Leveraging Digital Cousins for Ensemble Q-Learning in Large-Scale Wireless Networks

Talha Bozkus, *Graduate Student Member, IEEE* and Urbashi Mitra, *Fellow, IEEE*

***Abstract*—Optimizing large-scale wireless networks, including optimal resource management, power allocation, and throughput maximization, is inherently challenging due to their non-observable system dynamics and heterogeneous and complex nature. Herein, a novel ensemble Q-learning algorithm that addresses the performance and complexity challenges of the traditional Q-learning algorithm for optimizing wireless networks is presented. Ensemble learning with synthetic Markov Decision Processes is tailored to wireless networks via new models for approximating large state-space observable wireless networks. In particular, *digital cousins* are proposed as an extension of the traditional digital twin concept wherein multiple Q-learning algorithms on multiple synthetic Markovian environments are run in parallel and their outputs are fused into a single Q-function. Convergence analyses of key statistics and Q-functions and derivations of upper bounds on the estimation bias and variance are provided. Numerical results across a variety of real-world wireless networks show that the proposed algorithm can achieve up to 50% less average policy error with up to 40% less runtime complexity than the state-of-the-art reinforcement learning algorithms. It is also shown that theoretical results properly predict trends in the experimental results.**

***Index Terms*—Reinforcement learning, Q-learning, Markov decision processes, wireless networks.**

I. Introduction

THE optimization of large-scale real-world wireless networks is challenging due to their inherent complexity, dynamic nature, and many unobservable features [1], [2], [3]. To overcome these challenges, digital twins have been proposed as a solution within next-generation wireless networks [4], [5]. In the context of wireless networks, digital twins function as virtual, exact replicas of key systems, including IoT, edge, and mobile devices, as well as network infrastructures and communication protocols. They provide a framework for the modeling, simulation, troubleshooting, and optimization of a variety of problems, such as dynamic resource allocation, interference and latency minimization, and optimal spectrum allocation [6], [7], [8].

Herein, we use Markov Decision Processes (MDPs) to model wireless networks. MDPs provide a mathematical framework for modeling sequential decision-making problems under uncertainty [9]. In real-world scenarios, the underlying system dynamics of wireless networks are often non-observable or difficult to estimate accurately [10]. To this end, we will adapt model-free reinforcement learning approaches, in particular, Q-learning, to optimize wireless networks. Our approach will also incorporate model estimation.

Q-learning [9] has been widely employed for the optimization of wireless networks [11], [12], [13]. However, it suffers from certain limitations, particularly when dealing with large state-space wireless networks. Several variants of the Q-learning algorithm have been proposed to address these limitations. The estimation bias is considered in [14], [15], whereas the work of [16], [17] deals with the estimation variance. Other methods improve learning speed [18] and sample efficiency [19], deal with the high dimensionality of the state space [20], [21], and address exploration challenges [22], [23]. On the other hand, the work of [24], [25] is tailored to the optimization of wireless networks. These algorithms can also be categorized based on their implementation strategies, with some using a single Q-function estimator on a single Markovian environment [9], [18], [19], while others employ multiple Q-function estimators on a single Markovian environment [14], [15], [16], [17]. The design of **multiple** Q-function estimators that can be employed on **multiple** Markovian environments has not been well-studied (see [26], [27], [28]). However, the multiplicity of Markovian environments can accelerate training, improve data efficiency, and yield more accurate and stable Q-functions.

The traditional digital twin approach of employing exact replicas of the original Markovian system can leverage the best outcomes from different replicas (such as the most accurate or stable Q-functions) and enable parallel exploration of diverse states and actions. Moreover, digital twins with the same algorithms, but different parameter values, can enable more efficient parameter optimization. Despite their promising advantages, traditional digital twins have several challenges [29], [30]. They lack the ability to incorporate variations or alternative representations of the original Markovian environment, which restricts the adaptability of the algorithm to different scenarios and settings. Creating replicas that precisely mimic noisy, dynamic,

Manuscript received 30 September 2023; revised 17 January 2024; accepted 5 February 2024. Date of publication 15 February 2024; date of current version 28 February 2024. This work was supported in part by ARO under Grant W911NF1910269; in part by DOE under Grant DE-SC0021417; in part by Swedish Research Council under Grant 2018-04359; in part by NSF under Grant CCF-2008927, Grant RINGS-2148313, Grant CCF-2200221, Grant CIF-2311653; in part by ONR under Grant 503400-78050, Grant N00014-15-1-2550; and in part by USC + Amazon Center on Secure and Trusted Machine Learning. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shaofeng Zou. *(Corresponding author: Talha Bozkus.)*

The authors are with Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: bozkus@usc.edu; ubli@usc.edu).

Digital Object Identifier 10.1109/TSP.2024.3366434

or uncertain environments can enhance noise. Insufficient or biased sampling in the original environment negatively affects the accuracy of the digital twins, leading to sub-optimal learning and decision-making.

Herein, we offer the concept of *digital cousins* for wireless networks in the context of reinforcement learning to overcome these challenges of traditional digital twins by exploiting our prior work [26], [27], [28], [31]. Digital cousins are distinct, yet structurally related synthetic Markovian environments (SMEs). These environments are inherently different, but share similar characteristics and dynamics, enabling improved performance with respect to exploration and optimization. In particular, digital cousins enable the reinforcement learning agent to explore longer trajectories, discover new state-action pairs, and learn from indirect experiences, which facilitates faster learning of the environment. These environments also help the agent exploit patterns, particularly in structured environments as found in wireless networks, and adapt to changing dynamics by leveraging past experiences. Overall, these advantages result in substantial improvements in the speed, data efficiency, accuracy, and robustness of the Q-learning algorithm, as will be shown later. It is worth noting that digital cousins improve exploration, and thus learning, in part because each different Markovian environment runs at a different *time scale*.

Our previous work [31] presented low-complexity techniques for modeling and approximating observable MDPs for policy optimization. Furthermore, we emphasized the advantages of utilizing multiple SMEs to improve the accuracy and complexity of the traditional Q-learning algorithm across non-observable MDPs for policy learning in [26], [27], [28]. The current work **synthesizes** both approaches and specifically focuses on the optimization of real-world wireless networks that vary in topology, scale, complexity, objective, and implementation.

The current work has key differences with respect to [28], which lead to significant improvements. Firstly, we particularly focus on the optimization of different wireless networks versus the unstructured networks considered in [28]. The structured nature of wireless networks enables efficient learning by estimating the underlying model of different SMEs with lower complexity as well as the design of structural state-aggregation algorithms to minimize the memory complexity of the Q-learning algorithm across large state-spaces. In particular, we employ a more accurate, robust, and practical way of modeling multiple SMEs through the use of co-link matrices; in [28], powers of the probability transition matrix are employed. The current approach captures complex, bidirectional, and interdependent dynamics of Markovian transitions within wireless networks and enhances the robustness of multiple SMEs against variability and uncertainties inherent in these transitions.

Secondly, our proposed algorithm utilizes a straightforward policy comparison as the cost metric, avoiding the complexities of the divergence metric for Q-functions in [28]. This design results in increased computational efficiency and enhanced resilience to noise, errors, and numerical instabilities in the Q-functions. These two key changes, the use of co-link matrices to model different SMEs and direct policy optimization, also yield improvements in our theoretical results. Our bounds are more accurate and require fewer assumptions, approximations, and constraints on system parameters than our prior work. Throughout the paper, we highlight the differences between the current work and our prior work [28].

Asynchronous Advantage Actor-Critic (A3C) [32] conceptually resembles our approach, Ensemble Synthetic Q-learning (ESQL). Both A3C and ESQL involve parallel learning to explore the state space; however, these strategies have fundamental differences. In particular, A3C is an actor-critic algorithm that models the critic (value function) and actor (policy) via neural networks. Thus, A3C suffers from the classical challenges of neural networks with respect to training time, computational complexity, robustness, and interpretability. Moreover, the critic's value function may be inaccurate or inconsistent with the actor's policy, leading to instability. ESQL utilizes Q-learning and hence obviates these issues.

With regards to parallel learning, A3C uses multiple copies of the same environment (like digital twins). In contrast, ESQL constructs synthetic environments that are *distinct* but structurally related and mutually informative in a model-based fashion (digital cousins, not twins). In A3C, each network is an independent neural network initialized differently and explores the same environment in different ways, whereas, in ESQL, each network is an independent Q-learning learning different representations of the original environment through n-hop/multi-time-scale information. Unlike A3C, which periodically synchronizes network parameters, ESQL requires no synchronization; instead, Q-functions from each environment contribute to the global Q-function with weighting, enabling exploitation of the most useful environments. As ESQL exploits structural properties of multiple environments, data-efficient estimation strategies are possible to reduce sample complexity, and structural-state-action aggregation algorithms can mitigate memory complexity unlike A3C. Finally, while A3C offers performance that is competitive with ESQL, it does so with a significant increase in computational complexity; furthermore, the performance advantage of A3C for small networks degrades as the network grows in size.

The **main contributions** of the paper are as follows:

(i) We leverage the innovative concept of *digital cousins* as an extension of traditional digital twins and new models for approximating large state-space observable wireless networks to construct multiple distinct, yet structurally related SMEs for ensemble model-free learning.

(ii) We propose a novel ensemble learning algorithm, where multiple Q-learning algorithms are run in parallel on multiple SMEs. Their outputs are fused into a single Q-function estimate via a policy comparison between different environments to produce a near-optimal deterministic policy with low complexity.

(iii) We analyze the stability and convergence of the proposed algorithm from deterministic and probabilistic perspectives and provide upper-bound analysis on estimation bias and variance. We also consider a wider range of independence assumptions on the error from each digital cousin versus [28]. This improved analysis is facilitated by our tailored focus on wireless networks.

(iv) Numerical simulations across different real-world wireless networks show that the proposed algorithm outperforms the state-of-the-art reinforcement learning algorithms as well as methods proposed in our prior work [26], [27], [28] by achieving up to 50% less average policy error with up to 40% less runtime complexity. A considerable amount of reduction in memory complexity is also achieved through an original structural state-action aggregation algorithm. In addition, the experimental results are shown to follow the theoretical results closely.

We use the following notation: vectors are bold lower case (**x**); matrices are bold upper case (**A**); sets are in calligraphic font ($\mathcal{S}$); and scalars are non-bold (α).

II. System Model and Methods

A. Infinite Horizon Discounted Cost MDP Model

MDPs consist of 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 $p_a(s, s')$, 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* [9] 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 [9]. Hence, we herein consider deterministic and stationary policies.

B. Model-Free Reinforcement Learning: Q-Learning

When the system dynamics (p and c) are unknown or non-observable, Q-learning can be used to solve (1) and (2). The objective of Q-learning is to determine the optimal policy π^* by learning the Q functions for all state-action pairs (s, a). This learning process is governed by the 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 [9]. 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). By interacting with the environment and collecting samples $\{s, a, s', c\}$, the agent updates the Q-functions using (3). The learning strategy involves determining 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 [33]. The optimal policy and value functions can be derived from the Q-functions as:

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

C. Extension of Digital Twins: Digital Cousins

In our prior work [26], [27], [28], we introduced the novel concept of *digital cousins* for general networks, which represents a set of closely related synthetic Markovian environments that share analogous structures and characteristics with both the original Markovian environment and each other. This approach effectively captures the essence of the original Markovian environment while introducing variations, addressing many of the limitations of traditional digital twins.

