Appendix

I. Proofs

A. The structure and decomposition of $\mathbf{L}^{(n)}$

1) **The block circulant approximation***:* We firstly show the block-circulant approximations on $\mathbf{P}_0$ and $\mathbf{P}_1$. Using the identity $\mathbf{P} = \mathbf{B} \otimes \mathbf{C}$ and the transition probabilities between buffer states, $\mathbf{P}_0$ and $\mathbf{P}_1$ can be expressed as follows:

$$\mathbf{P}_0 = (1-p)\mathbf{S}^0 \otimes \mathbf{C} + p\mathbf{S}^1 \otimes \mathbf{C} + \mathbf{N}_0, \tag{1}$$

$$\mathbf{P}_1 = (1-p)\mathbf{S}^{-1} \otimes \mathbf{C} + p\mathbf{S}^0 \otimes \mathbf{C} + \mathbf{N}_1, \tag{2}$$

where $\mathbf{N}_0, \mathbf{N}_1$ are noise matrices with $NM \times NM$ dimensions, and $\mathbf{S}$ is a $N \times N$ circular shift matrix defined as:

$$\mathbf{S}_{i,j} = \mathbf{1}\{(i+1) \bmod N = j\}, \tag{3}$$

which satisfies the equality $\mathbf{S}^T = \mathbf{S}^{-1}$. We remove $\mathbf{N}_0$ and $\mathbf{N}_1$ so that $\mathbf{P}_0$ and $\mathbf{P}_1$ are block-circulant.

2) **The relationship between $\mathbf{C}^n$ and $\mathbf{C}$***:* We now derive the relationship between $\mathbf{C}^2$ and $\mathbf{C}$.

$$\begin{aligned}
\mathbf{C}^2_{i,j} &= \sum_{k=0}^{M-1} \mathbf{C}_{i,k}\mathbf{C}_{k,j} \\
&= \sum_{k=0}^{M-1} c_k c_j \\
&= c_j \\
&= \mathbf{C}_{i,j},
\end{aligned} \tag{4}$$

where the 2^{nd} and 4^{th} equalities follow from the distribution of channel state, and the 3^{rd} equality follows from the stochastic property (rows sum up to 1) of $\mathbf{C}$. By inspection, $\mathbf{C}^k = \mathbf{C}$ as well as $(\mathbf{C}^T)^k = \mathbf{C}^T$ for k $\geq$ 1. Now, we derive the relationship between $\mathbf{C}^{(n)}$ and $\mathbf{C}^{(2)}$:

$$\begin{aligned}
\mathbf{C}^{(n)} &= \sum_{k=0}^{n-2} \mathbf{C}^{n-k-1}(\mathbf{C}^T)^{k+1} + (\mathbf{C}^T)^{n-k-1}\mathbf{C}^{k+1} \\
&= \sum_{k=0}^{n-2} \mathbf{C}\mathbf{C}^T + \mathbf{C}^T\mathbf{C} \\
&= (n-1)(\mathbf{C}\mathbf{C}^T + \mathbf{C}^T\mathbf{C}) \\
&= (n-1)\mathbf{C}^{(2)},
\end{aligned} \tag{5}$$

where the 1^{st} equality follows from the n^{th} order similarity matrix of $\mathbf{C}$, 2^{nd} equality follows from the previous results: $\mathbf{C}^k = \mathbf{C}$ and $(\mathbf{C}^T)^k = \mathbf{C}^T$, and the last equality follows from the 2^{nd} order similarity matrix of $\mathbf{C}$. Notice that it is trivial to compute $\mathbf{C}^{(n)}$ as the complexity of computing $\mathbf{C}^{(n)}$ is linear in n.

