Ensemble Link Learning for Large State Space Multiple Access Communications

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

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

***Abstract*—Wireless communication networks are well-modeled by Markov Decision Processes (MDPs), but induce a large state space which challenges policy optimization. Reinforcement learning such as Q-learning enables the solution of policy optimization problems in unknown environments. Herein a graph-learning algorithm is proposed to improve the accuracy and complexity performance of Q-learning algorithm for a multiple access communications problem. By exploiting the structural properties of the wireless network MDP, several structurally related Markov chains are created and these multiple chains are sampled to learn multiple policies which are fused. Furthermore, a state-action aggregation method is proposed to reduce the time and memory complexity of the algorithm. Numerical results show that the proposed algorithm achieves a reduction of 80% with respect to the policy error and a reduction of 70% for the runtime versus other state-of-the-art Q learning algorithms.**

***Index Terms*—Markov decision process (MDP), wireless networks, state aggregation, Q-learning, graph learning**

I. Introduction

Markov Decision Processes (MDPs) have been extensively used to model wireless communication networks [1], [2]. As wireless communication networks become more heterogeneous and complex, the design and implementation of optimal control strategies become more challenging. When the system dynamics and cost functions are not fully known, dynamic programming (DP) [3] cannot be employed and Q-learning, *model-free* reinforcement learning (RL) algorithm, can be used to learn the optimal policy. Recently, many variants of Q-learning have been introduced to overcome several performance and computational challenges such as overestimation and underestimation bias of Q-functions [4], [5]. In [6], neighbouring nodes share Q-functions with each other for performance improvement. Deep Q-networks use a neural network to approximate the Q-functions [7]. In [8], different multiple Q-functions are combined to improve the overall performance similar to ensemble learning.

We introduce an *ensembling* approach, in which the agent learns by collecting samples from multiple structurally related Markovian systems. To this end, we employ the *co-link* graph symmetrization which capture higher-order information between states via multiple Markov chains. We can exploit a block-circulant approximation on the original transition graph to characterize the structure of the co-link matrices. The unknown probability transition matrix is learned via a small set of collected trajectories and then used to create the co-link matrices which will drive the behavior of the virtual Markovian systems. As these matrices have different transition probabilities, they offer different trajectories for exploration. Furthermore, the special structure of the co-link matrices and cost functions further enable a state-action aggregation method which significantly lowers the memory complexity of the algorithm.

The main contributions of the paper are as follows: (i) We characterize the structure of the n^{th} order co-link matrix based on a block-circulant approximation. (ii) We propose a graph-learning algorithm based on an ensembling of the Q-functions from different, but structurally related Markov chains. (iii) We further provide a structural state-action aggregation idea to tackle the increasing memory complexity for large networks. On modest-sized networks, up to 99% memory reduction is achievable. (iv) Numerical results show that the use of samples obtained from structurally related Markov chains achieves 80% less average policy. In addition, the proposed method outperforms the other existing Q-learning algorithms, achieving a reduction of 60% in policy error and a reduction of 70% in runtime complexity.

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

II. System Model and Methods

A. Markov Decision Processes

MDPs are characterized by 4-tuples $\{\mathcal{S}, \mathcal{A}, p, c\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the finite state and action spaces, respectively. We denote s_t as the *state* and a_t as the *action* taken at discrete time period t. The transition from s to s' occurs with the probability $p_a(s,s') = p(s' = s_{t+1} \mid s = s_t, a = a_t)$, which is stored in the $(s,s',a)^{th}$ element of the transition probability tensor $\mathbf{P}$, and a bounded average cost $\mathbf{c}_a(s) = \sum_{s' \in \mathcal{S}} p_a(s,s')\, c_a(s,s')$ is incurred, which is stored in the s^{th} element of the cost vector $\mathbf{c}_a$, where $c_a(s,s')$ is the instantaneous transition cost from s to s' under action a. We focus on an infinite-horizon discounted cost MDP, where $t = \mathbb{Z}^+ \cup \{0\}$. Our goal is to solve the following optimization problem:

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

where for all $s \in \mathcal{S}$ and $a_t \in \mathcal{A}$, where $\mathbf{v}_\pi$ is the *value function* [3] under a given *policy* π, $\mathbf{v}^*$ is the *optimal value function*, 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 (*stochastic*) per state 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 space [3]. Hence, we concentrate on deterministic and stationary policies herein. Dynamic programming algorithms such as *value iteration* and *policy iteration* can be employed to solve (1) iteratively [3].