There are several ways to create SMEs (and the corresponding PTTs) based on the PTT of the original environment $\mathbf{P}$. A natural approach is to employ a function of $\mathbf{P}$ or $\mathbf{P}^T$. It is desirable to maintain a consistent relationship between transition probabilities $p_a(s, s')$ across different environments while ensuring that the PTTs of different environments are row-stochastic or can be converted to such form via appropriate normalization techniques without altering the underlying structures. We herein employ the *colink method*, which was originally introduced in [34], which involves constructing a set of symmetric matrices based on the original probability transition matrix. Our prior work [31] demonstrated the effectiveness of the colink method for policy optimization in large state-space wireless networks (with known $\mathbf{P}$). Herein, the structure of wireless networks can be exploited to improve the estimation of the co-link matrices for learning in contrast to the unstructured network models considered in [28].

The colink method produces the n^{th} *order similarity tensor*, denoted by $\mathbf{L}^{(n)}$ ($\mathbf{L}_i^{(n)}$ is the matrix for the i^{th} action), which can be expressed as:

$$\mathbf{L}_i^{(n)} \stackrel{\text{def}}{=} \sum_{k=0}^{n-2} \mathbf{P}_i^{n-k-1} (\mathbf{P}_i^T)^{k+1} + (\mathbf{P}_i^T)^{n-k-1} \mathbf{P}_i^{k+1}, \ \forall i \in \mathcal{A}, \quad (5)$$

where $\mathbf{P}_i$ is the PTM corresponding to the i^{th} action in $\mathbf{P}$. We employ l_1 normalization on $\mathbf{L}_i^{(n)}$, $\forall i \in \mathcal{A}$ (so that the sum

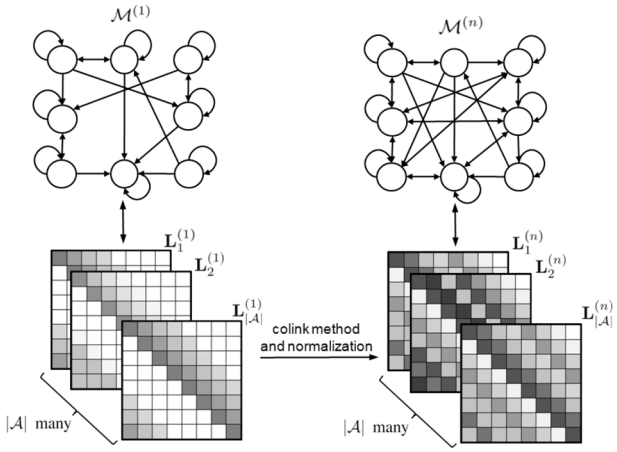


Fig. 1. The relationship between $\mathbf{L}^{(1)}$ (= $\hat{\mathbf{P}}$), $\mathbf{L}^{(n)}$, $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(n)}$.

of each row in $\mathbf{L}_i^{(n)}$ is 1). This transformation preserves the structure of $\mathbf{L}^{(n)}$ while generating row-stochastic PTMs, as discussed in [31]. The resulting synthetic Markovian environment, denoted as $\mathcal{M}^{(n)}$, captures the n-step probabilistic transitions between states represented by $\mathbf{L}^{(n)}$.

When dealing with large dimensions of $\mathbf{P}$ or a high order n, utilizing (5) can pose computational challenges due to the increasing number of matrix multiplications involved. However, it is possible to overcome this issue by leveraging the structural properties of $\mathbf{P}$. By exploiting properties such as diagonal, circulant, Toeplitz, or block structures, alternative approaches can be developed to compute (5) with significantly lower complexity [31].

The original PTT, $\mathbf{P}$, is initially unknown; thus, it needs to be estimated to create the PTTs of multiple SMEs. If the number of one-step transitions between states can be counted, the transition probabilities can be accurately estimated via *sample averaging* [35]. If there is a particular structure in one-step transition probabilities, as found in wireless networks, a more data-efficient estimation can also be performed [26], [31]. We denote the estimated PTT by $\hat{\mathbf{P}}$, and construct $\mathbf{L}^{(n)}$ using $\hat{\mathbf{P}}$. The relationship between the original Markovian environment ($\mathcal{M}^{(1)}$) and the n^{th} synthetic Markovian environment ($\mathcal{M}^{(n)}$, $n > 1$) is given in Fig. 1.

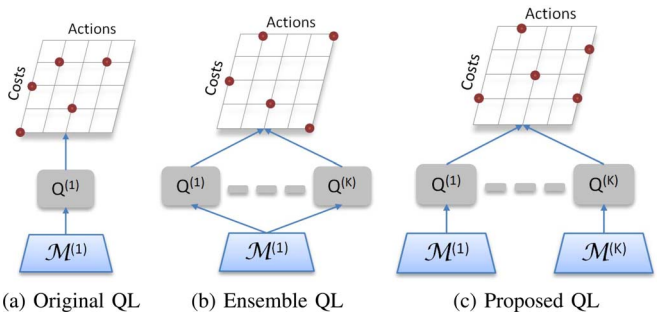
The accuracy of $\mathbf{L}^{(n)}$ and $\mathcal{M}^{(n)}$ is affected by the quality of sampling and the order n. Insufficient sampling can lead to accumulated errors during matrix multiplications, resulting in lower accuracy for higher-order environments (*i.e.* for large n). Additionally, as n increases, the interpretation of $\mathbf{L}^{(n)}$ becomes less intuitive, making it challenging to understand state transitions over longer time horizons. While $\mathbf{L}^{(n)}$ tends to converge to a fixed tensor as n increases, this convergence may also cause the loss of underlying system dynamics. In particular, our prior work [31] showed the existence of an optimal n that maximizes the accuracy of the estimated policy. We underline that this optimal n is not large, of the order 5.

The use of colink methods provides several advantages over our prior approach [28]. Co-link modeling effectively captures **bidirectional** relationships in Markovian transitions, which enables improved modeling of feedback loops and other interdependencies seen in wireless networks [31]. Incorporating bidirectional relationships can also enhance the robustness of the synthetic Markovian environments against stochastic impairments that might be present in the underlying Markovian transitions. The nature of colink representations also ensures sufficient variability across multiple SMEs.

Algorithm 1 Ensemble Synthetic Q-Learning (ESQL)

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)
      $\boldsymbol{\pi}_t^{(n)}(s) \leftarrow \text{argmin}_{a'} \mathbf{Q}_t^{(n)}(s, a')$
      $\mathbf{w}_t^{(n)} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{S}|} \mathbf{1}(\boldsymbol{\pi}_t^{(1)}(j) = \boldsymbol{\pi}_t^{(n)}(j))$
    end for
    $\mathbf{w}_t \leftarrow \text{softmax}(\mathbf{w}_t)$
    $\mathbf{Q}_{t+1}^{it} \leftarrow u_t \mathbf{Q}_t^{it} + (1 - u_t) \sum_{n=1}^{K} \mathbf{w}_t^{(n)} \mathbf{Q}_t^{(n)}$
    $t \leftarrow t + 1$
  end
end while
$\hat{\boldsymbol{\pi}}(s) \leftarrow \text{argmin}_{a'} \mathbf{Q}^{it}(s, a')$



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

Fig. 2. Comparison of Q-learning (QL) algorithms based on their implementation strategies.

III. Algorithm

This section introduces the proposed algorithm, Ensemble Synthetic Q-Learning (ESQL) (Algorithm 1). We leverage $K - 1$ 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 comparison between the traditional Q-learning algorithm, conventional ensemble Q-learning algorithms, and the proposed algorithm is given in Fig. 2, where $\mathbf{Q}^{(n)}$ represents the Q-function estimator for the Q-learning algorithm running on $\mathcal{M}^{(n)}$ for $n \in \{1, 2, ..., K\}$.

Algorithm 1 has several inputs: the trajectory length (l), the minimum number of visits required for each state-action 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 the K different environments ($\mathbf{Q}^{(n)}$ for $n \in \{1,2,...,K\}$). Let $\mathbf{w}_t$ denote the weight vector of size K at time t, where $\mathbf{w}_t^{(n)}$ represents the n^{th} element of $\mathbf{w}_t$. At $t=0$, the weight vector $\mathbf{w}_0$ is initialized randomly in line 1. Each element is chosen uniformly random from the range [0,1], and the vector is subsequently softmax-normalized to ensure that $\sum_{n=1}^{K} \mathbf{w}_0^{(n)} = 1$. This initialization serves to break the symmetry. The iterations continue until each state-action pair in $\mathcal{M}^{(1)}$ is visited at least v times (in line 2), ensuring sufficient capture of different state-action dynamics. 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. Independent samples are collected from each different Markovian environment, and corresponding Q-tables are updated independently in line 6. We note that although the initial states are the same, different actions are taken following the epsilon-greedy policy of each different environment; thus, 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, we obtain a set of *individual policies*, $\boldsymbol{\pi}_t^{(n)}$, by minimizing the current Q-functions $\mathbf{Q}_t^{(n)}$. In line 8, we compute the *correct estimation rate vector* $\mathbf{w}_t$, and its n^{th} element ($\mathbf{w}_t^{(n)}$) represents how close the individual policy $\boldsymbol{\pi}_t^{(n)}$ is to the policy of the original environment $\boldsymbol{\pi}_t^{(1)}$. Hence, $\mathbf{w}_t^{(n)}$ is a good indicator of how useful the most recent samples from $\mathcal{M}^{(n)}$ are for constructing the ensemble output. In line 10, the vector $\mathbf{w}_t$ is softmax-normalized, and used to update the Q-function output of Algorithm 1, denoted as $\mathbf{Q}_t^{it}$, in line 11. During the update of $\mathbf{Q}_t^{it}$, the algorithm balances *exploitation* by utilizing a fraction u_t of the $\mathbf{Q}_t^{it}$ from the previous iteration, while promoting *exploration* by sampling multiple Markovian environments based on their weights, with their contributions weighted by $1-u_t$. The estimated policy $\hat{\boldsymbol{\pi}}$ is derived from $\mathbf{Q}^{it}$ in line 15.

Our approach directly optimizes **policies** and compares the optimal policies of different environments to determine weights (in lines 7-8) in contrast to [28], where the Q-functions are optimized and utilized in the weighting mechanism (see Algorithm 1 of [28], lines 7-8). This approach simplifies algorithm design and improves performance in several ways. Firstly, policy comparison is computationally efficient, as it only involves comparing selected actions instead of the entire Q-table, resulting in faster computation. Secondly, updating the policy directly robustifies against bias, error, or noise in the Q-functions. Thirdly, policies are represented by discrete actions, which eliminates issues associated with taking the difference between small Q-functions, thus providing numerical stability. Lastly, direct policy optimization yields more interpretable results.