3) **The block-circulant structure of $\mathbf{C}^{(n)}$**: We herein derive the structure of $\mathbf{L}^{(n)}$ for arbitrary n:

$$\begin{aligned}
\mathbf{L}^{(n)} &= \sum_{k=0}^{n-2} \tilde{\mathbf{P}}^{n-k-1}(\tilde{\mathbf{P}}^T)^{k+1} + (\tilde{\mathbf{P}}^T)^{n-k-1}\tilde{\mathbf{P}}^{k+1} \\
&= \sum_{k=0}^{n-2} (\tilde{\mathbf{S}} \otimes \mathbf{C})^{n-k-1}((\tilde{\mathbf{S}} \otimes \mathbf{C})^T)^{k+1} + ((\tilde{\mathbf{S}} \otimes \mathbf{C})^T)^{n-k-1}(\tilde{\mathbf{S}} \otimes \mathbf{C})^{k+1} \\
&= \sum_{k=0}^{n-2} (\tilde{\mathbf{S}}^{n-k-1} \otimes \mathbf{C}^{n-k-1})((\tilde{\mathbf{S}}^T)^{k+1} \otimes (\mathbf{C}^T)^{k+1}) + ((\tilde{\mathbf{S}}^T)^{n-k-1} \otimes (\mathbf{C}^T)^{n-k-1})(\tilde{\mathbf{S}}^{k+1} \otimes \mathbf{C}^{k+1})
\end{aligned} \tag{6}$$

$$
\begin{aligned}
&= \sum_{k=0}^{n-2} (\tilde{\mathbf{S}}^{n-k-1}(\tilde{\mathbf{S}}^T)^{k+1}) \otimes (\mathbf{C}^{n-k-1}(\mathbf{C}^T)^{k+1}) + ((\tilde{\mathbf{S}}^T)^{n-k-1}\tilde{\mathbf{S}}^{k+1}) \otimes ((\mathbf{C}^T)^{n-k-1}\mathbf{C}^{k+1}) \\
&= \left(\sum_{k=0}^{n-2} \tilde{\mathbf{S}}^{n-k-1}(\tilde{\mathbf{S}}^T)^{k+1}\right) \otimes \mathbf{C}\mathbf{C}^T + \left(\sum_{j=0}^{n-2} \tilde{\mathbf{S}}^{j+1}(\tilde{\mathbf{S}}^T)^{n-j-1}\right) \otimes \mathbf{C}^T\mathbf{C} \\
&= \left(\sum_{k=0}^{n-2} \tilde{\mathbf{S}}^{n-k-1}(\tilde{\mathbf{S}}^T)^{k+1}\right) \otimes \mathbf{C}\mathbf{C}^T + \left(\sum_{j=0}^{n-2} \tilde{\mathbf{S}}^{n-j-1}(\tilde{\mathbf{S}}^T)^{j+1}\right) \otimes \mathbf{C}^T\mathbf{C} \\
&= \left(\sum_{k=0}^{n-2} \tilde{\mathbf{S}}^{n-k-1}(\tilde{\mathbf{S}}^T)^{k+1}\right) \otimes (\mathbf{C}\mathbf{C}^T + \mathbf{C}^T\mathbf{C}) \\
&= \frac{1}{n-1}\left(\sum_{k=0}^{n-2} \tilde{\mathbf{S}}^{n-k-1}(\tilde{\mathbf{S}}^T)^{k+1}\right) \otimes (n-1)\mathbf{C}^{(2)} \\
&= \hat{\mathbf{P}}^{(n)} \otimes \mathbf{C}^{(n)}, \qquad (7)
\end{aligned}
$$

where the 2^{nd} equality follows from the partition of the average transition matrix as $\tilde{\mathbf{P}} = \tilde{\mathbf{S}} \otimes \mathbf{C}$ (where $\tilde{\mathbf{S}}$ is a $N \times N$ circulant matrix and can be obtained using (1) and (2)), the 3^{rd} and the 4^{th} equalities follow from the properties of the Kronecker product. In the 5^{th} line, we utilize the previous results: $\mathbf{C}^k = \mathbf{C}$ and $(\mathbf{C}^T)^k = \mathbf{C}^T$, change the second summation variable k to j and use the fact that circulant matrices commute. In the 6^{th} line, we make $j \to -j + n - 2$ substitution and change the order of summation. In the 7^{th} line, we merge two summations. The 8^{th} and last equalities follow from the similarity matrix definitions of $\mathbf{C}^{(2)}$ and $\mathbf{C}^{(n)}$.

