, 0.21, 0.37, 0.11]. Recall that we are doing cost *minimization*; thus, the smaller Q-function is more likely to correspond to the optimal action. In line 8, we compute the distance between the probability distributions $\hat{\mathbf{Q}}_t^{(1)}$ and $\hat{\mathbf{Q}}_t^{(n)}$ using the *averaged Jensen-Shannon divergence* (AJSD) defined as follows:

$$\text{AJSD}(\hat{\mathbf{Q}}_t^{(1)} \| \hat{\mathbf{Q}}_t^{(n)}) = \frac{1}{|s|} \sum_{s} \text{JSD}(\hat{\mathbf{Q}}_t^{(1)}(s,:) \| \hat{\mathbf{Q}}_t^{(n)}(s,:)), \quad (5)$$

where $\hat{\mathbf{Q}}_t^{(n)}(s,:)$ is the probability vector of size $|\mathcal{A}|$, and JSD between probability distributions p and q is defined as [30]:

$$\text{JSD}(p,q) = \frac{1}{2}\Big[\text{KL}\left(p \| \frac{p+q}{2}\right) + \text{KL}\left(q \| \frac{p+q}{2}\right)\Big], \quad (6)$$

where KL denotes the Kullback–Leibler divergence using base 2. Although there are several distance measures, including l_2 distance between $\mathbf{Q}_t^{(1)}$ and $\mathbf{Q}_t^{(n)}$ or KL divergence between $\hat{\mathbf{Q}}_t^{(1)}$ and $\hat{\mathbf{Q}}_t^{(n)}$, we employ JSD because (i) JSD is a symmetric measure in contrast to the KL divergence. (ii) JSD is a smoothed and bounded version of KL divergence (bounded to [0,1] and hence AJSD is also bounded to [0,1]); thus, it is robust to noise, outliers, or small perturbations in the Q-functions. (iii) Numerical results show that it provides superior performance to the other measures (see [31]). Herein, a larger $\mathbf{w}_t^{(n)}$ implies that the two sets of probabilities ($\hat{\mathbf{Q}}_t^{(1)}$ versus $\hat{\mathbf{Q}}_t^{(n)}$) are closer, so are the corresponding Q-functions ($\mathbf{Q}_t^{(1)}$ versus $\mathbf{Q}_t^{(n)}$). The vector $\mathbf{w}_t$ is softmax-normalized in line 10 and used to update the Q-function output of Algorithm 1, $\mathbf{Q}_t^{it}$, in line 11. When updating $\mathbf{Q}_t^{it}$, previous experience are exploited by utilizing fraction u_t of the $\mathbf{Q}_t^{it}$ from the previous iteration (*exploitation*), while multiple Markovian environments are sampled based on their weights (*exploration*), and their contributions are weighted by $1-u_t$. In the end, the estimated policy $\hat{\boldsymbol{\pi}}$ is obtained from $\mathbf{Q}^{it}$ in line 15.

The $\mathbf{Q}_t^{it}$ (*iterative*) is updated adaptively using the current weight vector $\mathbf{w}_t$; hence, it captures *asymmetric* information between different Markovian environments, *i.e.*, how the utility of samples obtained from $\mathcal{M}^{(n)}$ may change during iterations. For example, it is likely that $\mathcal{M}^{(1)}$ provides more useful samples at the beginning as it is the original environment, and there are not enough samples to capture the higher-order relationships versus the first-order relationships. On the other hand, $\mathcal{M}^{(n)}$ for larger n contributes more as the iterations increase as the first-order relationships may not aid as much in exploration.

The weights in Algorithm 1 ($\mathbf{w}_t^{(n)}, n \in \{1,2,...,K\}$) converge due to the fact that the distinct Q-functions converge to their optimal values via Q-learning [32], and the weights are calculated based on the attendant Q-functions. We will also verify the convergence of the weights numerically.

Our proposed algorithm combines features of online and offline RL methods. The PTT of the original environment is estimated and the corresponding Q-functions are updated in real-time, which is the *online* part. The PTTs of the multiple SMEs are constructed, and the corresponding Q-functions are learned using pre-collected data (i.e., the estimated PTT of the original environment), which is the *offline* part. It is important to note that our approach is **different** from hybrid RL [33]–[35], where the agent generally has access to an offline dataset and subsequently collects new data through interacting with the environment. On the other hand, our algorithm *continuously updates* the Q-functions in real-time while also leveraging pre-collected data to produce more accurate and stable Q-functions with low complexity.

A. Theoretical analysis

In this section, we provide several theoretical results for Algorithm 1. Assume the Q-function **errors** follow a distribution D with zero mean and finite variance as follows: [1]

$$\mathcal{X}_t^{(n)}(s,a) \stackrel{\text{def}}{=} \mathbf{Q}_t^{(n)}(s,a) - \mathbf{Q}^*(s,a) \sim D\Big(0, \frac{\lambda_n^2}{3}\Big), \quad (7)$$

[1] We observe that one can construct **small** state-space examples that do not adhere to this assumption; however, for the large scale examples we consider herein, numerical results suggest that the assumption is valid (see Fig.8b and Fig.8e).

for all (s, a) and n with $\lambda_n > 0$ where $\mathbf{Q}^*$ is the optimal Q-functions of the original Markovian environment. Prior work has considered the distribution D to be uniform, non-uniform, or normal for the $n = 1$ case [13], [36]–[38]. Herein, we make no assumptions on D; however, we do assume that a common distribution family governs the Q-function errors of all environments. This assumption is reasonable and practical because (i) all Markovian environments have the same state and action space, and the transition dynamics are related through n-hop relationships, (ii) the same cost function is employed; thus, there is a single objective function across all Q-learning algorithms, and (iii) the same parameters govern all Q-learning algorithms (such as learning rate and trajectory length). Simulations verify that the true distribution D is, in fact, very close to the normal distribution with zero-mean and finite variance for all n (see Fig.8b and Fig.8e).

Let $\mathbb{E}$ and $\mathbb{V}$ be the expectation and variance operators, $\lambda = \max_{n\in\{1,2,\ldots,K\}} \lambda_n$, and $\mathcal{E}_t(s,a) \overset{\text{def}}{=} \mathbf{Q}_t^{it}(s,a) - \mathbf{Q}^*(s,a)$.

Proposition 1. Let u_t be a constant: $u_t = u$. Under Assumption (7), Algorithm 1 produces unbiased Q-functions in the limit *i.e.* $\lim_{t\to\infty} \mathbb{E}[\mathcal{E}_t(s,a)] = 0$. If the Q-function errors of a given environment at different times are independent *i.e.* $\mathcal{X}_{t_1}^{(n)}(s,a) \perp \mathcal{X}_{t_2}^{(n)}(s,a)$ for all s, a, n, $t_1 \neq t_2$, the error variance in the limit can be upper bounded as: $\lim_{t\to\infty} \mathbb{V}[\mathcal{E}_t(s,a)] \leqslant \frac{(1-u)}{(1+u)}\lambda^2$. (see Appendix A)

This proposition shows that under the assumption (7), $\mathbf{Q}_t^{it}$ is an unbiased estimator of $\mathbf{Q}^*$ in the limit, and the upper bound on the error variance can be controlled by the parameters u and λ. Herein, a larger λ implies a higher uncertainty in the Q-function errors, which makes the upper bound on the variance looser. On the other hand, when the algorithm converges ($t \to \infty$), a larger u leads to less reliance on SMEs, reducing the uncertainty arising from multiple environments and yielding a tighter upper bound.

Corollary 1. If we remove the independence assumption in Proposition 1, the upper bound on the error variance in the limit can be updated as: $\lim_{t\to\infty} \mathbb{V}[\mathcal{E}_t(s,a)] \leqslant \frac{2\lambda^2}{(1+u)^2} + \frac{(1-u)}{(1+u)}\lambda^2$. (see [31])

The relaxation of the independence assumption introduces a bias term to the upper bound estimation, resulting in a looser bound than the one in Proposition 1. Nevertheless, a smaller λ or a larger u tightens the bound as in Proposition 1.

Corollary 2. If we use the form $u_t = 1 - e^{\frac{-t}{c_4}}$ with $c_4 > 0$ and the independence assumption in Proposition 1, the error variance converges to zero: $\lim_{t\to\infty} \mathbb{V}[\mathcal{E}_t(s,a)] = 0$.