The *iterative* update of $\mathbf{Q}_t^{it}$ with the current weight vector $\mathbf{w}_t$ captures *asymmetric* information between different Markovian environments. Initially, $\mathcal{M}^{(1)}$ provides more useful samples as it is the original Markovian environment, and there are limited observations for higher-order relationships versus the first-order relationships. As iterations progress, $\mathcal{M}^{(n)}$ for larger n contributes more, improving exploration and reducing reliance on the first-order relationships.

We emphasize that our proposed algorithm combines the features of online and offline RL methods. In real-time, we estimate the PTT of the original environment through sampling and update the corresponding Q-functions (**online** part). Simultaneously, we construct PTTs for multiple SMEs and learn their Q-functions using pre-collected data (**offline** part). Unlike hybrid RL algorithms [36], which uses both offline and online data collection, our approach continuously updates Q-functions in real-time while leveraging pre-collected data for improved accuracy and stability.

While our approach includes model-based learning components such as estimating the PTT of the original Markovian environment, it differs from model-based reinforcement learning [37] in several ways: (i) We focus on learning Q-functions using Q-learning across multiple Markovian environments to determine optimal policies, rather than accurately modeling environment dynamics. (ii) The estimated PTT is used to generate multiple SMEs to facilitate learning and exploration, and is not directly used for decision-making. (iii) We focus on sampling-driven cost collection as opposed to explicit cost function estimation.

A. Deterministic Analysis

We next present a set of theoretical results for Algorithm 1, including convergence results and stability analysis in a deterministic setting. We underscore that in [28], the analysis is mostly for the case of a distributional assumption on the Q-function errors. We shall tackle the probabilistic analysis for our current framework in the sequel. In addition to a series of results for the deterministic case; our analysis herein has some key differences from that in [28]. As previously noted, the incorporation of wireless networks and the use of co-link representations for the network dynamics enables us to exploit structures, facilitating tighter results. Additionally, given the consideration of direct policy comparison versus a divergence error metric for the Q-functions in [28], there are additional simplifications and tighter results possible. Finally, we observe that several results in [28] require assumptions and approximations. An example assumption is needing a particular time-varying update ratio to ensure convergence, whereas, herein, we can employ a constant update ratio $u_t = u$.

Proposition 1: Let $\Delta_t^{it}(s,a)$ denote the Q-function update of the output of Algorithm 1 for (s,a) from time t to $t+1$ as: $\Delta_t^{it}(s,a) = \mathbf{Q}_{t+1}^{it}(s,a) - \mathbf{Q}_t^{it}(s,a)$. Then,

$$\lim_{t\to\infty} |\Delta_t^{it}(s,a) - \Delta_{t-1}^{it}(s,a)| = 0, \tag{6}$$

for all (s,a). (See Appendix-A)

This proposition provides insight into the behavior of the Q-function updates over time. In particular, as more iterations are performed, the updates to the Q-function become increasingly stable, indicating the stabilization of the overall learning process. Our result leverages the convergence properties of the traditional Q-learning [9] and the convergence of the weights. We note that Proposition 3 [28] bounds the difference between the

Q-functions of the original Markovian environment and different SMEs; while Proposition 1 herein shows that the ensemble Q-function converges to a fixed Q-function, and consequently, the ensemble policy converges to a fixed policy.

Proposition 2: Let $\epsilon_t^{(n)}(s,a)$ be the weighted Q-function update of the n^{th} environment for (s,a) from time t to $t+1$ as: $\epsilon_t^{(n)}(s,a) = \mathbf{w}_{t+1}^{(n)}\mathbf{Q}_{t+1}^{(n)}(s,a) - \mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)}(s,a)$. Moreover, let $\theta^{(n)}(s,a)$ be the smallest positive constant that satisfies $|\epsilon_t^{(n)}(s,a)| \leqslant \theta^{(n)}(s,a)$ for all $t, n, (s,a)$. Then,

$$\lim_{t\to\infty} |\Delta_t^{it}(s,a)| \leqslant \sum_{n=1}^{K} \theta^{(n)}(s,a), \tag{7}$$

for all (s,a). (See Appendix-B)

The proposition provides an upper bound on $|\Delta_t^{it}(s,a)|$ in the limit depending on the characteristics of the weighted Q-function updates in each environment. The environments with larger $\theta^{(n)}$ values will have a greater impact on the bound. Furthermore, a tighter upper bound indicates more stable and controlled updates, potentially leading to faster convergence and improved learning efficiency.

Proposition 2 bounds the summed error over each Markovian environment and thus, this bound is implicitly a function of the number of Markovian environments, K. In contrast, Proposition 2 [28] shows that the scaling of the error variance between the ensemble Q-function and the optimal one is bounded by a constant that scales inversely with K. It should be noted that both propositions directly exploit the bounded nature of key functions albeit rather different ones: weighted Q-function errors and update ratio u here and the weights and the softmax operator in [28].

Corollary 1: It takes at most $t = \frac{\log\left(1-\frac{\beta}{\sum_{n=1}^{K}\theta^{(n)}(s,a)}\right)}{\log(u)}$ iterations to ensure that $|\Delta_t^{it}(s,a)| \leqslant \beta$ for any $\beta > 0$ and (s,a). (See Appendix-C)

By appropriately choosing the parameter u in Algorithm 1, we can effectively control the speed of convergence and the accuracy of the Q-function updates. In particular, a larger u leads to faster convergence.

Proposition 3: If there exist positive constants $\phi^{(n)}(s,a) \in (0,1)$ such that $\frac{|\epsilon_{t+1}^{(n)}(s,a)|}{|\epsilon_t^{(n)}(s,a)|} \leqslant \phi^{(n)}(s,a)$ for all $n, t, (s,a)$, then

$$\lim_{t\to\infty} |\Delta_t^{it}(s,a)| = 0 \tag{8}$$

for all (s,a). (See Appendix-D)

This proposition provides a sufficient deterministic condition for the convergence of Algorithm 1. If the weighted Q-function update of each Markovian environment is non-increasing over time, Algorithm 1 converges to the optimal Q-functions. We will show numerically for different settings that the constants $\phi^{(n)}(s,a)$ exist with high probability. We establish this result by utilizing the non-increasing nature of weighted Q-function errors under the assumption of constant u_t, while Corollary 2 of [28] relies on a strict time-varying form on the update ratio of the algorithm and independence assumptions on the stochastic Q-function errors.

Proposition 4: The weights $\mathbf{w}_t^{(n)}$ in Algorithm 1 converge to the following final weights:

$$\mathbf{w}^{(n)} = \frac{e^{\frac{1}{S}\sum_{j=1}^{|\mathcal{S}|}\mathbf{1}(\boldsymbol{\pi}^{(1)}(j)=\boldsymbol{\pi}^{(n)}(j))}}{\sum_{i=1}^{K} e^{\frac{1}{S}\sum_{j=1}^{|\mathcal{S}|}\mathbf{1}(\boldsymbol{\pi}^{(1)}(j)=\boldsymbol{\pi}^{(i)}(j))}}, \tag{9}$$

where $\boldsymbol{\pi}^{(n)} = \arg\min_{\boldsymbol{\mu}}(\mathbf{I} - \gamma\tilde{\mathbf{L}}_{\boldsymbol{\mu}}^{(n)})^{-1}\mathbf{c}_{\boldsymbol{\mu}}$ and $\tilde{\mathbf{L}}_{\boldsymbol{\mu}}^{(n)}$ is the l_1-normalized version of $\mathbf{L}_{\boldsymbol{\mu}}^{(n)}$ for all n.

This proposition shows that the final weights of Algorithm 1 can be computed without relying on the Q-functions. Furthermore, the final weights are non-monotonic across the order n, but the weight for the original system ($n=1$) is always the largest. We note that the order of weights can change across iterations, as will be shown numerically. This result can be easily shown using the compact expression for the Q-functions [9] and the fact that the weight vector $\mathbf{w}$ is normalized using the softmax operator in Algorithm 1. We herein have a closed-form solution to the convergent weights of Algorithm 1; thus, one can directly compare the relative importance of each Markovian environment. In contrast, Proposition 4 [28] provides a partial ordering between the Q-functions of any Markovian environment; thus, the results are less precise.

B. Probabilistic Analysis

The results above did not make any assumptions about the underlying distributions of the Q-function **errors**. Herein, as in [28], we assume that the Q-function errors follow a distribution. In particular, the distribution D has zero-mean and finite variance as follows:

$$\mathcal{X}_t^{(n)}(s,a) \stackrel{\text{def}}{=} \mathbf{Q}_t^{(n)}(s,a) - \mathbf{Q}^*(s,a) \sim D\Big(0, \frac{\lambda_n^2}{3}\Big), \tag{10}$$

for all $n, (s,a)$ with $\lambda_n > 0$ where $\mathbf{Q}^*$ is the optimal Q-functions of the original Markovian environment. Unlike prior studies that assumed specific distributions for D [38], we take a more general approach by adopting a common distribution family that governs the Q-function errors across all environments, as employed and validated in [27], [28]. This assumption is reasonable due to the shared state and action space, the use of the same cost function, and common learning parameters across all Markovian environments. Simulations also confirm that the true distribution D exhibits a zero-mean and finite variance for all n (see Fig. 8(g)). A key difference between our current work and our prior work [28] is that we herein consider a variety of different independence assumptions on the errors between different Markovian environments, which capture the dynamics of different wireless network environments more effectively relative to [28].

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

Proposition 5: Let u_t be constant: $u_t = u$. Under assumption (10), Algorithm 1 produces unbiased Q-functions in the limit *i.e.* $\lim_{t\to\infty} \mathbb{E}[\mathcal{E}_t(s,a)] = 0$ for all (s,a), and the upper bounds on the error variance in the limit based on different independence assumptions are given in Table I. (See Appendix-E)

TABLE I
UPPER BOUNDS ON $\lim_{t\to\infty} \mathbb{V}[\mathcal{E}_t(s,a)]$

Independence Assumption	Upper Bound
$\mathcal{X}_t^{(n)} \perp \mathcal{X}_t^{(m)} \forall t,\ \mathcal{X}_{t_1}^{(n)} \perp \mathcal{X}_{t_2}^{(n)} \forall n$ (strict)	$\frac{(1-u)}{(1+u)} \frac{\lambda^2}{3}$
$\mathcal{X}_{t_1}^{(n)} \perp \mathcal{X}_{t_2}^{(n)} \forall n$ (modest)	$\frac{(1-u)}{(1+u)} \lambda^2$
no independence assumption	$\frac{2\lambda^2}{(1+u)^2} + \frac{(1-u)}{(1+u)} \lambda^2$

This proposition shows that under the assumption (10), $\mathbf{Q}_t^{it}$ is an unbiased estimator of $\mathbf{Q}^*$ in the limit, and the variance of the error can always be upper bounded. Clearly, the stricter independence assumption leads to a tighter bound. Herein, a larger λ implies a higher uncertainty in the Q-function errors, yielding looser upper bounds. On the other hand, when the algorithm converges ($t \to \infty$), a larger u leads to less reliance on SMEs. We note that when there is high uncertainty in the system, upon initialization, the digital cousins provide observations at multiple time scales, improving exploration. However, it should be underscored that the optimal Q-functions for each SME may not be the same, and be different from that of the original Markovian environment. Thus, as uncertainty diminishes, we want a lower reliance on SMEs and, thus, a larger u.