B. Q-Learning

Dynamic programming algorithms are *model-based*, that is, they require full knowledge of the system including the transition probability function and the cost function to solve the optimization problem. However, when the system is not fully observable or if the state-space (or action-space) is too large to iteratively solve (1), the *model-free* algorithms such as *Q-learning* can be utilized [3].

Q-learning algorithm seeks to solve the optimization problem and find the optimal policy π^* by learning the Q functions $Q(s,a), \forall (s,a) \in \mathcal{S} \times \mathcal{A}$ using the update rule:

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

where $\alpha \in (0,1)$ is the learning rate. In practice, ϵ-*greedy* policies are used to deal with the *exploration-exploitation* trade-off, *i.e.,* with probability ϵ, a random action is taken (exploration) and with probability $1-\epsilon$, a greedy action, which minimizes the Q-function of the next state is chosen (exploitation). This balance is crucial to ensure that sufficient information about the system is captured by visiting each state-action pair sufficiently many times. The agent interacts with the environment and collects samples $\{s, a, s', c\}$ to update Q-functions using (2). The learning strategy must specify the trajectory length (l), and the minimum number of visits to each state-action pair (v), which can be used as a stopping condition for the sampling operation. Q-functions converge to their optimal values with probability one, *i.e.,* $Q \xrightarrow{w.p.1} Q^*$ if necessary conditions are satisfied [9].

C. Co-link graph symmetrization method

As graph signal processing [10] (GSP) provides computational tools for graphical systems, they are a natural tool for MDPs described by their transition graphs. Much GSP work has focused on undirected (symmetric) graphs [10], [11]; however, the graph associated with our network is directed. Challenges with prior proposed GSP methods for directed graphs [12] are computational expense and stability issues. Thus, graph symmetrization methods are of interest. The goal is to preserve the underlying structure of the directed graph via a symmetric proxy. Progress has been made to capture the directionality between nodes, the degree of nodes and the weight of nodes [13], [14]; however, they only focus on the influence of *one-hop* neighbors and do not capture longer-range dependencies from *multiple-hops*. To this end, the *co-link* symmetrization method [15] preserves multi-hop edge dynamics during symmetrization.

To avoid losing information about distant node interaction in large graphs, bibliographic symmetrization [13], [16] has been considered, which exploits second-order node relationships. Herein, we consider the n^{th} *order similarity matrix*, denoted by $\mathbf{L}^{(n)}$ [15]. This matrix can be expressed in a compact summation form as:

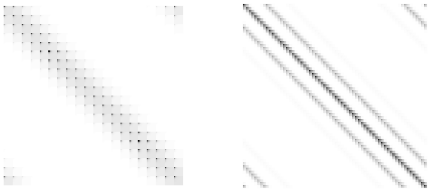
$$\mathbf{L}^{(n)} = \sum_{k=0}^{n-2} \mathbf{P}^{n-k-1} (\mathbf{P}^T)^{k+1} + (\mathbf{P}^T)^{n-k-1} \mathbf{P}^{k+1}, \quad (3)$$

where $\mathbf{P}$ is the transition matrix of a given MDP, and the derivation of $\mathbf{L}^{(n)}$ follows from the n^{th} order co-citation and co-reference processes [15].

We will show in the subsequent sections that with l_1 normalization, $\mathbf{L}^{(n)}$ matrices for each different n correspond to different but structurally related transition graphs, each of which can be used to improve the accuracy and computational complexity of the Q-learning algorithm. As each $\mathbf{L}^{(n)}$ capture different orders of node relationships, they can be employed to optimize policies for different states, which allows a simultaneous optimization of multiple policies. This procedure will also reduce the variance between the optimal and learned policies, a key goal of ensemble methods. As a result, the accuracy of learned policy is improved.