We choose the parameter u_t such that $u_t \xrightarrow{t\to\infty} 1$ as u should be small initially to explore multiple environments in the beginning (exploration) and should increase to utilize previously obtained samples with time (exploitation *i.e.* less reliance on the synthetic environments as we learn $\mathbf{Q}_t^{it}$ better). Herein, using multiple Markovian environments accelerates the convergence of $\mathbf{Q}_t^{it}$ towards $\mathbf{Q}^*$, while adjusting u_t pushes $\mathbf{Q}_t^{it}$ in the desired direction. Consequently, Algorithm 1 converges to the optimal Q-functions: $\mathbf{Q}_t^{it} \xrightarrow{t\to\infty} \mathbf{Q}^*$. We will numerically verify that the independence assumption is **almost always** satisfied. The parameter c_4 adjusts the decay rate of u_t, which is crucial to tune the amount of exploration. In particular, a larger c_4 implies a slower decay rate, which is needed for larger networks, where more exploration is necessary. This result aligns with Proposition 1 and Corollary 1, as larger values of u lead to tighter upper bounds on the error variance and $u = 1$ makes the upper bounds zero.

The convergence proof of the algorithm in our prior work [23] can be adapted to show the convergence of the Algorithm 1 deterministically without any distribution or independence assumption on the Q-function errors.

Proposition 2. The upper bound on the error variance decreases with the number of Markovian environments K as $\mathbb{V}[\mathcal{E}_t(s,a)] \leqslant \frac{c(\lambda,u)}{K}$ for all t, where $c(\lambda, u)$ is a constant of K. (see Appendix B)

The proposition shows that increasing the number of Markovian environments (Q-learning algorithms running on different environments simultaneously) reduces the upper bound on variance. This aligns with the primary objective of ensemble algorithms. Unlike the bound in Proposition 1, this upper bound explicitly depends on K, is valid without any assumptions on independence or structure of u_t, and holds for all t. The decay rate of the upper bound is determined by the function $c(\lambda, u)$, which incorporates both λ and u. Furthermore, numerical results will show that increasing K may not always yield an increasing reduction in the upper bound, *i.e.* there is a diminishing return of increasing K.

Proposition 3. Let the output policy of Algorithm 1 be $\hat{\pi}$, and the Q-functions of the n^{th} environment under the policy $\hat{\pi}$ be $\mathbf{Q}_{\hat{\pi}}^{(n)}$. If the spectral norm of the PTM of the original Markovian environment under $\hat{\pi}$ is upper bounded by 1 *i.e.* $\|\mathbf{P}_{\hat{\pi}}\| < 1$, Algorithm 1 produces Q-functions on different environments that satisfy the following upper bound:

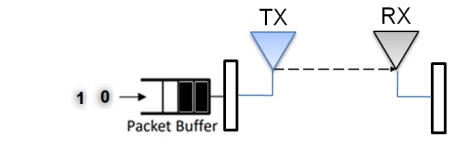
$$\|\mathbf{Q}_{\hat{\pi}}^{(1)} - \mathbf{Q}_{\hat{\pi}}^{(n)}\| < \frac{\gamma}{1-\gamma^n}\frac{1-\gamma^{n-1}}{1-\gamma}\|\mathbf{c}_{\hat{\pi}}\|, \tag{8}$$

where $n > 1$, the norm $\|\cdot\|$ is the spectral norm for matrices, and l_2 norm for vectors. (see Appendix C)

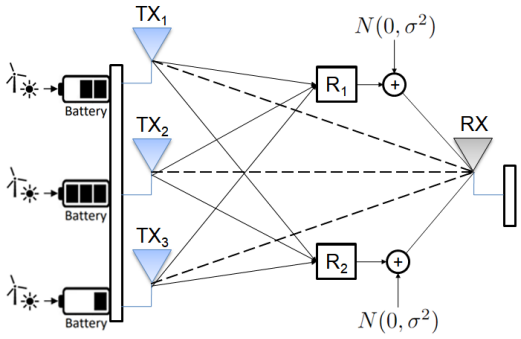
This proposition shows that the Q-functions of different environments under the same policy are closely related as a result of the structural relationship between different Markovian environments. As $n \to \infty$, the upper bound primarily depends on the cost function, eliminating the influence of the learning parameter γ. This suggests n should not be chosen very large for practical purposes. Note that this behavior should not imply the monotonicity of the Q-functions as a function of n, as will be seen later, *i.e.* $\|\mathbf{Q}_{\hat{\pi}}^{(1)} - \mathbf{Q}_{\hat{\pi}}^{(n)}\|$ is a non-monotonic function of n. It is also worth emphasizing that this result holds without any assumptions on the independence or structure of u_t.

Proposition 4. The $\mathbf{Q}_{\hat{\pi}}^{(n)}$ satisfy the following (partial) ordering when $\gamma \to 1$ (*i.e.* when the underlying discounted MDP starts to resemble an undiscounted MDP): (see Appendix D)

$$\mathbf{Q}_{\hat{\pi}}^{(n)}(s) \geqslant \mathbf{Q}_{\hat{\pi}}^{(2n)}(s) \geqslant \mathbf{Q}_{\hat{\pi}}^{(4n)}(s) \geqslant \mathbf{Q}_{\hat{\pi}}^{(8n)}(s)..., \tag{9}$$



(a) SISO network model



(b) MISO network energy harvesting model with relays

Fig. 3: Examples of wireless network models.

for all s, where $\mathbf{Q}_{\hat{\pi}}^{(1)}(s)$ is the largest and n is an odd number.

This proposition provides partial ordering between the utilities of different environments based on the corresponding Q-functions. For example the 2^{nd} order environment is *more useful* than the 4^{th} order environment as it produces Q-functions that are closer to the original Q-functions (*i.e.* $|\mathbf{Q}_{\hat{\pi}}^{(1)}(s) - \mathbf{Q}_{\hat{\pi}}^{(2)}(s)| \leqslant |\mathbf{Q}_{\hat{\pi}}^{(1)} - \mathbf{Q}_{\hat{\pi}}^{(4)}|$. This idea is intuitive as the 4-hop node relationships are implicitly contained in the 2-hop node relationships. However, no conclusion can be inferred between the 2^{nd} and 3^{rd} order relationships. This result is valid without any independence or structural assumption on u_t.

This result offers an efficient method for partially optimizing n and K without requiring an exhaustive search. In scenarios with limited computational resources, the total number of Markovian environments and the selection of initial environments can be based on the given partial ordering. This ensures that only the most useful environments are utilized in the algorithm, allowing for effective resource allocation. Moreover, this result is particularly useful for network settings when long-term planning is more important, for example, when the future costs are more important than the immediate costs or taking some actions may not minimize the immediate rewards, but they will be more beneficial in the long run.

IV. Numerical Results

In this section, we consider a variety of performance metrics to assess the accuracy and complexity performance of the proposed algorithm across different network models.

A. Network models

We consider four different network models, which differ in their design, complexity, and implementation. (See [31] for further details)

1) Randomized graphs: We consider the Erdős-Rényi (ER) random-directed graph model. The PTM has $|\mathcal{S}|$ nodes, and each edge is created with a probability 0.2. We create $|\mathcal{A}|$ many PTMs and concatenate them to obtain the PTT, which is used for sampling and creating the SMEs. The cost function assigns a uniform $[0, 1]$ random cost to each state-action pair.

2) Cliff-walking environment: We consider the cliff-walking environment [5]. The number of columns is chosen to be approximately three times the number of rows in the grid so that the number of states is equal to $|\mathcal{S}|$ (for example, 60 columns and 20 rows $\rightarrow$ 1800 states). If the agent moves to the cliff region, a unit cost is incurred. Moving to the safe grid results in a negative unit cost, while any other movement incurs a cost of 0.01.

3) SISO wireless network model: We consider the model of [17], in which there is a single transmitter (TX) and receiver (RX) as shown in Fig.3a. The goal is to determine when the transmitter should *transmit* data or *remain silent* to minimize the sum of transmission and packet drop costs.

4) MISO energy harvesting wireless network with Gaussian interference channels and multiple relays: We consider the model of [26]. An example network with three transmitters (TX_1, TX_2, TX_3), a single receiver (RX), and two relays (R_1, R_2) is shown in Fig.3b. The goal is to determine when transmitters should *directly* transmit or transmit *through relays* in order to maximize the overall throughput while minimizing the sum of battery and packet drop costs for each transmitter.