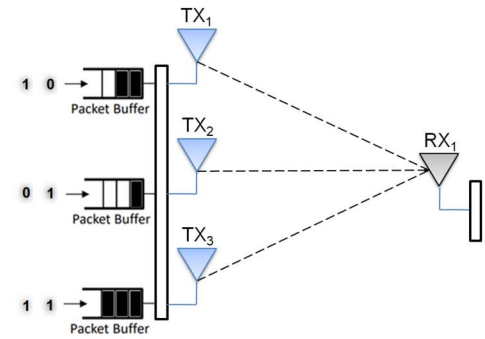
Corollary 2: By assuming a specific time-varying structure on the parameter u_t, as stated in Proposition 5, and an independence assumption on the Q-function errors, it is possible to derive a sufficient probabilistic condition for convergence for Algorithm 1. This condition can be employed to show convergence when Proposition 3 fails to apply [27].

IV. NUMERICAL RESULTS

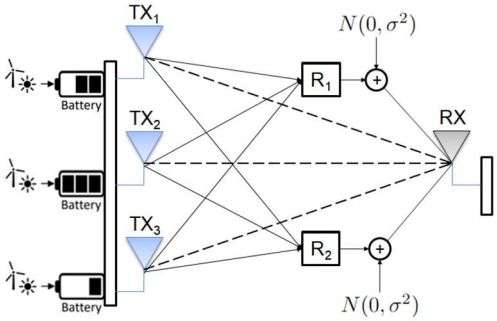
In this section, we evaluate the accuracy and complexity of the proposed algorithm in wireless networks, considering four distinct wireless network models. These models vary in their topology, scale, complexity, objective, and implementations and serve as representative examples that accurately depict the complexities and challenges of real-world wireless networks. The detailed descriptions and parameter optimizations can be found in [39]. We assume that data arrivals to each transmitter in the first three network models follow an independent and identically distributed distribution, such as Bernoulli as in [31].

A. Wireless Network Models

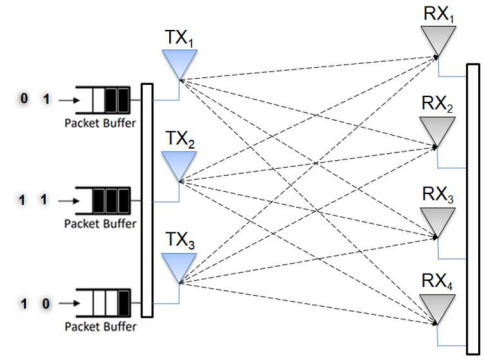
1) MISO Network With Interference Channels: There are multiple transmitters (TX_1, TX_2, TX_3) and a single receiver (RX) as shown in Fig. 3(a). Each transmitter has a finite-length buffer that stores the incoming data packets, and the channel between each transmitter and receiver is modeled with a multi-state Gilbert-Elliot channel model [31] with different parameters. Each transmitter experiences its own channel to the receiver; furthermore, collisions can occur due to interference between transmitters transmitting simultaneously. The magnitude of interference is inversely proportional to the distance between transmitters. The objective is to determine when each transmitter should *transmit* data or *remain silent* given the buffer load of each transmitter and overall channel conditions in order to minimize the overall cost. The cost function has three components: the buffer cost, the channel cost (incurred when the transmitter transmits under unfavorable channel conditions),



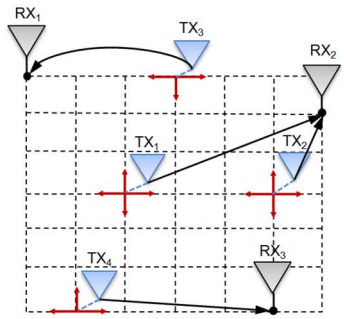
(a) Model 1: MISO network with interference channels



(b) Model 2: MISO energy harvesting network with Gaussian interference channels and multiple relays



(c) Model 3: MIMO network with interference channels



(d) Model 4: MIMO network with mobile transmitters

Fig. 3. Examples of wireless network models.

and the collision cost (proportional to the number of data packets being transmitted simultaneously).

2) MISO Energy Harvesting Network With Multiple Relays: There are multiple transmitters (TX_1, TX_2, TX_3), multiple relays (R_1, R_2), and a single receiver (RX) as shown in Fig. 3(b). Each transmitter is equipped with a finite-sized battery that stores the harvested energy packets in discrete units. All the channels are modeled with a standard Gilbert-Elliot model, and the channel parameters are chosen such that the channel between transmitters and relays as well as relays and receivers are more likely to be in a good state than the direct channel between transmitters and receivers (due to shorter distance and less interference). The outputs of the relays are corrupted by AWGN noise. The objective is to determine when transmitters should *directly* transmit or transmit *through relays* given the battery load of each transmitter and overall channel conditions in order to minimize the overall cost. The cost function consists of three components: the negative throughput (inversely proportional to successfully received packets), the drop cost (balancing relays' load), and the battery consumption cost (proportional to total battery usage).

3) MIMO Network With Interference Channels: We consider a MIMO network model with multiple transmitters (TX_1, TX_2, TX_3), and multiple receivers (RX_1, RX_2, RX_3, RX_4) as shown in Fig. 3(c). Each transmitter is equipped with a finite-sized data buffer for storing incoming data packets. All the channels are modeled with a standard Gilbert-Elliot model, and the channels between closer TX-RX pairs (such as TX_1 and RX_1) are more likely to be in a good state compared to the channels between farther TX-RX pairs (such as TX_1 and RX_4) due to shorter distance and less interference. Each channel allows only one data transmission at a time. If a transmitter needs to send data to multiple receivers, the channels in the best conditions will be used to optimize transmission. Transmitters sending packets to the same receiver cause interference with each other. The magnitude of interference is inversely proportional to the distance between the transmitters. The objective is to determine *how many data packets* each transmitter should transmit given the buffer load of each transmitter and overall channel conditions in order to minimize the overall cost. The cost function comprises four components: the buffer cost, the channel cost, the collision cost, and the receiver load cost (balancing receivers' load).

4) MIMO Network With Mobile Transmitters: We consider a MIMO network model with multiple mobile transmitters (TX_1, TX_2, TX_3, TX_4) and multiple fixed receivers (RX_1, RX_2, RX_3) as shown in Fig. 3(d). Each transmitter moves randomly at different constant speeds within possible directions, as indicated by the red arrows. Their movement is constrained within a specified bounded area. They can only move in integer multiples of unit steps and are not allowed to occupy the same location as receivers. The channel between the pair of transmitters and receivers follows a path-loss model. The objective is to determine *the optimal association between transmitters and receivers* given the speed of transmitters, the positions of transmitters and receivers, and the overall channel conditions in order to minimize the overall cost. The black arrows illustrate one possible association. The cost function comprises three components: the negative throughput, the receiver load cost, and the interference cost (inversely proportional to the squared distance between transmitters transmitting to the same receiver).

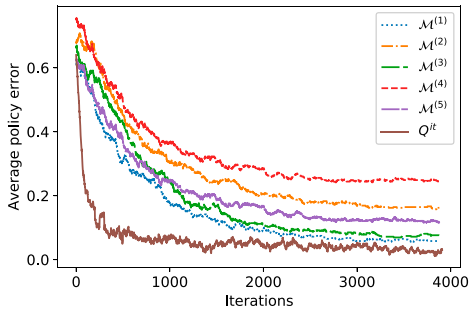


Fig. 4. APE performances across different environments.

B. Average Policy Error (APE) Performance

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

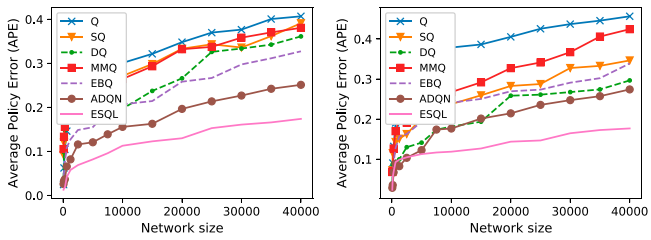
$$APE = \frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \mathbf{1}\left(\boldsymbol{\pi}^*(s) \neq \hat{\boldsymbol{\pi}}(s)\right). \tag{11}$$

We analyze the APE performance of Algorithm 1 in comparison to the traditional Q-learning running on the original Markovian environment $\mathcal{M}^{(1)}$ and four different SMEs $\mathcal{M}^{(n)}$ for $n = \{2, 3, 4, 5\}$ in Fig. 4, where Q^{it} represents the APE of Algorithm 1. The simulations are conducted using the MIMO network with interference channels with a network size of 40000 using the following parameter values: $K = 5$, $u_t = 0.5$, $v = 50$, $l = 15$, $\gamma = 0.99$, $\alpha_t = \frac{1}{1+\frac{t}{1000}}$, $\epsilon_t = \min(0.99^t, 0.01)$, $\lambda = 1$, where λ_n for $n \in \{1, 2, 3, 4, 5\}$ are estimated numerically. The parameters are optimized through cross-validation (see [39] for details). The simulation is carried out 50 times, and the APE results are averaged. Clearly, a near-zero APE can be achieved with a significantly small number of iterations (less than 0.1 APE within 500 iterations), as can be seen in Fig. 4. The sharp decline in the Q^{it} curve at the initial stages corresponds to the *exploration* phase, followed by the *exploitation* phase. Compared to other Q-learning algorithms, the exploration stage in Algorithm 1 is significantly fast, highlighting the advantages of leveraging multiple SMEs to enhance exploration capabilities. Furthermore, the APE results demonstrate a non-monotonic pattern across various n values, with the original environment $\mathcal{M}^{(1)}$ consistently achieving the smallest APE, as predicted by Proposition 4.

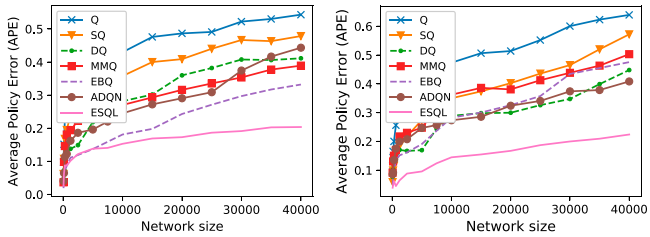
For comparison, we consider several value-based model-free reinforcement learning algorithms with different objectives and implementation strategies (number of estimators and Markovian environments). To ensure a fair comparison, all algorithms follow the same strategy as Algorithm 1. Table II provides an overview of these algorithms. For specific details on the parameter optimization of each algorithm, see [39].