Let's consider $\hat{\mathbf{P}}^{(n)}$. If we replace $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{S}}^T$ by their expressions, we get a polynomial in $\mathbf{S}$:

$$
= \frac{1}{n-1}[\sum_{k=0}^{n-2}((1-p)\mathbf{S}^{-1} + \mathbf{S}^0 + p\mathbf{S}^1)^{n-k-1}(((1-p)\mathbf{S}^{-1} + \mathbf{S}^0 + p\mathbf{S}^1)^T)^{k+1}]. \qquad (8)
$$

Here, the terms with the smallest degree and the largest degree in $\mathbf{S}$ are $\mathbf{S}^{-n}$ and $\mathbf{S}^n$ with the following coefficients, respectively:

$$
\frac{1}{n-1}[\sum_{k=0}^{n-2}(1-p)^{n-k-1}p^{k+1}] \quad \text{and} \quad \frac{1}{n-1}[\sum_{k=0}^{n-2}(1-p)^{k+1}p^{n-k-1}]. \qquad (9)
$$

Notice that these summations are the same by inspection. Carrying out this operation for all degrees show that the coefficients of $\mathbf{S}^{-k}$ and $\mathbf{S}^k$ are the same for all $k \in \{-n, -n+1, \ldots, n\}$. There are $2n+1$ terms in total yet $n+1$ of them are unique at most, which implies that $\hat{\mathbf{P}}^{(n)}$ is circulant with $n+1$ unique coefficients at most. Consequently, $\mathbf{L}^{(n)}$ is block-circulant with at most $n+1$ unique blocks, where each block is obtained by the Kronecker product of $\hat{\mathbf{P}}^{(n)}$ and $\mathbf{C}^{(n)}$, which gives a decomposition for $\mathbf{L}^{(n)}$ in terms of smaller matrices.

B. The approximation of $\hat{\mathbf{P}}^{(n)}$

It is costly to obtain $\hat{\mathbf{P}}^{(n)}$ by evaluating its compact sum expression. Thus, we take advantage of the structural similarity between $\hat{\mathbf{P}}^{(n)}$ and Pascal triangle to find an approximate form. We want to show that $\hat{\mathbf{P}}^{(n)}$ is circulant with $2n+1$ coefficients per row and at most of $n+1$ of them are unique. Let α_k denote the unique coefficients for $k \in \{0, 1, ..., n\}$. Following the decomposition of $\mathbf{L}^{(n)}$, it is easy to see that the structure of the first row of $\hat{\mathbf{P}}^{(n)}$ is the same as that of $\mathbf{L}^{(n)}$ except the blocks, $\mathbf{L}_k$, are replaced by coefficients α_k. If p is close to 1, we can drop $1-p$ term and express $\tilde{\mathbf{S}} \approx (\mathbf{S}^0 + \mathbf{S}^1)$. We also ignore the $\frac{1}{n-1}$ coefficient for now.

$$
\begin{aligned}
\hat{\mathbf{P}}^{(n)} &\approx \sum_{k=0}^{n-2}(\mathbf{S}^0 + \mathbf{S}^1)^{n-k-1}((\mathbf{S}^0 + \mathbf{S}^1)^T)^{k+1} \\
&\approx \sum_{k=0}^{n-2}(\mathbf{S}^0 + \mathbf{S}^1)^{n-k-1}(\mathbf{S}^{-1} + \mathbf{S}^0)^{k+1}
\end{aligned} \qquad (10)
$$

$$\approx \sum_{k=0}^{n-2} (\mathbf{S}^0 + \mathbf{S}^1)^{n-k-1} \mathbf{S}^{-(k+1)} (\mathbf{S}^0 + \mathbf{S}^1)^{k+1}$$
$$\approx \sum_{k=0}^{n-2} (\mathbf{S}^0 + \mathbf{S}^1)^{n} \mathbf{S}^{-(k+1)}$$
$$\approx (\mathbf{S}^0 + \mathbf{S}^1)^{n} \sum_{k=0}^{n-2} \mathbf{S}^{-(k+1)}, \tag{11}$$

where the 2^{nd} line follows from (3), and the 4^{th} line follows from the commutative property of circular matrices.