B. Average Policy Error Results

Let $\boldsymbol{\pi}^*$ be the optimal policy from (2), and $\hat{\boldsymbol{\pi}}$ be the output policy of Algorithm 1. Since our main concern is optimal control, we define the *average policy error (APE)* as follows:

$$APE = \frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \mathbf{1}(\boldsymbol{\pi}^*(s) \neq \hat{\boldsymbol{\pi}}(s)). \tag{10}$$

We analyze the performance of Algorithm 1 over Q-learning in Fig.4. The simulation is carried out using model-4 with network size 5000 with the following parameters: $\gamma = 0.95$, α_t=$\frac{1}{1+\frac{t}{100}}$, ϵ_t=$\min(0.99^t, 0.01)$, $u_t = 1 - e^{\frac{-t}{1000}}$, $v = 40$, $l = 10$, $K = 4$. These parameters are optimized through cross-validation (see Section IV-F for further details). The curves represent APE of Q-learning algorithms on the original environment $\mathcal{M}^{(1)}$ and three different SMEs $\mathcal{M}^{(n)}$ for $n = 2, 3, 4$, while Q^{it} represents the APE of Algorithm 1. Clearly, a near-zero APE (around 0.05) can be achieved with a significantly small number of iterations (around 700). The sharp decline in the Q^{it} curve at the beginning (up to the 200^{th} iteration) corresponds to the *exploration* stage, followed by the *exploitation* stage. Compared to any other algorithm, the exploration stage in Algorithm 1 is fast, which shows the advantages of utilizing multiple Markovian environments to enable a deep and efficient exploration and accelerate the overall training. Notice that individual Q-learning algorithms can only achieve a slightly higher APE than that of Algorithm 1 if they run for a significantly long time. We observe that the APE results are not monotonic across n (APE of Q-learning

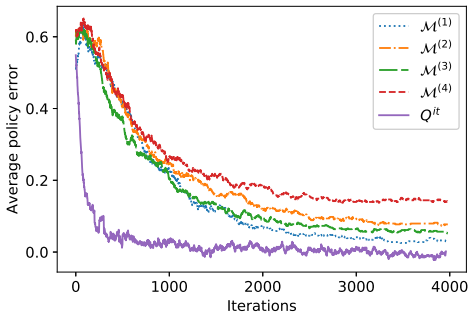


Fig. 4: APE performances across different environments.

Algorithm	Objective	Strategy	
		Estimator	Environment
Simple Q (Q) [4]	-	Single	Single
Speedy Q (SQ) [14]	Convergence rate	Single	Single
Double Q (DQ) [10]	Bias	Multi	Single
MaxMin Q (MMQ) [18]	Bias & variance	Multi	Single
Ensemble Bootst. Q (EBQ) [19]	Bias	Multi	Single
Averaged DQN (ADQN) [12]	Stability, Variance	Multi	Single
n-hop Ensemble Q (nEQL)	**Variance, Learning speed**	**Multi**	**Multi**

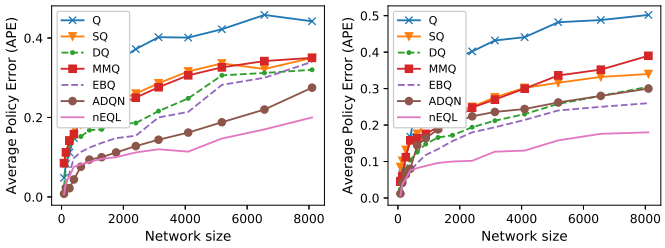
TABLE I: Q-learning algorithm and variants.

run on $\mathcal{M}^{(2)}$ is lower than that of $\mathcal{M}^{(4)}$), which is in line with Proposition 4.

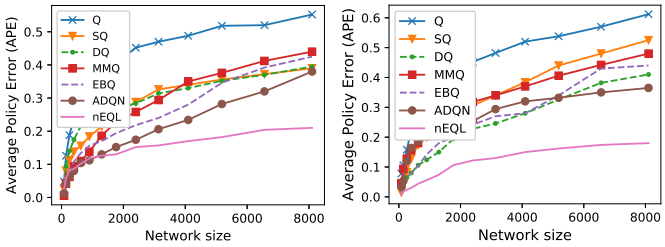
In order to provide a performance comparison, we employ several Q-learning algorithms, each with different objectives and strategies (number of estimators). Table I provides an overview of these algorithms. Specifically, we focus on value-based model-free algorithms which follow the same strategy as Algorithm 1 to ensure a fair comparison. For further details regarding the parameter optimization of each algorithm, refer to [31].

The APE of different algorithms across network size and different models are given in Fig.5a-5d. Overall, our proposed algorithm consistently achieves a lower APE compared to other algorithms: 25% less for model-1, 30% less for model-2, 45% less for model-3, and 55% less for model-4. The APE gains become more clear for larger networks as using multiple Markovian environments enables deep and efficient exploration by combining the n-hop relationships between states into a single estimate, and the weighting mechanism based on JSD enables the algorithm fully exploit the most useful environments during training by assigning higher weights.

The proposed algorithm demonstrates significant APE gains, particularly with model-3 and model-4, showing its effectiveness and practicality across real-world networks. In general, ADQN and EBQ produce the lowest APE among other algorithms; however, they have inferior performance compared to nEQL because neural network-based algorithms (ADQN) do not leverage the structural properties of multiple Markovian environments. The performance of EBQ shows the advantages of ensemble algorithms. However, it does not employ a weighting mechanism when calculating the ensemble but uses simple averaging; thus, it performs worse than nEQL, which employs an adaptive weighting mechanism. We also observe that as the model complexity increases (from model-1 to model-4), the APE of all algorithms increases. The proposed algorithm,



(a) APE for model-1 (b) APE for model-2



(c) APE for model-3 (d) APE for model-4

Fig. 5: APE results across different network models.

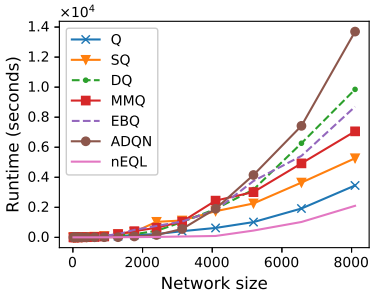


Fig. 6: Runtimes for different algorithms.

however, yields the least increase in APE, making it the most accurate algorithm. The APE order of the other algorithms remains consistent across different models.

C. Computational Runtime Complexity Results

The algorithm 1 can be shown to have the following average time-complexity $O\left(\frac{|\mathcal{S}||\mathcal{A}|v}{K} f(l,\epsilon)\right)$, where f is some non-monotonic function of l and ϵ. The derivation closely follows that presented in [22]. The runtime complexity increases with the network size ($|\mathcal{S}|$ and $|\mathcal{A}|$) as well as the number of visit requirement to each state-action pair (v). On the other hand, the non-monotonicity of f in l, ϵ implies that there are optimal values for l, ϵ; thus, parameter-tuning is required. The complexity is also inversely proportional to the number of Markovian environments (K), which may seem counter-intuitive. However, the number of samples that need to be collected from each Markovian environment decreases with K, leading to an overall reduction in the training runtime complexity.

The runtime of nEQL includes time spent on sampling (visiting each state-action pair at least v times), constructing multiple SMEs $K-1$ times using matrix power operations, and the time until convergence of Q-learning algorithms on

different Markovian environments. The overall runtime is determined by the slowest Q-learning algorithm, as they are run in parallel. Other algorithms' runtime results represent the time until each algorithm converges. This runtime measure also serves as a measure of *computational power complexity*. Across fixed network sizes and four different models, the runtime of algorithms is comparable. Therefore, a single runtime result is presented in Fig.6. The proposed algorithm achieves 40% less runtime than the other algorithms across large networks, which can be attributed to several reasons: (i) Utilizing multiple Markovian environments reduces the need for long trajectories to capture distant node relationships. (ii) The amount of exploration is improved due to the running of multiple Markovian environments simultaneously. (iii) The algorithm can be terminated early with minimal APE as shown in Fig.4. (iv) The average runtime complexity decreases with K hyperbolically. The complexity reduction is independent of the network model or underlying assumptions, making the proposed algorithm an efficient approach for learning various complex environments.