TABLE II
Q-LEARNING ALGORITHM AND VARIANTS

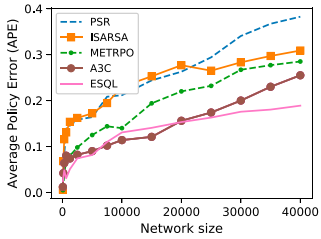
Algorithm	Objective	Strategy	
		Estimator	Environment
Simple Q (Q) [9]	-	Single	Single
Speedy Q (SQ) [18]	Convergence rate	Single	Single
Double Q (DQ) [14]	Bias	Multi	Single
MaxMin Q (MMQ) [17]	Bias & variance	Multi	Single
Ensemble Bootst. Q (EBQ) [15]	Bias	Multi	Single
Averaged DQN (ADQN) [16]	Stability, Variance	Multi	Single
Ensemble Synthetic Q (ESQL)	**Variance, Learning speed**	**Multi**	**Multi**



(a) Q-learning algorithms for Model-1(b) Q-learning algorithms for Model-2



(c) Q-learning algorithms for Model-3(d) Q-learning algorithms for Model-4



(e) Non-Q-Learning algorithms for Model 3

Fig. 5. APE of different algorithms across different models.

The APE of different algorithms as a function of network size across different network models are given in Fig. 5(a)–5(d). The proposed algorithm consistently outperforms other algorithms, achieving a lower APE across all models. In particular, it achieves 30% less APE for model-1, 40% less APE for model-2, 45% less APE for model-3, and 50% less APE for model-4. This performance improvement can be attributed to several reasons. Firstly, the utilization of multiple Markovian environments allows for deep and efficient exploration by aggregating higher-order relationships between states into a single estimate. Secondly, the adaptive weighting mechanism, based on policy comparison, exploits the most informative environments during training by assigning higher weights, unlike ensemble methods such as MMQ, DQ, or EBQ. Lastly, the algorithm leverages structural similarities among multiple Markovian environments, providing an advantage over neural network-based algorithms like ADQN that do not utilize such properties. The APE gains become larger for larger networks, which highlights the practicality of the algorithm in large-scale wireless networks.

We also compare the performance of the proposed algorithm with the algorithms presented in our prior work [26], [27], [28] where the co-link representations are not used, and the cost-metric is the divergence between Q-functions. To ensure a fair comparison, we simulate each algorithm on the MIMO wireless network with interference channels with a network size of 40000. The parameters in Section IV-B are employed for the proposed algorithm. For the other algorithms, we individually optimize the hyperparameters by cross-validation. Extensive simulations demonstrate that the proposed algorithm achieves a 15% less APE (with 10% less runtime) compared to [26], a 15% less APE (with 20% less runtime) compared to [27], and a 10% less APE (with 10% less runtime) compared to [28]. Furthermore, we observe that varying network parameters, such as arrival probability, buffer size, and the number of transmitters and receivers within a reasonable range, lead to a 25% change in APE for [26], a 20% change in APE for [27], a 20% change in APE for [28], and only a 5% change in APE for the proposed algorithm, which indicates that the proposed algorithm is the most robust amongst our prior approaches. These results highlight that the proposed algorithm is particularly well-suited for optimizing wireless networks due to their unique structural properties [31].

We also compare to *non-Q-learning RL* algorithms: (i) Value function-based policy sampling and reconstruction (PSR) [40], (ii) Improved Sarsa (ISARSA) [41], (iii) Model-ensemble trust-region policy optimization (METRPO) [42], and (iv) Asynchronous Advantage Actor Critic (A3C) [32]. The simulations are carried out using the same settings in Section IV-B. See [39] for further details.

ESQL achieves up to 25 % less APE than other algorithms across large networks, as demonstrated in Fig. 5(e). PSR faces challenges in accurately estimating the value function, sensitivity to network parameters, and the need for near-perfect estimation of the PTT $\hat{\mathbf{P}}$. Although METRPO utilizes an ensemble of deep neural networks, highlighting the power of ensemble learning, it does not leverage the structural properties of multiple Markovian environments and faces challenges in generalization across large networks. A3C produces a similar performance to ESQL (even slightly outperforms it for small network sizes) as both methods employ parallel learning. While there is no clear performance difference for small and modest-sized networks, ESQL achieves up to 25% less APE for large networks because A3C suffers from the limitations of digital twins, the challenges to achieve fine-tuned training and accurate generalization for large networks, and the lack of utilization of the structure of the underlying network. Finally, as will be seen, A3C achieves its performance at the expense of high computational complexity.

C. Average Runtime Complexity Performance

The average runtime complexity of Algorithm 1 can be shown to be $O\left(\frac{|\mathcal{S}||\mathcal{A}|v}{K} f(l,\epsilon)\right)$, where f is a non-monotonic function of l and ϵ. The proof of the runtime complexity of algorithms presented in [26], [27] is applicable here. The runtime complexity increases with the network size ($|\mathcal{S}|$ and $|\mathcal{A}|$) and the number of visits to each state-action pair (v). However, the non-monotonicity of the function f with respect to l and ϵ suggests the existence of optimal values for these parameters, necessitating parameter-tuning. The complexity is also inversely proportional to the number of Markovian environments (K), which may seem counter-intuitive. This result follows from the fact that the number of samples that need to be collected from each Markovian environment decreases with K [26], [27].

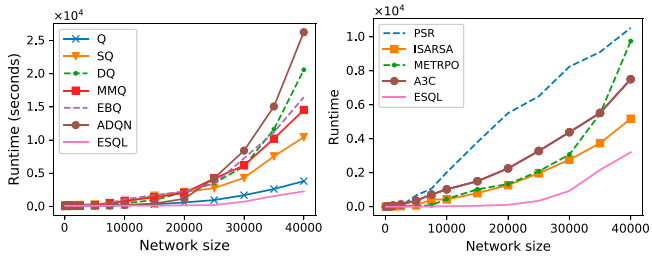
The runtime of the proposed algorithm consists of three main components: sampling (ensuring each state-action pair is visited at least v times), constructing multiple SMEs $K-1$ times using (5), and the time until the slowest Q-learning algorithm on different SMEs converge (as they run in parallel). For other algorithms, their runtime represents the time until convergence. The runtimes of algorithms are consistently comparable across fixed network sizes and different network models. Thus, a single runtime result is presented in Fig. 6(a). The proposed algorithm achieves 40% less runtime than the other algorithms across large state-spaces as it utilizes higher-order SMEs to capture distant node relationships, reducing the need for long trajectories that are required for small-order SMEs. In addition, the algorithm improves exploration by running multiple Markovian environments simultaneously. This result is consistent with both Fig. 4 and the theoretical average runtime complexity.

We also present the runtime complexity of our algorithm compared to non-Q-learning algorithms (in Section IV-B) in Fig. 6(b). Overall, the proposed algorithm achieves up to 40% less runtime complexity than other algorithms across large networks. ISARSA is a simple algorithm and thus has relatively lower complexity than the other algorithms. In contrast, PSR exhibits high complexity due to increased sampling to obtain an accurate estimate of PTT and policy interpolation, and neural-network-based approaches (A3C and METRPO) suffer from long training for multiple networks.

We underline that as the algorithm complexity increases (e.g., Q → SQ → ADQN), the corresponding APE generally decreases, but the runtime complexity increases. However, the proposed algorithm achieves a small APE with a small runtime, overcoming the performance-complexity trade-off. We also see that the complexity reduction is independent of the network model, making the proposed algorithm an efficient approach for learning various complex environments.

D. Memory Complexity Analysis

Memory complexity is a critical concern in our algorithm, as we need to store Q-functions for K different environments. This issue is a common drawback in similar algorithms [14], [15], [17], yet it is exacerbated here as a result of K different environments. To address this concern, we introduced a state-action aggregation method in our previous work [26]. This approach



(a) Q-learning algorithms (b) Non-Q-learning algorithms

Fig. 6. Runtime of different algorithms.

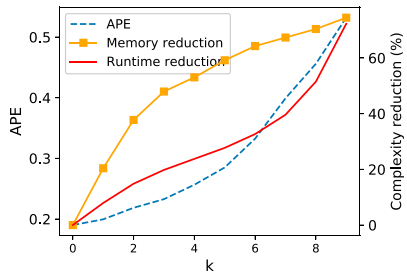


Fig. 7. APE and memory reduction through state aggregation vs k.

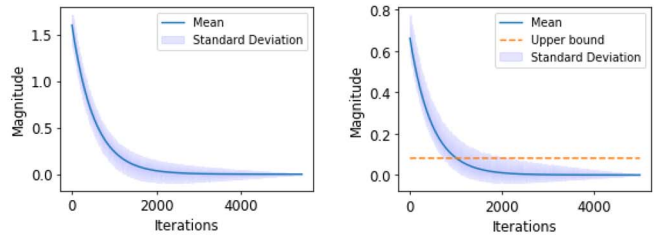
is based on the intuition that, under a smooth cost function, the Q-functions of neighboring states, for a given action, exhibit minimal differences. In other words, if the changes in the cost function for neighboring states are bounded by a small constant, we can represent the Q-functions of the k nearest neighboring states (including the state itself) using a single Q-function. Choosing a large value for k minimizes memory requirements and runtime complexity (since the size of state space is reduced) but comes at the cost of information loss, leading to a larger APE. Thus, a trade-off exists between the APE, and memory needs as well as runtime complexity.

The simulations are carried out using the same settings in Section IV-B. (One can observe that the cost function of the network model meets the specified condition.) The results are given in Fig. 7, with the y-axes showing the APE and the percentage reduction in complexity (memory and runtime), respectively, while the x-axis represents the number of nearest neighbors k. For clarity, the case $k=0$ implies no aggregation. While the APE increases with k, the reduction in the memory needs and runtime complexity also increases. Thus, by appropriately choosing k, a good performance-complexity balance can be achieved. We also emphasize that the proposed state-action aggregation idea can similarly minimize the memory requirements for all four wireless network models.

E. Numerical Consistency of Propositions

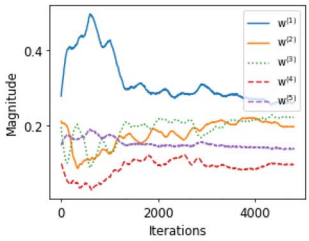
In this section, we simulate the results in the propositions and compare theoretical results with simulation results. The same simulation settings in Section IV-B are employed.