D. Wireless Network Model

We generalize the model of [16] by considering a wireless network system with R transmitters, a single receiver, and time-slotted communication. The action space of each transmitter is $\mathcal{A} = \{\text{Silent}, \text{Transmit}\}$. Each transmitter has a data buffer with size $N-1$ with $N > 1$. The data arrivals occur at the beginning of each slot and follow R i.i.d. Bernoulli distributions with the common arrival probability p. A packet drop occurs when there is a new arrival and the buffer is full. Let $b_t \in \{0, 1, \ldots, N-1\}$ denote the buffer state. Then, we define the buffer transition tensor $\mathbf{B}$ such that $\mathbf{B}_{i,j,k}$ stores the buffer transition probability from state i to j under the action k. Each transmitter experiences its own channel; however collisions can occur at the receiver due to simultaneous transmissions. We employ a generalized Gilbert-Elliot channel model, where the evolution of the channel is modeled as a Markov chain with M states. Let h_t denote the channel state following a discretized exponential distribution with a mean 1 (as in [16]). The channel state is h with probability $\int_h^{h+1} \mathrm{e}^{-x}\, \mathrm{d}x$ for $h \in \{0, 1, \ldots, M-2\}$ and $\int_h^{\infty} \mathrm{e}^{-x}\, \mathrm{d}x$ otherwise, where h denotes the channel index, $h \in \{0, 1, \ldots, M-1\}$. Similarly, we define the channel transition matrix $\mathbf{H}$ such that $\mathbf{H}_{i,j}$ stores the channel transition probability from state i to j. At any time t $\in \{0, 1, \ldots\}$, the state of the system is given by the tuple $s_t = (b_t, h_t)$. Thus, the transition tensor for the single transmitter system ($R = 1$) is given by: $\mathbf{P} = \mathbf{B} \otimes \mathbf{H}$, where $\otimes$ is the Kronecker product operator. For the multi-transmitter



(a) $R = 1, n = 4$ (b) $R = 2, n = 2$

Fig. 1: The gray-scaled adjacency matrices of $\mathbf{L}^{(n)}$

system ($R > 1$), the joint transition tensor is obtained by taking the Kronecker product of $\mathbf{P}$ by itself R times.

We want to minimize the expected discounted cost in (1) and find the optimal policy π^* where the single-stage cost function c in time slot t for a single transmitter is given as:

$$c = \eta \frac{b_t}{N-1} \mathbf{1}\{a_{t,i} = 0\} + (1-\eta) \frac{h_t}{M-1} \mathbf{1}\{a_{t,i} = 1\}, \quad (4)$$

where $\mathbf{1}(\cdot)$ is the indicator function, and the cost components are the backlog and transmission costs, respectively, with $\eta \in [0, 1]$. Let $\mathbf{V}_1$ be a cost matrix of size $|\mathcal{S}| \times 2$ storing the cost components for each state-action pair for a single transmitter. Then, for the multi-transmitter system ($R > 1$), the cost matrix $\mathbf{V}_R$ is obtained by taking the Kronecker product of $\mathbf{V}_1$ by itself R times. The following cost function captures the collision effects and is added to each component of $\mathbf{V}_R$:

$$c_{\text{collision}} = \zeta \mathbf{1}\{a_{t,i} = 1\} \mathbf{1}\{\sum_{j=1}^{R} \mathbf{1}\{a_{t,j} = 1\} > 1\}, \quad (5)$$

where $\zeta \in [0, 1]$ is the weight for the collision cost. Hence, the final cost function in time slot t is obtained by adding (5) to (4).