The original Q-learning has a relatively lower runtime than the other benchmarks as it is a straightforward algorithm. SQ aims to improve the learning (and convergence) rate; therefore, it also produces a relatively lower runtime. On the other hand, EBQ has multiple estimators, while ADQN suffers from a long training period, which becomes computationally intractable for large networks. It is worth noting that as the model complexity increases (e.g., Q → NQ → ADQN), the corresponding APE generally decreases, but the runtime complexity increases. However, nEQL achieves a small APE with a small runtime, overcoming the performance-complexity trade-off.

To solve (1) and (2), one could consider using dynamic programming methods such as value iteration instead of Q-learning, given that we estimate the PTTs of multiple Markovian environments using $\hat{\mathbf{P}}^n$. We herein follow a similar iterative procedure to Algorithm 1. While we keep sampling to update $\hat{\mathbf{P}}$ continuously, we construct the PTTs of multiple SMEs using the current estimate of $\hat{\mathbf{P}}$ every 10 iterations, and obtain the value functions $\mathbf{v}_t^{(n)}$ for $n \in \{1, 2, ..., K\}$ using the value iteration algorithm. We then assign the weights $\mathbf{w}_t^{(n)}$ as:

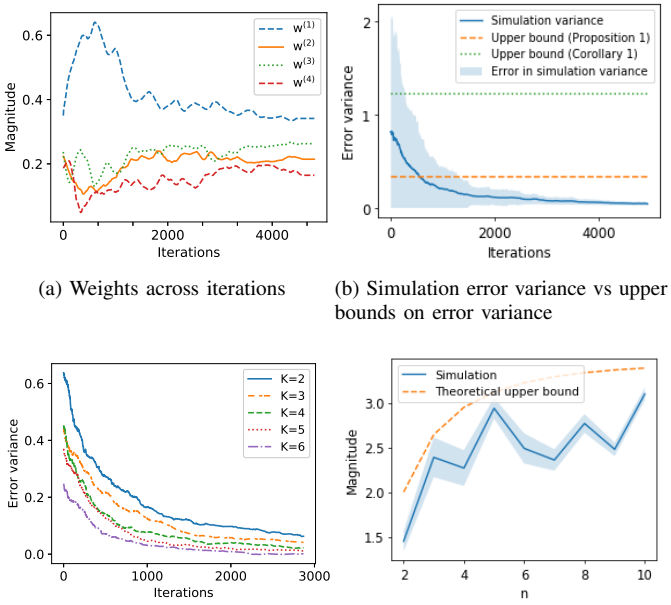
$$\mathbf{w}_t^{(n)} \leftarrow -\|\mathbf{v}_t^{(1)} - \mathbf{v}_t^{(n)}\|_2, \ \forall n \tag{11}$$

$$\mathbf{w}_t \leftarrow \text{softmax}(\mathbf{w}_t), \tag{12}$$

We finally update the value function of the proposed algorithm $\mathbf{v}_t^{it}$ similar to Algorithm 1 using u_t (with the same form as in Corollary 2). We carry out these operations iteratively k times, where k is sufficiently large to ensure the convergence. This approach has runtime-complexity $O(k|\mathcal{S}|^3)$ since we carry out value iteration k times, which has a complexity of $O(|\mathcal{S}|^3)$. In our simulations, with the settings in Section IV-B, this strategy incurs 50% more runtime complexity and 60% more APE across large networks, making it inferior to our proposed algorithm.

D. Convergence of weights

We herein demonstrate the changing weights over time using the simulation settings from Section IV-B in Fig.7a. There



(a) Weights across iterations

(b) Simulation error variance vs upper bounds on error variance

(c) Simulation error variance vs K

(d) Q-function norm difference vs n

Fig. 7: Numerical validation of assumptions for theoretical results.

is a sharp increase in $\mathbf{w}^{(1)}$ at the beginning because $\mathcal{M}^{(1)}$ is the original environment, and there are not enough samples to capture the higher-order relationships. Moreover, it is not clear which $\mathcal{M}^{(n)}$ provides the most useful samples as the weights $\mathbf{w}^{(n)}$ for $n > 1$ keep changing. As iterations continue, $\mathbf{w}^{(1)}$ decreases up to some point, but $\mathcal{M}^{(1)}$ remains the most useful environment. The weights $\mathbf{w}^{(n)}$ for $n > 1$ increase and converge to a fixed value. The final magnitudes of $\mathbf{w}^{(n)}$ are non-monotonic across n ($\mathbf{w}^{(1)} > \mathbf{w}^{(3)} > \mathbf{w}^{(2)} > \mathbf{w}^{(4)}$), validating the partial ordering of Proposition 4. We note that similar weight patterns can be also shown across different network parameters and models, although (i) the final values of the weights, (ii) the order of environment utilities, and (iii) the iteration index at which the weights converge may vary.

E. Numerical validation of propositions

In this section, we simulate the results in the propositions and compare theoretical results with simulation results. The same simulation settings in Section IV-B are employed.

We compute the simulation variance of the Q-function errors numerically as follows:

$$\mathbb{V}[\mathcal{E}_t(s,a)] \approx \frac{1}{2\Delta_t} \sum_{t'=t-\Delta_t}^{t+\Delta_t} \mathcal{E}_t(s,a)^2 - \Big[\frac{1}{2\Delta_t} \sum_{t'=t-\Delta_t}^{t+\Delta_t} \mathcal{E}_t(s,a)\Big]^2, \tag{13}$$

with $\Delta_t \ll t$. The upper bounds on the variance from Proposition 1 and Corollary 1 (with $u_t = 0.5$ and $\lambda = 1$) and the simulation variance (with $(s,a) = (6,2)$ and $\Delta_t = 20$) are shown in Fig.7b, where the blue-shaded region represents the simulation error. As iterations continue, the simulation variance becomes smaller than the upper bound from Proposition 1; hence, using a time-varying u_t produces more accurate

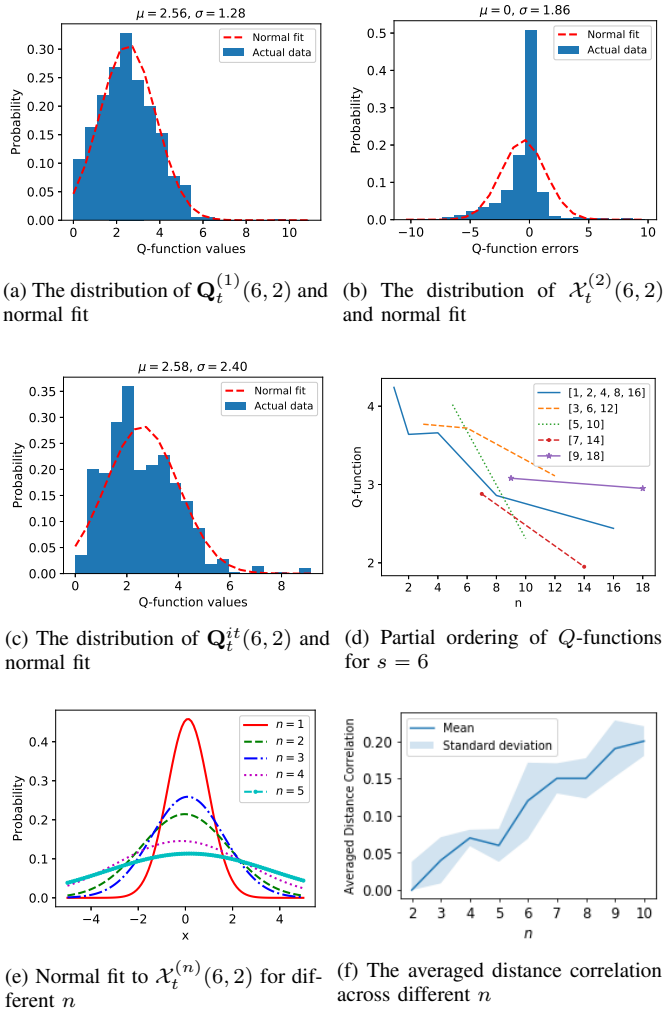
results. The simulation variance eventually converges to zero, which is in line with Corollary 2. Herein, λ is chosen to be 1, and changing λ may affect the initial error variance and its decay rate.