The magnitude of the difference between the consecutive Q-function updates for the state-action pair (s,a) averaged over

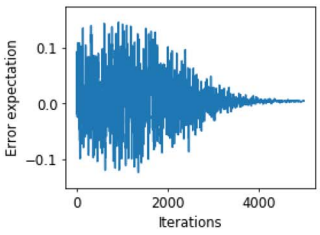


(a) $|\Delta_t^{it}(s,a) - \Delta_{t-1}^{it}(s,a)|$ averaged overall (s,a) vs iterations

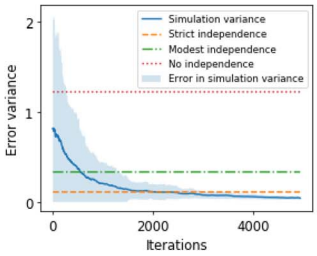
(b) $|\Delta_t^{it}(7,1)|$ vs iterations



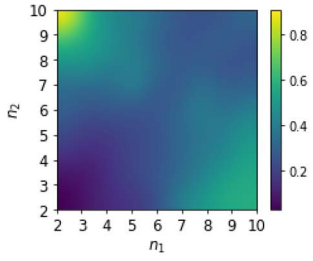
(c) Weights across iterations



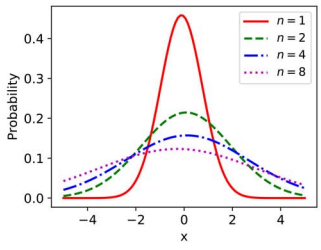
(d) Expectation of error



(e) Upper bounds on error variance



(f) Averaged distance correlation



(g) Normal fit to $\mathcal{X}_t^{(n)}(7,1)$ for different n

Fig. 8. Simulation of theoretical results.

all state-action pairs over time is shown in Fig. 8(a), where the blue curve and shaded area represent the mean and standard deviation over 50 simulations. The magnitude consistently converges to zero with increasing iterations, as shown by Proposition 1, which shows that Algorithm 1 produces increasingly stable updates over time. We note that similar curves with different initial magnitudes and decay rates can also be shown for different (s,a) pairs.

The magnitude of the Q-function updates for the state-action pair $(s,a) = (7,1)$ over time, $|\Delta_t^{it}(7,1)|$, is depicted in Fig. 8(b). The blue curve and shaded area represent the mean and standard deviation over 50 simulations. Additionally, the dotted curve illustrates the upper bound on $|\Delta_t^{it}(7,1)|$ from Proposition 2. Clearly, the magnitude of the updates in the limit can be upper bounded. The tightness of the upper bound depends on the state-action pair and the system parameters (including K). Furthermore, the result shows that the magnitude of the updates converges to zero as the iterations progress, thereby confirming the convergence of Algorithm 1 as stated in Proposition 3. To further validate the convergence result and practicality of Proposition 3, we observe that the constants $\phi^{(n)}(7,1)$ for $n = 1,2,3,4,5$ exist and are all less than 1. (In particular, the values are 0.998, 0.9925, 0.9898, 0.9946, and 0.9808, respectively.)

The weights of five different Markovian environments ($\mathcal{M}^{(n)}$ for $n = 1,2,3,4,5$) over time are illustrated in Fig. 8(c). Initially, there is a sharp increase in the weight $\mathbf{w}^{(1)}$ because $\mathcal{M}^{(1)}$ is the original environment, and there are insufficient samples to capture the higher-order relationships. Additionally, it is unclear which $\mathcal{M}^{(n)}$ provides the most useful samples, as the weights $\mathbf{w}^{(n)}$ for $n > 1$ continue to fluctuate. 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 the iterations continue, the weight $\mathbf{w}^{(1)}$ gradually decreases up to a certain point, but $\mathcal{M}^{(1)}$ remains the most useful environment. On the other hand, the weights $\mathbf{w}^{(n)}$ for $n > 1$ increase and eventually converge to a fixed value. The final magnitudes of these weights, however, exhibit a non-monotonic pattern across n, and can be shown to follow the compact expression in Proposition 4. Similar patterns can be observed across different networks and models, although there may be variations in (i) the final weight values, (ii) the order of environment utilities, and (iii) the iteration index at which the weights converge.

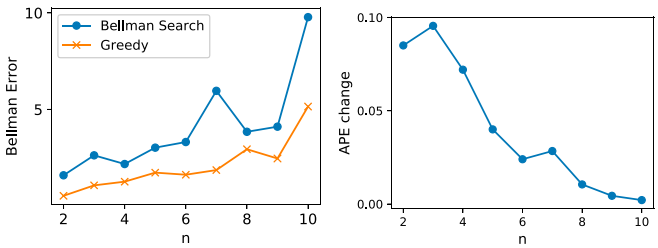
We approximate the expectation of the Q-function errors numerically as follows:

$$\mathbb{E}[\mathcal{E}_t(s,a)] \approx \frac{1}{2\Delta_t} \sum_{t'=t-\Delta_t}^{t+\Delta_t} \mathcal{E}_t(s,a), \tag{12}$$

with $\Delta_t \ll t$. The expectation of the Q-function error for the state-action pair $(s,a) = (7,1)$, $\mathbb{E}[\mathcal{E}_t(7,1)]$ over time, in Proposition 5 is shown in Fig. 8(d), where the result is averaged over 50 simulations. Clearly, the expectation of the Q-function errors converges to zero with iterations. We approximate the variance of the Q-function errors similarly 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 three upper bounds on the error variance based on different independence assumptions from Proposition 1 and the simulation variance for $(s,a) = (7,1)$ (with $\Delta_t = 20$) are shown in Fig. 8(e), where the blue curve and shaded area represents the mean and standard deviation over 50 simulations. Clearly, a more strict independence assumption leads to a tighter upper bound. Moreover, as iterations continue, the simulation variance becomes smaller than all upper bounds and eventually converges to zero, which is in line with Fig. 8(a) and 8(b). Herein, the choice of λ and u may change the initial error variance and its decay rate. These two results (the expectation and variance converging to zero) show that the distribution assumption (10) is accurate. Similarly, Fig. 8(g) demonstrates



(a) Bellman error vs order n (b) APE change vs order n

Fig. 9. The effect of the order n on performance.

that the Q-function errors of different environments, across various values of n, can be well-modeled by zero-mean normal distributions with different variances, which further validates the assumption (10).

To assess the practicality of Proposition 5 and independence assumptions, we employ the averaged distance correlation (ADC) metric [43]. In particular, we compute the ADC between $\mathcal{X}_{t_1}^{(n_1)}(7,1)$ and $\mathcal{X}_{t_2}^{(n_2)}(7,1)$ for $n_1, n_2 \in [2,10]$ and average the results over all $t_1 \neq t_2 \in [0, 5000]$. The ADC captures both linear and non-linear correlations without assuming specific variable distributions like Pearson's correlation coefficient. The results are presented in Fig. 8(f). For all $n_1 = n_2$ (right diagonal), the ADC is very close to 0, indicating independent Q-function errors across different times, validating the modest independence assumption. When n_1 and n_2 are close, the ADC is almost 0, also confirming the strict independence assumption. However, for larger and distant n_1 and n_2, weak correlations may emerge. Thus, the strict independence assumption may not always hold contrary to the modest independence assumption.

F. Optimization of the Order n

Determining the optimal set of environments involves determining the optimal number of environments as well as their identity (e.g. for $K = 4$, $\{1^{st}, 2^{nd}, 3^{rd}, 4^{th}\}$ versus $\{1^{st}, 2^{nd}, 3^{rd}, 5^{th}\}$ versus all other collections of four possible environments). We offer a strategy in [28] for arbitrary graphs/networks, whereas herein, we utilize an approach from [31] where we proposed the colink representation for wireless network policy optimization. In particular, we adapt the classical definition of the Bellman error (BE) (see [44], [45]) to the difference between the Q-functions of the original environment and n^{th} environment under the policy π as:

$$BE(\mathbf{Q}_{\boldsymbol{\pi}}^{(n)}) = \mathbf{c}_{\boldsymbol{\pi}} + \gamma \mathbf{P}_{\boldsymbol{\pi}} \mathbf{Q}_{\boldsymbol{\pi}}^{(n)} - \mathbf{Q}_{\boldsymbol{\pi}}^{(1)}. \tag{14}$$

In [31], the true PTT is known; herein, we apply the algorithm presented in [31] to the estimated PTT ($\hat{\mathbf{P}}$) to obtain the approximate policy to be used in (14). We then select the top K orders (environments) that minimize the l_2 norm $\|BE(\mathbf{Q}_{\boldsymbol{\pi}}^{(n)})\|_2$.

We present the Bellman error across $n \in [2,10]$ in Fig. 9(a) using two methods: (i) Bellman search (blue curve) using [31] and (14) and (ii) Greedy method (orange curve) that exhaustively searches the environment and policy that minimize the APE of Algorithm 1. The same simulation settings in Section IV-B are employed. The Bellman error clearly exhibits a non-monotonic behavior with n, but small values of n generally produce smaller errors. While the blue curve consistently shows higher Bellman errors for all n (since it uses an approximate policy), the relative ranking of orders is similar across the two methods, which suggests that the Bellman search can effectively rank environments. For $K = 3$, Bellman search provides an ordered set {2,4,3} while the greedy approach selects {2,3,4}. Thus, both methods find the same set of environments, but the relative informativeness for environments 3 and 4 are flipped. Similar trends apply to different K values. Overall, the Bellman search achieves 94% accuracy in finding the optimal K environments (for $K \in [2,10]$) and has 70% less runtime complexity than the greedy method, hence enabling an effective initialization of Algorithm 1, which is particularly useful for real-world wireless applications with time, memory or resource constraints.

The sensitivity of Algorithm 1 with respect to order n is shown in Fig. 9(b). In particular, we initialize Algorithm 1 with the first ten Markovian environments ($K = 10$). Then, we run the algorithm with $K = 9$ after excluding the n^{th} environment for $n \in [2,10]$ and report the absolute change in APE. While APE changes vary with n, the maximum change in APE is under 10%, highlighting the robustness of Algorithm 1 with respect to n. The result also closely follows Fig. 9(a) as the set of orders that lead to the largest change in APE are the same as the orders that minimize the Bellman error. Similar results can also be shown for different network settings.

G. Practical Implications and Interpretability

By integrating the novel concept of digital cousins into ensemble Q-learning, the proposed algorithm effectively accounts for the influence of actions and states that span **multiple hops** in the wireless network graph. This addresses scenarios like multiple packet changes in transmitter buffers, significant channel quality variations, or multi-step movements of mobile transmitters. This could enhance the performance, reliability, and robustness of real-world wireless networks.

When there is a high probability of multiple data arrivals within a short period, considering buffer occupancy changes beyond a single data packet (n^{th} digital cousin (environment) considers n packet changes at a time) helps transmitters proactively adjust their transmission strategies to avoid buffer congestion, leading to more stable buffer occupancy levels. If there is a high probability of consecutive data losses, transmitters can dynamically adjust the transmission rate to maintain a reliable communication link. This can improve overall network robustness, particularly for densely populated networks with many transmitters and receivers. On the other hand, when there is a high probability of the channel deteriorating in the next time steps, transmitters must avoid making sub-optimal decisions. Without considering n-hop transitions, transmitters might assume the good channel conditions will persist and continue transmitting data continuously. However, by incorporating n-hop transition information, transmitters can intelligently anticipate the potential deterioration in the channel in the future and adjust their transmission rate to avoid unnecessary energy

consumption. Similarly, more balanced traffic loads at receivers can be achieved via traffic smoothing by considering n-step movements and future locations of transmitters in wireless networks with mobile transmitters.