E. The representation of $\mathbf{L}^{(n)}$

We herein present an approximate characterization of $\mathbf{L}^{(n)}$ which will facilitate computation – the approximation is good when $p \approx 1$. For clarity, we first provide results for $R = 1$ and then discuss how to generalize to $R > 1$. Let $\mathbf{P}_0$ and $\mathbf{P}_1$ be two matrices corresponding to the "silent" and "transmit" actions in $\mathbf{P}$. We use the average transition matrix $\tilde{\mathbf{P}} = \frac{\mathbf{P}_0 + \mathbf{P}_1}{2}$ and employ a block-circulant approximation on $\tilde{\mathbf{P}}$ to derive $\mathbf{L}^{(n)}$ in (3). It can be shown that $\mathbf{L}^{(n)}$ is a symmetric block-circulant matrix of dimensions $NM \times NM$ and fully specified by $2n + 1$ non-zero blocks of dimensions $M \times M$ per row, where at most $n + 1$ of them are unique [17]. Let $\{\mathbf{L}_0, \mathbf{L}_1, \ldots, \mathbf{L}_n\}$ be the set of unique blocks, which satisfy $\mathbf{L}_k = \mathbf{L}_{N-k}$ for $k \in \{0, 1, \ldots, N-1\}$. The matrix $\mathbf{L}^{(n)}$ can be decomposed into two smaller matrices as follows:

$$\mathbf{L}^{(n)} = \hat{\mathbf{P}}^{(n)} \otimes \mathbf{H}^{(n)}, \quad (6)$$

where $\hat{\mathbf{P}}^{(n)}$ is a $N \times N$ symmetric circulant matrix, and $\mathbf{H}^{(n)}$ is the n^{th} order similarity matrix of $\mathbf{H}$ from (3) [17]. It is computationally trivial to compute $\mathbf{H}^{(n)}$ and approximate $\hat{\mathbf{P}}^{(n)}$ using Pascal's triangle with a low complexity [17]. For this reason, (6) is utilized for computation of $\mathbf{L}^{(n)}$ instead of (3).

For $R > 1$, it can also be shown that $\mathbf{L}^{(n)}$ is a block-circulant matrix with $(2n+1)^R$ non-zero blocks per row, where there are at most $\frac{(2n+1)^R+1}{2}$ unique blocks, which satisfy an analogous circulant property to that of $R = 1$ case. The adjacency matrix of $\mathbf{L}^{(n)}$ with different n and R are illustrated in Fig.1. Notice that there is a common block-circulant structure regardless of R and n.

F. Creating multiple Markovian systems

The underlying probability transition matrix, $\mathbf{P}$, is initially unknown. However, the approximate transition probabilities can be estimated via *sample averaging*. Let $\mathbf{P}_{ij} = p(s' = j \mid s = i)$, and v_{ij} be the number of times that the simulator moves from state i to j. Then, the maximum likelihood estimator of $\mathbf{P}$ is given as follows:

$$\hat{\mathbf{P}}_{ij} = \frac{v_{ij}}{\sum_{k=1}^{|\mathcal{S}|} v_{ik}}, \quad (7)$$

where $\hat{\mathbf{P}}_{ij} \to \mathbf{P}_{ij}$ for all i, j as $v_{ik} \to \infty$ for all k.

The simulator collects samples from the unknown environment, *i.e.* the Markov chain of $\mathbf{P}$, sufficiently many times, and constructs $\hat{\mathbf{P}}$. Then, a block-circulant approximation on $\hat{\mathbf{P}}$ is employed. Note that it is adequate to collect samples for only a subset of states, *representative states*, which are in the same block-row, and $\hat{\mathbf{P}}$ can be constructed using the block-row repetitions. This structural property considerably lowers the number of samples to estimate $\hat{\mathbf{P}}$.

The co-link matrices ($\mathbf{L}^{(n)}$) for different Markovian environments are then constructed employing the co-link technique with $\hat{\mathbf{P}}$. Then, each row of $\mathbf{L}^{(n)}$ is l_1 normalized, *i.e.*

$$\mathbf{L}_i^{(n)} \leftarrow \frac{\mathbf{L}_i^{(n)}}{\sum_j \mathbf{L}_{ij}^{(n)}}, \quad (8)$$

where $\mathbf{L}_i^{(n)}$ is the i^{th} row of $\mathbf{L}^{(n)}$. The normalization is needed to make meaningful performance comparisons between different $\mathbf{L}^{(n)}$ as the elements of $\mathbf{L}^{(n)}$ grow as a function of n from (3). We note that this transformation (i) does not change the block-circulant structure of $\mathbf{L}^{(n)}$, and (ii) creates stochastic transition matrices for $\mathbf{L}^{(n)}$, each of which corresponds to a different virtual Markov chain with the same states as in $\mathbf{P}$.