The simulation error variances across different K are shown in Fig.7c. For each K, the simulation variance reduces and converges to zero eventually. The convergence may require more than 3000 iterations, particularly with small K, due to slower convergence as indicated by the theoretical runtime complexity in Section IV-C. Hence, small non-zero error variances are observed at $t = 3000$. At a fixed iteration index, the simulation variance also reduces as K increases, which is in line with Proposition 2. We also observe the diminishing return of increasing K. In particular, the reduction in error variance from K systems to $K + 1$ systems becomes less significant as K increases. This implies that increasing K may not always yield substantial improvements because increasing K results in new environments that exhibit high structural similarity with those of lower; the corresponding PTTs converge to fixed tensors. Consequently, sampling from these new environments may not provide novel samples. Furthermore, a larger K leads to a reduction in the weights assigned to existing environments, which diminishes the impact of each individual environment on the ensemble estimate.

Both the upper bound on the l_2 norm difference between the Q-function vectors of the original environment and n^{th} environment under the output policy of Algorithm 1 as a function of $n \in [2, 10]$ (from Proposition 3) and the actual norm difference is illustrated in Fig.7d, where the blue curve is the mean, and the blue shaded area is the standard deviation of the results over 20 simulations. The norm difference, which represents the utility of the n^{th} order environment, exhibits a non-monotonic behavior across different n, and the results align with Proposition 4. We also observe that the upper bound converges as n increases, yet the tightness of the bound depends on n.

The estimated distributions of $\mathbf{Q}_t^{(1)}(s, a)$, $\mathcal{X}_t^{(n)}(s, a)$ and $\mathbf{Q}_t^{it}(s, a)$ over time (until convergence) for $(s, a, n) = (6, 2, 2)$ are shown in Fig.8a, Fig.8b and Fig.8c, respectively. These distributions can be accurately modeled using normal distributions with corresponding means and variances. Furthermore, we observe that $\mathcal{X}_t^{(2)}(6, 2)$ has zero-mean, which shows the accuracy of the zero-mean assumption (7). Similarly, Fig.8e, which shows the normal fitting of Fig. 8c duplicated for various values of n, demonstrates that the Q-function errors of different environments can be well-modeled by zero-mean normal distributions with different variances. This figure further validates the assumption (7). We note that similar results can also be shown for different (s, a) pairs.

The Q-function of $s = 6$ of the n^{th} environment under the output policy of Algorithm 1 is shown in Fig.8d for $n \in [1, 20]$. The simulation is carried out with $\gamma = 1 - 10^{-5}$. We observe that partial orderings given in Proposition 4 hold. For example we have $\mathbf{Q}_{\hat{\pi}}^{(1)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(2)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(4)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(8)}(6)$ and $\mathbf{Q}_{\hat{\pi}}^{(3)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(6)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(12)}(6)$ but the relationships between the Q-functions of different partial ordering groups are non-monotonic across n such as $\mathbf{Q}_{\hat{\pi}}^{(6)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(4)}(6)$ and



(a) The distribution of $\mathbf{Q}_t^{(1)}(6, 2)$ and normal fit

(b) The distribution of $\mathcal{X}_t^{(2)}(6, 2)$ and normal fit

(c) The distribution of $\mathbf{Q}_t^{it}(6, 2)$ and normal fit

(d) Partial ordering of Q-functions for $s = 6$

(e) Normal fit to $\mathcal{X}_t^{(n)}(6, 2)$ for different n

(f) The averaged distance correlation across different n

Fig. 8: Numerical validation of assumptions in theoretical results.

$\mathbf{Q}_{\hat{\pi}}^{(7)}(6) \geqslant \mathbf{Q}_{\hat{\pi}}^{(16)}(6)$. Extensive simulations demonstrate that similar orderings hold for all states s.

To assess the practicality of the assumption employed in Proposition 1, we compute the averaged distance correlation (ADC) [39] between $\mathcal{X}_{t_1}^{(n)}(6, 2)$ and $\mathcal{X}_{t_2}^{(n)}(6, 2)$ with $n \in [2, 10]$ and averaging the results over all $t_1 \neq t_2 \in [0, 5000]$. ADC is particularly used because it captures both linear and non-linear correlations, is robust to outliers, and does not assume any particular distribution of the variables like Pearson's correlation coefficient. The results are shown in Fig.8f. For example, for $n = 2$, the ADC is almost 0, indicating that the Q-function errors of the 2^{nd} environment at different times are independent. When n is modestly large, the ADC is sufficiently small to infer that the correlation is not statistically significant, and the independence assumption holds. However, for very large values of n, weak correlations may emerge, suggesting that very large values of n might not provide the best performance.

There are several reasons that explain the independence of the Q-function errors at different times: (i) Each environment is Markovian; hence, the errors in Q-function estimates can be considered independent. (ii) Each environment is stationary

(with constant transition probabilities and reward distributions over time), and the errors in Q-function estimates can be assumed independent across different time steps. (iii) The epsilon-greedy policy introduces randomness during the action selection process. (iv) The time-varying learning rate (α_t), epsilon probability (ϵ_t) and the update ratio (u_t) introduces time-dependent randomness and variability.

F. Parameter Tuning

The parameters of Algorithm 1 require fine-tuning across various settings. We discuss how to select $v, l, K, \alpha_t, \epsilon_t$ for small ($|\mathcal{S}| \leqslant 10^3$), modest-sized ($|\mathcal{S}| \in [10^3, 10^4]$), and large networks ($|\mathcal{S}| \geqslant 10^4$). Table II summarizes the parameters that yield near-optimal APE values in our numerical results. However, optimal parameter selection requires cross-validation.

The number of visits (v) needed for each state-action pair is independent of network size. The proposed algorithm allows a small v, around $v \approx 40$, to achieve near-optimal performance. Additionally, the proposed algorithm enables us to have short trajectories (l) while ensuring: (i) near-optimal APE performance, (ii) minimized runtime and computations per trajectory, (iii) sufficient capture of samples from neighboring states, (iv) preservation of initial state importance despite discounting in long trajectories, and (v) prevention of following redundant paths and loops that provide no new samples.

Increasing the number of Markovian environments (K) reduces the average runtime complexity of the algorithm. However, there are drawbacks to consider: (i) emphasizing high-order node relationships may lead to the loss of low-order node relationships, (ii) $\hat{\mathbf{P}}^{(n)}$ converges to a fixed tensor as n increases, rendering samples from corresponding SMEs redundant and potentially degrading performance, (iii) a diminishing return is observed in the error variance (as shown in Fig.7c), and (iv) memory requirements increase. Hence, K should be small enough to avoid these drawbacks, while also increasing with the network size to prevent a rise in other parameters (specifically v, l) and reduce runtime complexity.

The learning rate (α_t) must adhere to the convergence conditions of Q-learning [24] and have a suitable decay to adjust the learning speed. We assume the form $\alpha_t = \frac{1}{1+{}^{t}/_{c_1}}$, where $c_1 > 0$ determines the decay rate and should increase with $|\mathcal{S}|$, v, l, and K. On the other hand, the parameter ϵ_t is essential for balancing exploration and exploitation. We use the form $\epsilon_t = \max((c_2)^t, c_3)$, where $c_2 > 0$ adjusts the decay rate, and $0 < c_3 \ll 1$ determines the minimum exploration probability. As the system parameters increase, c_2 should also increase to ensure sufficient exploration. Furthermore, c_3 should be small and positive, allowing for exploration with a low probability when the policy is nearly converged.

G. Memory complexity

The memory needs of the proposed algorithm increase with K as the Q-functions of K different Markovian environments need to be stored in tables. To alleviate this challenge, several strategies can be employed, including a linear and non-linear approximation of Q-functions or state aggregation methods [40]–[43]. In our prior work [22], we proposed a special state-action aggregation algorithm to handle the increasing memory complexity of a model-free learning algorithm, similar to Algorithm 1, where Q-functions of state-action pairs that incur the same cost are grouped into a single Q-function. The algorithm is applicable to networks that have bounded and smoothly changing cost functions (as in model-2, 3, and 4) and can significantly reduce the memory needs while negligibly increasing the APE.