H. Open Questions

The proposed algorithm provides a novel implementation of digital cousins via the colink method and enables reductions in sample and runtime complexities as well as performance improvements over the consideration of general graphs [28]. As noted previously, one open question is designing a modest complexity strategy for determining the optimal number of environments (K) to employ in Algorithm 1. While more environments improve performance and reduce runtime complexity, there are diminishing returns as seen [28]. ESQL inherits several properties from traditional Q-learning and hence is particularly tailored to finite but large discrete state-action spaces. Thus, another open question is addressing continuous state-action spaces; thus discretization or approximation via neural networks is an option. However, we see that given the structural properties of wireless networks, a pure non-parametric approach such as a neural network also has its own limitations. The quality of sampling and the accuracy of the estimated PTT also impacts performance as discussed in Section II-C; whether further improvements are possible, given that our current strategy explicitly exploits the structure in the PTT and associated colink matrices, is unclear. There are also design issues involved in determining an appropriate normalization strategy to form the digital cousins, as colink matrices are not necessarily row-stochastic. Hence, one can consider alternative methods such as ℓ_1, ℓ_2, linear, softmax, min-max, etc).

V. Conclusion

We herein presented a novel Q-learning algorithm to mitigate the performance and complexity challenges of the traditional Q-learning to solve policy optimization for real-world wireless networks by leveraging the novel concept of *digital cousins*. Digital cousins are a collection of distinct but structurally related synthetic Markovian environments that offer strong improvements over traditional digital twins. The current work synthesizes the notion of digital cousins with low complexity, high-performance modeling of wireless networks via co-link representations. The proposed algorithm employs multiple Q-function estimators on these environments and fuses the outputs into a single estimate based on an adaptive weighting mechanism. Simulations across a variety of representative wireless networks show that the proposed algorithm produces a near-optimal policy with significantly lower complexity and outperforms the existing Q-learning and several state-of-the-art RL algorithms and prior work in terms of accuracy, runtime complexity, and robustness. The stability and convergence of the algorithm are rigorously analyzed from deterministic and probabilistic viewpoints, and several theoretical upper bounds on the first and second-order statistics of the Q-function errors are given. It is also shown that the simulation results closely follow the theoretical results.

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. The Q-function output of Algorithm 1 can be expressed as follows:

$$\mathbf{Q}_t^{it} = (1-u)\sum_{i=0}^{t-1} u^{t-i-1}\sum_{n=1}^{K}\mathbf{w}_i^{(n)}\mathbf{Q}_i^{(n)}, \tag{15}$$

which can be obtained by repeatedly plugging the expression of $\mathbf{Q}_{t-1}^{it}$ into the expression of $\mathbf{Q}_t^{it}$ in line 11 in Algorithm 1 for all t. Using (15) and algebraic manipulations, the following can be shown:

$$\mathbf{Q}_t^{it} - u\mathbf{Q}_{t-1}^{it} = (1-u)\sum_{n=1}^{K}\mathbf{w}_{t-1}^{(n)}\mathbf{Q}_{t-1}^{(n)}. \tag{16}$$

Let $\Delta_t^{it} = \mathbf{Q}_{t+1}^{\mathbf{it}} - \mathbf{Q}_t^{\mathbf{it}}$ and $\epsilon_t^{(n)} = \mathbf{w}_{t+1}^{(n)}\mathbf{Q}_{t+1}^{(n)} - \mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)}$. If we rewrite (16) with the variable change $t \to t+1$, and subtract (16) from the new expression side by side, we obtain the following:

$$\Delta_t^{it} - u\Delta_{t-1}^{it} = (1-u)\sum_{n=1}^{K}\left[\mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)} - \mathbf{w}_{t-1}^{(n)}\mathbf{Q}_{t-1}^{(n)}\right]. \tag{17}$$

Then, we can show the following:

$$\Delta_t^{it} = u\Delta_{t-1}^{it} + (1-u)\sum_{n=1}^{K}\epsilon_{t-1}^{(n)}. \tag{18}$$

$$< \Delta_{t-1}^{it} + (1-u)\sum_{n=1}^{K}\epsilon_{t-1}^{(n)}, \tag{19}$$

where (18) follows from (17) and the definition of $\epsilon_{t-1}^{(n)}$ and (19) follows from the fact that $u \in (0, 1)$. Then, using (19) and the triangle inequality, we can bound the difference between consecutive updates as follows:

$$|\Delta_t^{it} - \Delta_{t-1}^{it}| < (1-u)\sum_{n=1}^{K}|\epsilon_{t-1}^{(n)}|. \tag{20}$$

By the convergence of Q-learning, $\lim_{t\to\infty}\mathbf{Q}_{t-1}^{(n)} = \mathbf{Q}_t^{(n)}$, and by the convergence of the weights from Proposition 4, $\lim_{t\to\infty}\mathbf{w}_{t-1}^{(n)} = \mathbf{w}_t^{(n)}$ for all n. Thus, $\lim_{t\to\infty}|\epsilon_{t-1}^{(n)}| = 0$ for all n, and $\lim_{t\to\infty}|\Delta_t^{it} - \Delta_{t-1}^{it}| = 0$.

B. Proof of Proposition 2

If we express (17) with the variable change $t \to t-1$, multiply by u, and then add to the expression (17) side by side, we obtain the following:

$$\Delta_t^{it} - u^2\Delta_{t-2}^{it} = (1-u)\sum_{n=1}^{K}\epsilon_{t-1}^{(n)} - u\epsilon_{t-2}^{(n)}. \tag{21}$$

If we carry out the same operation for all t, we obtain the following:

$$\Delta_t^{it} - u^t\Delta_0^{it} = (1-u)\sum_{k=0}^{t-1}u^k\sum_{n=1}^{K}\epsilon_{t-k-1}^{(n)}. \tag{22}$$

Using the fact that $\Delta_0^{it} = 0$, we obtain the compact expression for Δ_t^{it} as follows:

$$\Delta_t^{it} = (1-u)\sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \epsilon_{t-k-1}^{(n)}. \tag{23}$$

Let $\theta^{(n)}$ be the smallest positive constant that satisfies $|\epsilon_t^{(n)}| \leqslant \theta^{(n)}$ for all t, n. Then, we proceed as follows:

$$|\Delta_t^{it}| \leqslant (1-u)\sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} |\epsilon_{t-k-1}^{(n)}|. \tag{24}$$

$$\leqslant (1-u)\sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \theta^{(n)}. \tag{25}$$

$$\leqslant \sum_{n=1}^{K} \theta^{(n)}(1-u^t), \tag{26}$$

where (24) follows by taking the absolute value of both sides in (23) and using the triangle inequality, (25) follows the definition of $\theta^{(n)}$, and (26) follows from the sum of finite geometric series and the fact that $u \in (0,1)$. If we take the limit of both sides, using the fact that $\lim_{t\to\infty} u^t = 0$, the result follows.

C. Proof of Corollary 1

We want to find the time index t at which the $|\Delta_t^{it}| \leqslant \beta$. If we convert (26) into equality and solve for t as follows, we obtain the desired result: $\beta = \sum_{n=1}^{K} \theta^{(n)}(1-u^t)$.

D. Proof of Proposition 3

Assume there exist constants $\phi^{(n)} \in (0,1)$ such that $\frac{|\epsilon_{t+1}^{(n)}|}{|\epsilon_t^{(n)}|} \leqslant \phi^{(n)}$ for all n, t. We choose $\phi = \max_{n\in\{1,2,..K\}} \phi^{(n)}$, and proceed as follows:

$$|\Delta_t^{it}| \leqslant (1-u)\sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \left|\epsilon_{t-k-1}^{(n)}\right| \tag{27}$$

$$\leqslant (1-u)\sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \left|\epsilon_0^{(n)}\phi^{t-k-1}\right| \tag{28}$$

$$\leqslant (1-u)\sum_{k=0}^{t-1} u^k \phi^{t-k-1}\left|\sum_{n=1}^{K} \epsilon_0^{(n)}\right| \tag{29}$$

$$\leqslant (1-u)\sum_{k=0}^{t-1} \max(u,\phi)^{t-1}\left|\sum_{n=1}^{K} \epsilon_0^{(n)}\right| \tag{30}$$

$$\leqslant (1-u)t\max(u,\phi)^{t-1}\left|\sum_{n=1}^{K} \epsilon_0^{(n)}\right|, \tag{31}$$

where (27) follows from (24), (28) follows from by repetitively plugging the inequality expression for $|\epsilon_{t-1}^{(n)}|$ ($|\epsilon_{t-1}^{(n)}| \leqslant \phi^{(n)}|\epsilon_{t-2}^{(n)}|$) into the expression for $|\epsilon_t^{(n)}|$ ($|\epsilon_t^{(n)}| \leqslant \phi^{(n)}|\epsilon_{t-1}^{(n)}|$) for all t, (29) follows from the fact that $\phi^{(n)}$ is a positive constant, (30) follows from by bounding the term by choosing the maximum of u and ϕ, and (31) follows from the fact that u and ϕ are independent of time index k. If we take the limit of both sides in (31):

$$\lim_{t\to\infty} |\Delta_t^{it}| = 0, \tag{32}$$

which follows from the fact that $u, \phi \in (0,1)$.

E. Proof of Proposition 5

We first prove the expectation.

$$\lim_{t\to\infty} \mathcal{E}_t = \lim_{t\to\infty} \mathbf{Q}_t^{it} - \mathbf{Q}^*. \tag{33}$$

$$= \lim_{t\to\infty} (1-u)\sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbf{Q}_i^{(n)} - \mathbf{Q}^*. \tag{34}$$

$$= \lim_{t\to\infty} (1-u)\sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} (\mathbf{Q}_i^{(n)} - \mathbf{Q}^*). \tag{35}$$

$$= \lim_{t\to\infty} (1-u)\sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}, \tag{36}$$

where (34) follows from (15), (35) 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 (36) follows from (10). If we take the expectations of both sides:

$$\lim_{t\to\infty} \mathbb{E}[\mathcal{E}_t] = \lim_{t\to\infty} (1-u)\sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbb{E}[\mathcal{X}_i^{(n)}] = 0 \tag{37}$$

which follows from the linearity of expectation and (10).