III. Link Learning Algorithm

In this section, we introduce the details of the proposed algorithm (Algorithm 1). It is a model-free algorithm, thus, the system dynamics including probability distributions and costs are unknown. We use K different Markov chains, each corresponding to l_1-normalized $\mathbf{L}^{(n)}$, $n = 1, 2, ..., K$, where $\mathbf{L}^{(1)} = \mathbf{P}$. Our aim is to learn the n^{th} order node (or link) relationships from different structurally related systems, and produce a single ensemble policy $\boldsymbol{\pi}^*$ with low complexity, and the complexity (time and memory) of finding such policy increases with most of the system parameters including $l, v, |\mathcal{S}|, |\mathcal{A}|, R, K$.

Let $u \in [0, 1)$ be the update ratio and $\mathbf{Q}^{(n)}$ be empty Q tables (of size $|\mathcal{S}| \times |\mathcal{A}|$) to store Q-functions $\forall n \in 1, ..., K$. We now briefly explain Algorithm 1. The stopping condition is satisfied when each state-action pair in the Markov chain corresponding to $\mathbf{L}^{(1)}$ is visited v times – there is no minimum visit requirement for $\mathbf{L}^{(n)}$ for $n > 1$. As n increases, the distribution of the coefficients in each row of $\mathbf{L}^{(n)}$ becomes more uniform, *i.e.* the coefficients get closer to each other

Algorithm 1 Link Learning

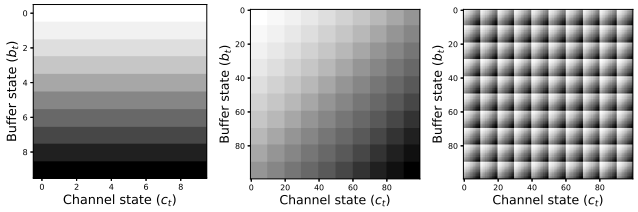
Input: l, v, u, $\mathbf{Q}^{(n)}$
Output: $\mathbf{Q^{it}}$, $\mathbf{Q^{nonit}}$, $\hat{\mathbf{w}}$, $\hat{\boldsymbol{\pi}}$

1: Initialize $\mathbf{w} \leftarrow \frac{1}{K}\mathbf{1}$
2: **while** each state-action pair in $\mathbf{L}^{(1)}$ not visited v times **do**
3: **for** each $n \in 1, ..., K$ **do**
4: collect samples of length l, update $\mathbf{Q}^{(n)}$ using (2)
5: $\hat{\boldsymbol{\pi}}(s) \leftarrow \text{argmin}_{a'} \sum_n \mathbf{w}_n \mathbf{Q}^{(n)}(s, a')$
6: $\boldsymbol{\pi}_n(s) \leftarrow \text{argmin}_{a'} \mathbf{Q}^{(n)}(s, a'),\ \forall n \in 1, ..., K$
7: $\mathbf{w}_n \leftarrow \frac{1}{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{S}|} \mathbf{1}(\hat{\boldsymbol{\pi}}(j) = \boldsymbol{\pi}_n(j))$
8: $\mathbf{w} \leftarrow softmax(\mathbf{w})$
9: $\mathbf{Q^{it}} \leftarrow u\mathbf{Q^{it}} + (1-u)\sum_n \mathbf{w}_n \mathbf{Q}^{(n)}$
10: $\hat{\mathbf{w}} \leftarrow \mathbf{w}$
11: $\mathbf{Q^{nonit}} \leftarrow \sum_n \hat{\mathbf{w}}_n \mathbf{Q}^{(n)}$
12: $\hat{\boldsymbol{\pi}}(s) \leftarrow \text{argmin}_{a'} \mathbf{Q^{it}}(s, a')$