Params	Small networks	Modest-sized networks	Large networks
l	$l \in [1, 5]$	$l \in [5, 10]$	$l \in [10, 20]$
K	$K \in \{2, 3\}$	$K \in \{3, 4, 5\}$	$K \in \{5, 6, 7, 8\}$
α_t	$c_1 \in \{10^2, 5 \cdot 10^2\}$	$c_1 \in \{10^2, 10^3\}$	$c_1 \in \{10^3, 10^4\}$
ϵ_t	$c_2 \in \{0.9, 0.95\}$ $c_3 \in \{0.01, 0.1\}$	$c_2 \in \{0.95, 0.99\}$ $c_3 \in \{0.01, 0.05\}$	$c_2 \in \{0.99, 0.999\}$ $c_3 \in \{0.005, 0.01\}$
u_t	$c_4 \in \{10^2, 5 \cdot 10^2\}$	$c_4 \in \{10^2, 10^3\}$	$c_4 \in \{5 \cdot 10^3, 10^4\}$

TABLE II: Optimized hyper-parameter values of Algorithm 1.

We note that more sophisticated function approximation algorithms can also be designed to work with any kind of network; however, they require the design of optimal basis functions, selection of appropriate features, or design of aggregation schemes, which are out of the scope of this paper.

V. Conclusions

In this paper, we presented a novel ensemble Q-learning algorithm to overcome the performance and complexity challenges of the original Q-learning across large MDPs. The proposed algorithm employs multiple Q-function estimators on multiple, distinct, synthetically created, and structurally related Markovian environments that run at different time-scales and fuses the outputs into a single estimate based on an adaptive weighting mechanism using Jensen-Shannon divergence. Extensive simulations across a variety of real-world networks show that the proposed algorithm produces a near-optimal policy with significantly lower complexity and outperforms other Q-learning algorithms in terms of accuracy and complexity. Several theoretical upper bounds on the error variance are given, and it is shown that the error variance decreases with the number of Markovian environments. An upper bound on the difference between the Q-functions of different environments is given, and the distance between Q-functions is characterized via Jensen-Shannon divergence. In the end, it is shown that the simulation results closely follow the theoretical results. Our analysis provides initial insights and justifications for the advantages of leveraging multiple synthetic Markovian environments in ensemble reinforcement learning. However, there are areas for further improvement and understanding of the nEQL algorithm. Specifically, our ongoing work focuses on devising computationally efficient sampling methods to ensure good data coverage (as described in [44]) across all Markovian environments. Additionally, we are exploring alternative approaches to construct synthetic systems that go beyond relying on n-hop environments to further improve the generalizability and practicality of our algorithm.

VI. Acknowledgements

The authors thank Prof. Tara Javidi of UCSD for ongoing discussions regarding this work and for suggesting that we consider multi-hop transition matrices based on our work on co-link approximations [26].

Appendix

A. Proof of Proposition 1

The following expressions are valid for all (s,a) pairs; hence, we drop the (s,a) notation for simplicity. We first prove the expectation.

$$\lim_{t\to\infty} \mathcal{E}_t = \lim_{t\to\infty} \mathbf{Q}_t^{it} - \mathbf{Q}^*. \tag{14}$$

$$= \lim_{t\to\infty} (1-u) \sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbf{Q}_i^{(n)} - \mathbf{Q}^*. \tag{15}$$

$$= \lim_{t\to\infty} (1-u) \sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} (\mathbf{Q}_i^{(n)} - \mathbf{Q}^*). \tag{16}$$

$$= \lim_{t\to\infty} (1-u) \sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}, \tag{17}$$

where (15) follows from the explicit expression for $\mathbf{Q}_t^{it}$, which can be obtained by repeatedly plugging the expression of $\mathbf{Q}_{t-1}^{it}$ in $\mathbf{Q}_t^{it}$ in line 11 in Algorithm 1, (16) follows from the fact that $\sum_{n=1}^{K} \mathbf{w}_t^{(n)} = 1$ for all t, and $(1-u)\sum_{i=0}^{t-1} u^{t-i-1} = 1$ as $t \to \infty$, and (17) follows from (7). If we take the expectation of both sides:

$$\lim_{t\to\infty} \mathbb{E}[\mathcal{E}_t] = \lim_{t\to\infty} (1-u) \sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbb{E}[\mathcal{X}_i^{(n)}] = 0 \tag{18}$$

which follows from the linearity of expectation and (7).

We now prove the upper bound on the variance. $\lim_{t\to\infty} \mathbb{V}[\mathcal{E}_t] =$

$$= \lim_{t\to\infty} \mathbb{V}\Big[(1-u) \sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}\Big]. \tag{19}$$

$$= \lim_{t\to\infty} (1-u)^2 \Big[\sum_{i=0}^{t-1} u^{2(t-i-1)} \Big[\sum_{n=1}^{K} \big(\mathbf{w}_i^{(n)}\big)^2 \mathbb{V}[\mathcal{X}_i^{(n)}] + 2 \sum_{n=1}^{K} \sum_{m=n+1}^{K} \mathbf{w}_i^{(n)} \mathbf{w}_i^{(m)} \operatorname{Cov}(\mathcal{X}_i^{(n)}, \mathcal{X}_i^{(m)}) \Big] \Big]. \tag{20}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \Big[\sum_{i=0}^{t-1} u^{2(t-i-1)} \Big[\sum_{n=1}^{K} \big(\mathbf{w}_i^{(n)}\big)^2 \mathbb{V}[\mathcal{X}_i^{(n)}] + 2 \sum_{n=1}^{K} \sum_{m=n+1}^{K} \mathbf{w}_i^{(n)} \mathbf{w}_i^{(m)} \sqrt{\mathbb{V}[\mathcal{X}_i^{(n)}] \mathbb{V}[\mathcal{X}_i^{(m)}]} \Big] \Big]. \tag{21}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \Big[\sum_{i=0}^{t-1} u^{2(t-i-1)} \Big[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbb{V}[\mathcal{X}_i^{(n)}] + 2 \sum_{n=1}^{K} \sum_{m=1}^{K} \mathbf{w}_i^{(n)} \mathbf{w}_i^{(m)} \sqrt{\mathbb{V}[\mathcal{X}_i^{(n)}] \mathbb{V}[\mathcal{X}_i^{(m)}]} \Big] \Big]. \tag{22}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \Big[\sum_{i=0}^{t-1} u^{2(t-i-1)} \Big[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \frac{\lambda^2}{3} + 2 \sum_{n=1}^{K} \sum_{m=1}^{K} \mathbf{w}_i^{(n)} \mathbf{w}_i^{(m)} \frac{\lambda^2}{3} \Big] \Big]. \tag{23}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \Big[\sum_{i=0}^{t-1} u^{2(t-i-1)} \lambda^2 \Big]. \tag{24}$$

$$\leqslant \frac{(1-u)}{(1+u)} \lambda^2, \tag{25}$$

where (19) follows from (17), (20) follows from the properties of the variance operator and the independence assumption, (21) follows from the Cauchy-Schwarz inequality for the variance, (22) follows from the fact that $\mathbf{w}_t^{(n)} \leqslant 1$ and dropping the constraint in the second summation, (23) follows from (7) and $\lambda = \max_n \lambda_n$, (24) follows from the fact that $\sum_{n=1}^{K} \mathbf{w}_t^{(n)} = 1$ for all t, and (25) follows from the infinite geometric sum formula and $u \in (0,1)$.

B. Proof of Proposition 2

We firstly bound the weight $\mathbf{w}^{(n)}$ into a tighter interval than [0,1]. We first do the calculations for $n \neq 1$. The maximum value of $\mathbf{w}^{(n)}$ is obtained when $\hat{\mathbf{Q}}^{(1)} = \hat{\mathbf{Q}}^{(n)}$ and $\hat{\mathbf{Q}}^{(i)}$ is maximally different than $\hat{\mathbf{Q}}^{(1)}$ for all $i \neq n$. In this case, $\mathbf{w}^{(n)} = 1$, and $\mathbf{w}^{(i)} = 0$ for all $i \notin \{n, 1\}$. When we apply the softmax operator on the $\mathbf{w}$, we obtain the following:

$$\mathbf{w}^{(n)} = \frac{e}{2e + (K-2)} \leqslant \frac{e}{K}. \tag{26}$$

which follows as there are $K > 1$ different weights, and $\mathbf{w}^{(1)} = 1$. On the other hand, the maximum value of $\mathbf{w}^{(1)}$ is obtained when $\hat{\mathbf{Q}}^{(i)}$ is maximally different than $\hat{\mathbf{Q}}^{(1)}$ for all $i \neq 1$. In this case, $\mathbf{w}^{(1)} = 1$, and $\mathbf{w}^{(i)} = 0$ for all $i \neq 1$. When we apply the softmax on $\mathbf{w}$, we obtain the following:

$$\mathbf{w}^{(1)} = \frac{e}{e + (K-1)} \leqslant \frac{e}{K}. \tag{27}$$

Combining (26) and (27), the following holds for all n:

$$\mathbf{w}^{(n)} \leqslant \frac{e}{K}. \tag{28}$$

We note that this bound is useful for $K > 2$. On the other hand, the minimum value of $\mathbf{w}^{(n)}$ for $n \neq 1$ is obtained when $\hat{\mathbf{Q}}^{(1)}$ is maximally different than $\hat{\mathbf{Q}}^{(n)}$ and $\hat{\mathbf{Q}}^{(i)} = \hat{\mathbf{Q}}^{(1)}$ for all $i \neq n$. In this case, $\mathbf{w}^{(n)} = 0$, and $\mathbf{w}^{(i)} = 1$ for all $i \neq n$. When we apply the softmax operator on the $\mathbf{w}$, we obtain the following:

$$\mathbf{w}^{(n)} = \frac{1}{1 + e(K-1)} \geqslant \frac{e^{-1}}{K}. \tag{29}$$

which follows because $K > 1$. Similarly, the minimum value of $\mathbf{w}^{(1)}$ is obtained when $\hat{\mathbf{Q}}^{(i)} = \hat{\mathbf{Q}}^{(1)}$ for all $i \neq 1$. In this case, $\mathbf{w}^{(i)} = 1$ for all i. When we apply the softmax operator on the $\mathbf{w}$, we obtain the following:

$$\mathbf{w}^{(1)} = \frac{1}{K}. \tag{30}$$

Combining (29) and (30), the following holds for all n:

$$\mathbf{w}^{(n)} \geqslant \frac{1}{K}. \tag{31}$$

Combining (28) and (31), the weights $\mathbf{w}^{(n)}$ can be shown to take values in the following interval: $\mathbf{w}^{(n)} \in [\frac{1}{K}, \frac{e}{K}]$. Then, the term $\sum_{n=1}^{K} \left(\mathbf{w}_i^{(n)}\right)^2$ in (20) can be upper bounded as:

$$\sum_{n=1}^{K} (\mathbf{w}_i^{(n)})^2 \leqslant \sum_{n=1}^{K} (\frac{e}{K})^2 \leqslant \frac{e^2}{K}. \tag{32}$$

The following expressions are valid for all (s, a) pairs; hence, we drop the (s, a) notation for simplicity. Using (32), (7), and $\lambda = \max_n \lambda_n$, we can show the following:

$$\mathbb{V}\big[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}\big] \leqslant \sum_{n=1}^{K} \left(\mathbf{w}_i^{(n)}\right)^2 \mathbb{V}\big[\mathcal{X}_i^{(n)}\big] \leqslant \frac{e^2}{K}\frac{\lambda^2}{3}. \tag{33}$$

Let $u_0 = 0$ and $\bar{u}_i = (1-u_i)\prod_{j=i+1}^{t-1} u_j$. Using the explicit expression for $\mathbf{Q}_t^{it}$, which can be obtained by repeatedly plugging the expression of $\mathbf{Q}_{t-1}^{it}$ in $\mathbf{Q}_t^{it}$ in line 11 in Algorithm 1 and the facts that $\sum_{i=0}^{t} \bar{u}_i = 1$ and $u_0 = 0$, the following can be derived:

$$\mathbf{Q}_t^{it} = \sum_{i=0}^{t-1} \bar{u}_i \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbf{Q}_i^{(n)}, \tag{34}$$

Then we can show the following:

$$\mathbb{V}[\mathcal{E}_t] = \mathbb{V}\Big[\sum_{i=0}^{t-1} \bar{u}_i \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}\Big]. \tag{35}$$

$$= \sum_{i=0}^{t-1} \bar{u}_i^2 \mathbb{V}[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}] + 2\sum_{i=0}^{t-1}\sum_{j\neq i}^{t-1} \bar{u}_i \bar{u}_j \operatorname{Cov}(\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}, \sum_{n=1}^{K} \mathbf{w}_j^{(n)} \mathcal{X}_j^{(n)}). \tag{36}$$

$$\leqslant \sum_{i=0}^{t-1} \bar{u}_i^2 \mathbb{V}[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}] + 2\sum_{i=0}^{t-1}\sum_{j=i+1}^{t-1} \bar{u}_i \bar{u}_j \sqrt{\mathbb{V}(\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}) \mathbb{V}(\sum_{n=1}^{K} \mathbf{w}_j^{(n)} \mathcal{X}_j^{(n)})}. \tag{37}$$

$$\leqslant \frac{e^2}{K}\frac{\lambda^2}{3}\Big[\sum_{i=0}^{t-1} \bar{u}_i^2 + 2\sum_{i=0}^{t-1}\sum_{j=i+1}^{t-1} \bar{u}_i \bar{u}_j\Big]. \tag{38}$$

$$\leqslant \frac{c(\lambda, u)}{K}, \tag{39}$$

where (35) follows from (34), (36) follows from the properties of the variance operator, (37) follows from the Cauchy-Schwarz inequality for the variance, (38) follows from (33), and (39) follows as $c(\lambda, u)$ is a constant of K.

C. Proof of Proposition 3

If the spectral norm of a matrix $\mathbf{P}$ is less than 1 ($\|\mathbf{P}\| < 1$), then the following can be shown [45]:

$$(\mathbf{I} - \mathbf{P})^{-1} = \sum_{i=0}^{\infty} \mathbf{P}^i, \tag{40}$$

where $\mathbf{I}$ is the identity matrix. The Q-functions under policy $\hat{\pi}$ can be obtained using Bellman's equation as follows [4]:

$$\mathbf{Q}_{\hat{\pi}} = (\mathbf{I} - \gamma \mathbf{P}_{\hat{\pi}})^{-1} \mathbf{c}_{\hat{\pi}}, \tag{41}$$

where $\mathbf{P}_{\hat{\pi}}$ and $\mathbf{c}_{\hat{\pi}}$ are PTM and cost vectors under policy $\hat{\pi}$, respectively. Then, we can proceed as follows:

$$\mathbf{Q}_{\hat{\pi}}^{(1)} - \mathbf{Q}_{\hat{\pi}}^{(n)} = (\mathbf{I} - \gamma \mathbf{P}_{\hat{\pi}})^{-1} \mathbf{c}_{\hat{\pi}} - (\mathbf{I} - \gamma \mathbf{P}_{\hat{\pi}}^{n})^{-1} \mathbf{c}_{\hat{\pi}}. \tag{42}$$

$$= \big[\gamma \mathbf{P}_{\hat{\pi}} \sum_{i=0}^{\infty} (\gamma \mathbf{P}_{\hat{\pi}})^{ni} \sum_{k=0}^{n-2} (\gamma \mathbf{P}_{\hat{\pi}})^k + \sum_{j=0}^{\infty} (\gamma^{nj} - \gamma^j) \mathbf{P}_{\hat{\pi}}^{nj}\big] \mathbf{c}_{\hat{\pi}}. \tag{43}$$

$$\leqslant \gamma \mathbf{P}_{\hat{\pi}} \sum_{i=0}^{\infty} (\gamma \mathbf{P}_{\hat{\pi}})^{ni} \sum_{k=0}^{n-2} (\gamma \mathbf{P}_{\hat{\pi}})^k \mathbf{c}_{\hat{\pi}} \tag{44}$$

where (42) follows from (41), (43) follows from the assumption that $\|\mathbf{P}_{\hat{\pi}}\| < 1$, (40) and grouping the common terms in the expansions, and (44) follows from the facts that $\gamma < 1$ and the elements of $\mathbf{P}_{\hat{\pi}}$ and that $\mathbf{c}_{\hat{\pi}}$ are non-negative. If we take the norm of both sides, we obtain:

$$\|\mathbf{Q}_{\hat{\pi}}^{(1)} - \mathbf{Q}_{\hat{\pi}}^{(n)}\| \leqslant \|\gamma \mathbf{P}_{\hat{\pi}} \sum_{i=0}^{\infty} (\gamma \mathbf{P}_{\hat{\pi}})^{ni} \sum_{k=0}^{n-2} (\gamma \mathbf{P}_{\hat{\pi}})^k \mathbf{c}_{\hat{\pi}}\|. \tag{45}$$

$$\leqslant \gamma \|\mathbf{P}_{\hat{\pi}}\| \sum_{i=0}^{\infty} (\gamma \|\mathbf{P}_{\hat{\pi}}\|)^{ni} \sum_{k=0}^{n-2} (\gamma \|\mathbf{P}_{\hat{\pi}}\|)^k \|\mathbf{c}_{\hat{\pi}}\|. \tag{46}$$

$$< \gamma \sum_{i=0}^{\infty} \gamma^{ni} \sum_{k=0}^{n-2} \gamma^k \|\mathbf{c}_{\hat{\pi}}\|. \tag{47}$$

$$< \frac{\gamma}{1-\gamma^n} \frac{1-\gamma^{n-1}}{1-\gamma} \|\mathbf{c}_{\hat{\pi}}\|, \tag{48}$$

where (45) follows from (44), (46) follows from the upper bound on the norm of a matrix-vector product [45], (47) follows from the assumption that $\|\mathbf{P}_{\hat{\pi}}\| < 1$, and (48) follows from the finite and infinite geometric sum formulas.