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

$$= \lim_{t\to\infty} \mathbb{V}\left[(1-u)\sum_{i=0}^{t-1} u^{t-i-1} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathcal{X}_i^{(n)}\right]. \tag{38}$$

$$= \lim_{t\to\infty} (1-u)^2 \left[\sum_{i=0}^{t-1} u^{2(t-i-1)} \left[\sum_{n=1}^{K} \left(\mathbf{w}_i^{(n)}\right)^2 \mathbb{V}[\mathcal{X}_i^{(n)}] + 2\sum_{n=1}^{K}\sum_{m=n+1}^{K} \mathbf{w}_i^{(n)}\mathbf{w}_i^{(m)} \operatorname{Cov}(\mathcal{X}_i^{(n)}, \mathcal{X}_i^{(m)})\right]\right]. \tag{39}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \left[\sum_{i=0}^{t-1} u^{2(t-i-1)} \left[\sum_{n=1}^{K} \left(\mathbf{w}_i^{(n)}\right)^2 \mathbb{V}[\mathcal{X}_i^{(n)}] + 2\sum_{n=1}^{K}\sum_{m=n+1}^{K} \mathbf{w}_i^{(n)}\mathbf{w}_i^{(m)} \sqrt{\mathbb{V}[\mathcal{X}_i^{(n)}]\mathbb{V}[\mathcal{X}_i^{(m)}]}\right]\right]. \tag{40}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \left[\sum_{i=0}^{t-1} u^{2(t-i-1)} \left[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbb{V}[\mathcal{X}_i^{(n)}] + 2\sum_{n=1}^{K}\sum_{m=1}^{K} \mathbf{w}_i^{(n)}\mathbf{w}_i^{(m)} \sqrt{\mathbb{V}[\mathcal{X}_i^{(n)}]\mathbb{V}[\mathcal{X}_i^{(m)}]}\right]\right]. \tag{41}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \left[\sum_{i=0}^{t-1} u^{2(t-i-1)} \left[\sum_{n=1}^{K} \mathbf{w}_i^{(n)} \frac{\lambda^2}{3} + 2\sum_{n=1}^{K}\sum_{m=1}^{K} \mathbf{w}_i^{(n)}\mathbf{w}_i^{(m)} \frac{\lambda^2}{3}\right]\right]. \tag{42}$$

$$\leqslant \lim_{t\to\infty} (1-u)^2 \left[\sum_{i=0}^{t-1} u^{2(t-i-1)} \lambda^2\right]. \tag{43}$$

$$\leqslant \frac{(1-u)}{(1+u)}\lambda^2, \tag{44}$$

where (38) follows by taking the variance of both sides in (36), (39) follows from the properties of the variance operator and the modest independence assumption, (40) follows from the Cauchy-Schwarz inequality for the variance, (41) follows from the fact that $\mathbf{w}_t^{(n)} \leqslant 1$ and dropping the constraint in the second summation, (42) follows from (10) and $\lambda = \max_n \lambda_n$, (43) follows from the fact $\sum_{n=1}^{K} \mathbf{w}_t^{(n)} = 1$ for all t, and (44) follows from the infinite geometric sum formula and $u \in (0, 1)$.

If we assume *strict independence*, the cross terms in (39) to (42) disappear, and the upper bound will be smaller by a factor of 3. Refer to [39] for the proof of the upper bound on the variance for no independence case.

REFERENCES

[1] M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, "6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 957–975, 2020.

[2] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah, "Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 9–39, Jan. 2023.

[3] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017.

[4] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16219–16230, Nov. 2021.

[5] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707, Oct. 2019.

[6] Y. He, J. Guo, and X. Zheng, "From surveillance to digital twin: Challenges and recent advances of signal processing for industrial Internet of Things," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 120–129, Sep. 2018.

[7] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 364–375, 2022.

[8] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, 2020.

[9] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Belmont, MA, USA: Athena Scientific, 2019.

[10] M. Elsayed and M. Erol-Kantarci, "AI-enabled future wireless networks: Challenges, opportunities, and open issues," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 70–77, Sep. 2019.

[11] P. Ding, X. Liu, Z. Wang, K. Zhang, Z. Huang, and Y. Chen, "Distributed Q-learning-enabled multi-dimensional spectrum sharing security scheme for 6G wireless communication," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 44–50, Apr. 2022.

[12] S. Lee, H. Yu, and H. Lee, "Multiagent Q-learning-based multi-UAV wireless networks for maximizing energy efficiency: Deployment and power control strategy design," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6434–6442, May 2022.

[13] F. Wilhelmi, B. Bellalta, C. Cano, and A. Jonsson, "Implications of decentralized Q-learning resource allocation in wireless networks," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Piscataway, NJ, USA: IEEE Press, 2017, pp. 1–5.

[14] H. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst. 23*, 2010, pp. 2613–2621.

[15] O. Peer, C. Tessler, N. Merlis, and R. Meir, "Ensemble bootstrapping for Q-learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 8454–8463.

[16] O. Anschel, N. Baram, and N. Shimkin, "Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 176–185.

[17] Q. Lan, Y. Pan, A. Fyshe, and M. White, "Maxmin Q-learning: Controlling the estimation bias of Q-learning," 2020, *arXiv:2002.06487*.

[18] M. Ghavamzadeh, H. Kappen, M. Azar, and R. Munos, "Speedy Q-learning," in *Proc. Adv. Neural Inf. Process. Syst. 24*, 2011.

[19] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 881–888.

[20] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[21] F. S. Melo and M. Ribeiro, "Q-learning with linear function approximation," *Proc. Learn. Theory: 20th Annu. Conf. Learn. Theory (COLT)*, vol. 20, Jun. 13–15, 2007, pp. 308–322.

[22] M. Riedmiller, "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method," *Proc. Mach. Learn.: 16th Eur. Conf. Mach. Learn. (ECML)*, vol. 16, Oct. 3–7, 2005, pp. 317–328.

[23] I. Osband, C. Blundell, A. Pritzel, and B. Roy, "Deep exploration via bootstrapped DQN," in *Proc. Adv. Neural Inf. Process. Syst. 29*, 2016.

[24] Z. Hajiakhondi-Meybodi, A. Mohammadi, M. Hou, and K. N. Plataniotis, "DQLEL: Deep Q-learning for energy-optimized LoS/NLoS UWB node selection," *IEEE Trans. Signal Process.*, vol. 70, pp. 2532–2547, 2022.

[25] S. Evmorfos, K. I. Diamantaras, and A. P. Petropulu, "Reinforcement learning for motion policies in mobile relaying networks," *IEEE Trans. Signal Process.*, vol. 70, pp. 850–861, 2022.

[26] T. Bozkus and U. Mitra, "Ensemble link learning for large state space multiple access communications," in *Proc. 30th Eur. Signal Process. Conf. (EUSIPCO)*, 2022, pp. 747–751.

[27] T. Bozkus and U. Mitra, "Ensemble graph Q-learning for large scale networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[28] T. Bozkus and U. Mitra, "Multi-timescale ensemble Q-learning for Markov decision process policy optimization," 2024, *arXiv:2402.05476*.

[29] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2230–2254, 4th Quart. 2022.

[30] A. Masaracchia, V. Sharma, B. Canberk, O. A. Dobre, and T. Q. Duong, "Digital twin for 6G: Taxonomy, research challenges, and the road ahead," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 2137–2150, 2022.

[31] T. Bozkus and U. Mitra, "Link analysis for solving multiple-access MDPs with large state spaces," *IEEE Trans. Signal Process.*, vol. 71, pp. 947–962, 2023.

[32] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2016, pp. 1928–1937.

[33] F. S. Melo, "Convergence of q-learning: A simple proof," Inst. Syst. Robot., Lisbon, Portugal, 2001. [Online]. Available: http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf

[34] H. Wang, C. Ding, and H. Huang, "Directed graph learning via high-order co-linkage analysis," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, New York, NY, USA: Springer-Verlag, 2010, pp. 451–466.

[35] B. A. Craig and P. P. Sendi, "Estimation of the transition matrix of a discrete-time Markov chain," *Health Econ.*, vol. 11, no. 1, pp. 33–42, 2002.

[36] Y. Song, Y. Zhou, J. B. Ayush Sekhari, A. Krishnamurthy, and W. Sun, "Hybrid RL: Using both offline and online data can make RL efficient," 2023, *arXiv:2210.06718*.

[37] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based reinforcement learning: A survey," *Found. Trends® Mach. Learn.*, vol. 16, no. 1, pp. 1–118, 2023.

[38] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proc. Connectionist Models Summer School*, vol. 6, Hillsdale, NJ, USA: Lawrence Erlbaum, 1993, pp. 1–9.

[39] T. Bozkus and U. Mitra, "Supplementary appendix." GitHub, 2023. Accessed: Jan. 10, 2024. [Online]. Available: https://github.com/talhabozkus/Digital-Cousins-for-Ensemble-QLearning

[40] L. Liu and U. Mitra, "On sampled reinforcement learning in wireless networks: Exploitation of policy structures," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2823–2837, May 2020.

[41] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, "An improved sarsa (λ) reinforcement learning algorithm for wireless communication systems," *IEEE Access*, vol. 7, pp. 115418–115427, 2019.

[42] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," 2018, *arXiv:1802.10592*.

[43] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *Ann. Statist.*, vol. 35, no. 6, pp. 2769–2794, Dec. 2007.

[44] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 752–759.

[45] R. Parr, C. Painter-Wakefield, L. Li, and M. Littman, "Analyzing feature generation for value-function approximation," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 737–744.

Talha Bozkus (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from the Bilkent University, Turkey, in 2020. He is currently working toward the Ph.D. degree with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA. His research interests include signal processing, network optimization, and reinforcement learning. He is the recipient of a USC Annenberg Fellowship for graduate study and the Bilkent University Comprehensive Scholarship for his undergraduate studies.

Urbashi Mitra (Fellow, IEEE) received the B.S. and M.S. degrees from the University of California at Berkeley, Berkeley, CA, USA, and the Ph.D. degree from Princeton University, Princeton, NJ, USA. She is currently the Gordon S. Marshall Chair of engineering with the University of Southern California. Her research interests include wireless communications, biological communication, underwater acoustic communications, communication and sensor networks, and the detection, estimation, and the interface of communication, sensing, and control. She was a recipient of the 1996 National Science Foundation CAREER Award and the 1997 OSU College of Engineering MacQuigg Award for Teaching, the 2000 OSU College of Engineering Lumley Award for Research, the 2001 Okawa Foundation Award, the Texas Instruments Visiting Professorship (Rice University, Fall 2002), the 2009 DCOSS Applications and Systems Best Paper Award, the Best Paper Award from the 2012 GLOBECOM Signal Processing Symposium for Communications, the 2012 U.S. National Academy of Engineering Lillian Gilbreth Lectureship, the 2014 to 2015 IEEE Communications Society Distinguished Lecturer, the Women in Communications Engineering Technical Achievement Award from the IEEE Communications Society in 2017, the 2016 U.K. Royal Academy of Engineering Distinguished Visiting Professorship, the 2016 USA Fulbright Scholar Award, the 2016 to 2017 U.K. Leverhulme Trust Visiting Professorship, and the 2021 USC Viterbi School of Engineering Senior Research Award. She was the inaugural Editor-in-Chief of IEEE TRANSACTIONS ON MOLECULAR, BIOLOGICAL, AND MULTI-SCALE COMMUNICATIONS. She was an Associate Editor of IEEE TRANSACTIONS ON SIGNAL PROCESSING (2012–2015), IEEE TRANSACTIONS ON INFORMATION THEORY (2007–2011), IEEE JOURNAL OF OCEANIC ENGINEERING (2006–2011), and IEEE TRANSACTIONS ON COMMUNICATIONS (1996–2001).