[17]. Thus, by the time $\mathbf{L}^{(1)}$ is sufficiently explored, all $\mathbf{L}^{(n)}$ for $n > 1$ will already be visited. In 3-4, the simulator interacts with K different Markovian environments, collects set of samples $\{s, a, s', c\}$ from trajectories of length l independently, and updates the Q tables for all n. In 5-6, we set the *candidate optimal policy* ($\hat{\boldsymbol{\pi}}$) to the minimizer of the most recent weighted combination of Q-functions, and *candidate individual policies* ($\boldsymbol{\pi}_n$) to the minimizer of the most recent individual Q-functions. In 7, we compute the *correct estimation rate vector* ($\mathbf{w}$ with $\mathbf{w}_n$ being the n^{th} element of $\mathbf{w}$) *i.e.,* $\mathbf{w}_n$ indicates how useful the most recent samples from $\mathbf{L}^{(n)}$ is. We then update $\mathbf{Q^{it}}$ using the softmax-normalized $\mathbf{w}$ in 9. Note that $\mathbf{Q^{it}}$ (*iterative*) is obtained iteratively using possibly different $\mathbf{w}$ at each iteration, yet $\mathbf{Q^{nonit}}$ (*non-iterative*) is constructed using only the final weight vector $\hat{\mathbf{w}}$.

A. Structural state-action aggregation

In [4], [5], the methods suffer from high memory needs due to storing Q-functions and state-action pair visits; this issue is exacerbated here as we store this information for each $n \in \{1, ..., K\}$. However, utilizing the block-circulant representation of $\mathbf{L}^{(n)}$ and the nice structure of the cost functions, we can strongly reduce our memory needs. We briefly explain the idea for $R = 1$, and provide an example for $R > 1$. If we look at the cost function for $R = 1$, we see that when $a_t = 0$ and $a_t = 1$, the cost function is only proportional to b_t and h_t, respectively. Assuming the Q tables are initialized to 0, Q-function of any state-action pair will be directly proportional to the received cost. This implies that (i) all states having the same b_t can be aggregated in the same group for "silent" action, and (ii) all states having the same h_t can be aggregated in the same group for "transmit" action. The same idea can be applied to $R > 1$ case. (All states having the same two buffer states can be aggregated in the same group for "silent-silent" action.) The gray-scaled aggregated Q-matrix examples are shown in Fig.2. Although there is a collision effect for $R > 1$, it does not depend on b_t or h_t, and thus does not affect the aggregation. It can be shown that for $R = 2$, the overall memory reduction is $\frac{(N+M)^2}{(2NM)^2}$. For a network with N=M=10, up to 99% memory reduction is achievable. The reduction also increases exponentially with R, N, M.



(a) $R = 1$, action-0 (b) $R = 2$, action-00 (c) $R = 2$, action-11

Fig. 2: The gray-scaled aggregated Q-matrices

B. Convergence

The necessary conditions for the convergence of the Q-learning are applicable for our algorithm. We also provide the following three preliminary results for $n = 1$. The proofs are given in [17].

1- Let $\Delta^{it}_{(t-1)}(s, a) = \mathbf{Q^{it}}_{(t-1)}(s, a) - \mathbf{Q^{it}}_{(t)}(s, a)$. Then, $|\Delta^{it}_{(t)}(s, a) - \Delta^{it}_{(t-1)}(s, a)| \xrightarrow{t\to\infty} 0, \forall s, a$.

2- $|\Delta^{it}_{(t)}(s, a)| < \theta(s, a)$, where θ is the biggest update for a state-action pair (s, a) in the Q-learning, *i.e.,* $\theta(s, a) = \max_t\{\mathbf{Q}_{(t)}(s, a) - \mathbf{Q}_{(t-1)}(s, a)\}$.

3- If there exists a constant $\phi(s, a) \in (0, 1)$ such that $|\mathbf{Q}_{(t)}(s, a) - \mathbf{Q}_{(t-1)}(s, a)| \leqslant \phi|\mathbf{Q}_{(t-1)}(s, a) - \mathbf{Q}_{(t-2)}(s, a)|, \forall t$, then, $|\Delta^{it}_{(t)}(s, a)| \xrightarrow{t\to\infty} 0, \forall s, a$.