D. Proof of Proposition 4

Assume $||\mathbf{P}_{\boldsymbol{\pi}}|| < 1$. We analyze $\mathbf{Q}_{\hat{\pi}}^{(n)}$ when $\gamma \to 1$.

$$\lim_{\gamma \to 1} \mathbf{Q}_{\hat{\pi}}^{(n)} = \lim_{\gamma \to 1} (\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\pi}}^{n})^{-1} \mathbf{c}_{\boldsymbol{\pi}} \tag{49}$$

$$= (\mathbf{I} - \mathbf{P}_{\boldsymbol{\pi}}^{n})^{-1} \mathbf{c}_{\boldsymbol{\pi}} \tag{50}$$

$$= \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^{n} \mathbf{c}_{\boldsymbol{\pi}} + (\mathbf{P}_{\boldsymbol{\pi}}^{n})^2 \mathbf{c}_{\boldsymbol{\pi}} + (\mathbf{P}_{\boldsymbol{\pi}}^{n})^3 \mathbf{c}_{\boldsymbol{\pi}} + ..., \tag{51}$$

which follows from (40). Then, we can write the following:

$$\lim_{\gamma \to 1} \mathbf{Q}_{\hat{\pi}}^{(1)} = \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}} \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^2 \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^3 \mathbf{c}_{\boldsymbol{\pi}} + ... \tag{52}$$

$$\lim_{\gamma \to 1} \mathbf{Q}_{\hat{\pi}}^{(2)} = \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^2 \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^4 \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^6 \mathbf{c}_{\boldsymbol{\pi}} + ... \tag{53}$$

$$\lim_{\gamma \to 1} \mathbf{Q}_{\hat{\pi}}^{(3)} = \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^3 \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^6 \mathbf{c}_{\boldsymbol{\pi}} + \mathbf{P}_{\boldsymbol{\pi}}^9 \mathbf{c}_{\boldsymbol{\pi}} + ... \tag{54}$$

We observe that $\mathbf{Q}_{\hat{\pi}}^{(1)}$ is the largest since it has all the terms that $\mathbf{Q}_{\hat{\pi}}^{(n)}$ contains for all $n \neq 1$. We also have the following partial orderings in the limiting case ($\gamma \to 1$):

$$\mathbf{Q}_{\hat{\pi}}^{(2)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(4)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(8)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(16)} ... \tag{55}$$

$$\mathbf{Q}_{\hat{\pi}}^{(3)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(6)} \geqslant \mathbf{v}_{\pi}^{(12)} ... \tag{56}$$

$$\mathbf{Q}_{\hat{\pi}}^{(5)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(10)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(20)} ... \tag{57}$$

$$\mathbf{Q}_{\hat{\pi}}^{(7)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(14)} \geqslant \mathbf{Q}_{\hat{\pi}}^{(28)} ..., \tag{58}$$

which follow as the elements of $\mathbf{P}_{\hat{\pi}}$ and $\mathbf{c}_{\hat{\pi}}$ are non-negative.

REFERENCES

[1] Yunxia Chen, Qing Zhao, and Ananthram Swami. Distributed spectrum sensing and access in cognitive radio networks with energy constraint. *IEEE Transactions on Signal Processing*, 57(2):783–797, 2009.

[2] Fan Zhang and Vincent K. N. Lau. Closed-form delay-optimal power control for energy harvesting wireless system with finite energy storage. *IEEE Transactions on Signal Processing*, 62(21):5706–5715, 2014.

[3] Shuqi Chai and Vincent K. N. Lau. Online trajectory and radio resource optimization of cache-enabled uav wireless networks with content and energy recharging. *IEEE Transactions on Signal Processing*, 68:1286–1299, 2020.

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

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[6] Dejan V. Djonin and Vikram Krishnamurthy. Q-learning algorithms for constrained markov decision processes with randomized monotone policies: Application to mimo transmission control. *IEEE Transactions on Signal Processing*, 55(5):2170–2181, 2007.

[7] Soummya Kar, José M. F. Moura, and H. Vincent Poor. $\mathcal{QD}$-learning: A collaborative distributed strategy for multi-agent reinforcement learning through $\mathrm{consensus} + \mathrm{innovations}$. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, 2013.

[8] Minh Hanh Ngo and Vikram Krishnamurthy. Monotonicity of constrained optimal transmission policies in correlated fading channels with arq. *IEEE Transactions on Signal Processing*, 58(1):438–451, 2010.

[9] Zohreh Hajiakhondi-Meybodi, Arash Mohammadi, Ming Hou, and Konstantinos N. Plataniotis. Dqlel: Deep Q-learning for energy-optimized los/nlos uwb node selection. *IEEE Transactions on Signal Processing*, 70:2532–2547, 2022.

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

[11] Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*, pages 1032–1040. PMLR, 2016.

[12] 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.

[13] 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.

[14] Mohammad Ghavamzadeh, Hilbert Kappen, Mohammad Azar, and Rémi Munos. Speedy Q-learning. *Advances in neural information processing systems*, 24, 2011.

[15] 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.

[16] Libin Liu and Urbashi Mitra. On sampled reinforcement learning in wireless networks: Exploitation of policy structures. *IEEE Transactions on Communications*, 68(5):2823–2837, 2020.

[17] Libin Liu, Arpan Chattopadhyay, and Urbashi Mitra. On solving mdps with large state space: Exploitation of policy structures and spectral properties. *IEEE Transactions on Communications*, 67(6):4151–4165, 2019.

[18] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin Q-learning: Controlling the estimation bias of Q-learning. *CoRR*, abs/2002.06487, 2020.

[19] 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.

[20] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *CoRR*, abs/2109.00157, 2021.

[21] Roger McFarlane. A survey of exploration strategies in reinforcement learning. *McGill University*, 2018.

[22] Talha Bozkus and Urbashi Mitra. Ensemble link learning for large state space multiple access communications. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 747–751, 2022.

[23] Talha Bozkus and Urbashi Mitra. Ensemble graph Q-learning for large scale networks. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

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

[25] Bruce A Craig and Peter P Sendi. Estimation of the transition matrix of a discrete-time markov chain. *Health economics*, 11(1):33–42, 2002.

[26] 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.

[27] Maren Awiszus and Bodo Rosenhahn. Markov chain neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2180–2187, 2018.

[28] Chris Sherlaw-Johnson, Steve Gallivan, and Jim Burridge. Estimating a markov transition matrix from observational data. *Journal of the Operational Research Society*, 46(3):405–410, 1995.

[29] Jitendra Tugnait. Adaptive estimation and identification for discrete systems with markov jump parameters. *IEEE Transactions on Automatic control*, 27(5):1054–1065, 1982.

[30] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[31] Talha Bozkus and Urbashi Mitra. Supplementary appendix. https://github.com/talhabozkus/TSP_23_supplementary_file.git, 2023.

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

[33] 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.

[34] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020.

[35] Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

[36] 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.

[37] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[38] Carlo D'Eramo, Andrea Cini, Alessandro Nuara, Matteo Pirotta, Cesare Alippi, Jan Peters, Marcello Restelli, et al. Gaussian approximation for bias reduction in Q-learning. 2021.

[39] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769 – 2794, 2007.

[40] Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. In *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*, pages 308–322. Springer, 2007.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[42] Daniel Russo. Approximation benefits of policy gradient methods with aggregated states. *Management Science*, 2023.

[43] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *AI&M*, 2006.

[44] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716, 2021.

[45] Gilbert Strang. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006.