IV. Numerical Results

Let $\boldsymbol{\pi}^*$ be the optimal policy corresponding to $\mathbf{v}^*$ and $\hat{\boldsymbol{\pi}}$ be the estimated policy from Algorithm 1. We define the *average policy error* as $\Upsilon = \frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \mathbf{1}\left(\boldsymbol{\pi}^*(s) \neq \hat{\boldsymbol{\pi}}(s)\right)$. The simulations are carried out with the following parameters: N = 10, M = 6, R = 2, $|\mathcal{S}|$ = 3600, p = 0.9, l = 5, v = 5, η = 0.5, ζ = 0.1, u = 0.5, ϵ = 0.25, γ = 0.9, α is initialized to 0.5 and decreases $\approx \frac{1}{t}$, where t is iteration index. In Fig.3, we compare the average policy error achieved using a single $\mathbf{L}^{(n)}(n = 1, \cdots 4)$ and our weighted approach ($\mathbf{Q^{it}}$). $\mathbf{Q^{it}}$ achieves nearly zero policy error with significantly lower runtime because: (i) multiple cumulative costs are minimized, and (ii) $\mathbf{Q^{it}}$ employs information from all of the transitions for $\mathbf{L}^{(n)}(n = 1, \cdots 4)$. Thus, we combine the effect of exploration for different states using different Markov chains enabling the initial sharp decrease for $\mathbf{Q^{it}}$.

In Fig.4, the average policy error performance results for different algorithms (see Table I) across different network sizes are given (with the same parameters), where the network size is obtained as follows: the interval [0, 2000] is formed by changing N, M with $R = 1$, [2000, 4000] is formed similarly with $R = 2$ and so on. Since the proposed algorithm uses different Q-functions to produce the optimal policy, only the value-based model-free algorithms, which use the same strategy, are employed for fair performance comparison. See [17] for the hyper-parameters for different methods. The use of $\mathbf{Q^{it}}$ results in significantly lower performance degradation across increasing network size. On the other hand, $\mathbf{Q^{non-it}}$ only considers the final characteristics of different Markov chains, and DQN does not utilize the structural properties of

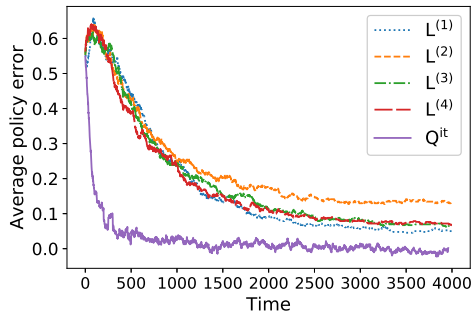


Fig. 3: Average policy error for different systems

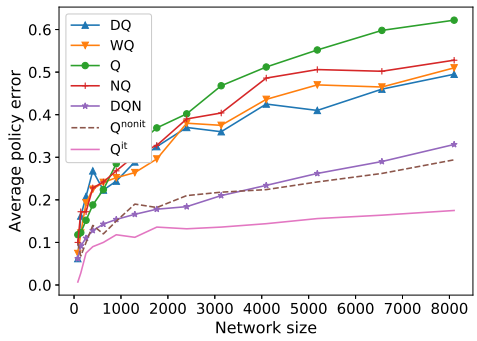


Fig. 4: Average policy error for different algorithms

Simple Q (Q)	(2)
Double Q (DQ)	[4]
Weighted Double Q (WQ)	[5]
Distributed Q (NQ)	[6]
Deep Q-Network (DQN)	[7]

TABLE I: Q-learning algorithm and variants – notation and citations.

r	reduction	Υ
1.0	$\approx$ %98	0.27
0.5	$\approx$ %95	0.22
0.25	$\approx$ %85	0.19
0.1	$\approx$ %81	0.14
0	%0	0.10

TABLE II: Memory reduction vs average policy error performance

the Markov chains; thus, they have inferior performances. The performances of DQ and WQ also suggest that the use of double estimators is not the optimal strategy.

We consider the effect of state-action aggregation. In Section III-A, we provided the scheme for maximal aggregation; herein we consider partial aggregation, where r is the ratio of employed aggregation over the maximal. As shown in Table II, there is a trade-off between the amount of memory reduction and the average policy error. Thus, whether a higher memory reduction or lower policy error is prioritized directly affects the selection of r. The effect of aggregation on the policy error performance decreases for large networks as a result of the performance of Algorithm 1 across large networks.

The time-complexity of the proposed link-learning algorithm can be shown to be $\approx O\left(\frac{(|\mathcal{S}||\mathcal{A}|)^{Rr}v}{K}f(l,\epsilon)\right)$, where f is a (possibly) non-monotonic function of l, ϵ. For the Q-learning algorithm, $r = K = 1$; thus, there is a clear improvement. As network size increases, the improvement increases due to the running of multiple Markov chains simultaneously, reducing l and v. The run-time is inversely proportional to the number of simultaneous Markov chains since sufficient sampling for $\mathbf{L}^{(\mathbf{n})}$ decreases as n increases due to the increasing monotonicity in the distribution of $\mathbf{L}^{(\mathbf{n})}$. Thus, sampling for $\mathbf{L}^{(\mathbf{n})}, n > 1$, can terminate early further reducing runtime complexity. The simulation results show that the proposed algorithm achieves 70% less runtime than the Q-learning methods given in Table I for large networks (see [17] for further experimental results).

V. Conclusions

In this paper, we propose a graph-learning algorithm to increase the accuracy and computational performance of the Q-learning algorithm, using the structural properties of the $\mathbf{L}^{(n)}$ matrices and cost matrices. The proposed algorithm utilizes different virtual Markov chains in an adapted Q-learning algorithm; thus, combining the different exploration capabilities of different chains, which effectively increases the total amount of exploration with relatively smaller complexity, without requiring long sample trajectories and a huge number of visit requirements. Furthermore, depending on the initialization of individual Q-learning algorithms, Q-functions of the proposed algorithm can be shown to fully converge. We also show that the proposed algorithm performs much better than other state-of-the-art Q-learning algorithms in terms of accuracy and complexity.

References

[1] S. Mao, M. H. Cheung, and V. W. Wong, "An optimal energy allocation algorithm for energy harvesting wireless sensor networks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012.

[2] M. M. Vasconcelos, M. Gagrani, A. Nayyar, and U. Mitra, "Optimal scheduling strategy for networked estimation with energy harvesting," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1723–1735, 2020.

[3] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[4] H. Hasselt, "Double Q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[5] Z. Zhang, Z. Pan, and M. J. Kochenderfer, "Weighted double Q-learning." in *IJCAI*, 2017, pp. 3455–3461.

[6] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," 1999.

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

[8] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 104–114.

[9] F. S. Melo, "Convergence of Q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.

[10] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[11] A. G. Marques, S. Segarra, and G. Mateos, "Signal processing on directed graphs: The role of edge directionality when processing and learning from network data," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 99–116, 2020.

[12] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," 2013.

[13] V. Satuluri and S. Parthasarathy, "Symmetrizations for clustering directed graphs," in *Proceedings of the 14th International Conference on Extending Database Technology*, 2011, pp. 343–354.

[14] B. He, H. Liu, X. H. Zhao, and Z. F. Li, "Weight-discounted symmetrization in clustering directed graphs," in *Advanced Materials Research*, vol. 756. Trans Tech Publ, 2013, pp. 2979–2987.

[15] H. Wang, C. Ding, and H. Huang, "Directed graph learning via high-order co-linkage analysis," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 451–466.

[16] L. Liu, A. Chattopadhyay, and U. Mitra, "On solving mdps with large state space: Exploitation of policy structures and spectral properties," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4151–4165, 2019.

[17] T. Bozkus and U. Mitra, "Supplementary proofs," https://github.com/talhabozkus/eusipco-proofs, 2022.