The coefficients in the expansion of $(\mathbf{S}^0 + \mathbf{S}^1)^n$ are the numbers on the $(n+1)^{th}$ line of the Pascal triangle. Multiplying the expansion with $\mathbf{S}^{-(k+1)}$ decreases the degrees in the expansion by $k+1$ yet keeps the coefficients the same, which corresponds to shifting the $(n+1)^{th}$ line of Pascal triangle by $k+1$. If we keep shifting the $(n+1)^{th}$ line by $k+1$ for $k \in \{0, 1, \dots n-2\}$ and add the resulting coefficients one under the other, we obtain all the coefficients. We also set $\alpha_n = 0$. The idea is illustrated for $n = 4$. There are $2n+1 = 9$ coefficients, whereas $n+1 = 5$ of them are unique. At the end, we set α_4 to 0.

				1	4	6	4	1	
			1	4	6	4	1		
+		1	4	6	4	1			
	0	**1**	**5**	**11**	**14**	**11**	**5**	**1**	**0**
	α_4	α_3	α_2	α_1	α_0	α_1	α_2	α_3	α_4

The unique coefficients are $\{14, 11, 5, 1, 0\}$. They are normalized so that they sum up to 1. (There is no problem for ignoring $\frac{1}{n-1}$ at the beginning due to normalization.) One important observation is that as n increases, the normalized unique coefficients becomes closer to each other. Thus, the distribution of $\mathbf{L}^{(n)}$ becomes more uniform. This follows from the fact that the length of the n^{th} line of the Pascal triangle becomes smaller compared to the number of non-zero coefficients, which is $2n+1$ as n increases.

C. Convergence results

1) **Proof of 3.2.1***:* Let n = 1. Assume $\mathbf{Q}$ and $\mathbf{Q^{it}}$ are the matrices storing the corresponding Q-functions. The general iteration rule is as follows (from Algorithm 1, line 10):

$$\mathbf{Q^{it}}(s,a) = u\mathbf{Q^{it}}(s,a) + (1-u)\mathbf{Q}(s,a). \tag{12}$$

Let $Q_{(t)}$ denote the Q-function of a specific state-action pair (s,a) at time t. For simplicity, we drop the notation for pair (s,a) from now on.

Then, $Q_{(t)}$ for t = {1,2,3,4} can be obtained using the iteration rule and the fact that $Q^{it}_{(0)} = 0$ as follows:

$$Q^{it}_{(1)} = uQ^{it}_{(0)} + (1-u)Q_{(0)}. \tag{13}$$
$$Q^{it}_{(2)} = uQ^{it}_{(1)} + (1-u)Q_{(1)} = (1-u)[uQ_{(0)} + Q_{(1)}]. \tag{14}$$
$$Q^{it}_{(3)} = uQ^{it}_{(2)} + (1-u)Q_{(2)} = (1-u)[u^2Q_{(0)} + uQ_{(1)} + Q_{(2)}]. \tag{15}$$
$$Q^{it}_{(4)} = uQ^{it}_{(3)} + (1-u)Q_{(3)} = (1-u)[u^3Q_{(0)} + u^2Q_{(1)} + uQ_{(2)} + Q_{(3)}]. \tag{16}$$

Note that $Q_{(0)}$ is also 0. However, for simplicity of the calculations, we keep it. We define the following update metrics for the Q-learning algorithm and the proposed algorithm:

$$\Delta^{it}_{(t-1)} = Q^{it}_{(t-1)} - Q^{it}_{(t)}. \tag{17}$$
$$\Delta_{(t-1)} = Q_{(t-1)} - Q_{(t)}. \tag{18}$$

Then, by simple algebraic manipulations, we can show that:

$$u\Delta^{it}_{(t-1)} - \Delta^{it}_{(t)} = (1-u)(Q_t - Q_{(t-1)}). \tag{19}$$
$$\Delta^{it}_{(t)} = u\Delta^{it}_{(t-1)} + (1-u)\Delta_{(t-1)}. \tag{20}$$
$$< \Delta^{it}_{(t-1)} + (1-u)\Delta_{(t-1)}, \tag{21}$$

where (21) follows from the fact that $u \in [0, 1)$. Then, the difference between consecutive updates can be bounded as follows:

$$\Delta^{it}_{(t)} - \Delta^{it}_{(t-1)} < (1-u)\Delta_{(t-1)}. \tag{22}$$

$$|\Delta^{it}_{(t)} - \Delta^{it}_{(t-1)}| < (1-u)|\Delta_{(t-1)}|. \tag{23}$$

By the convergence of the Q-learning algorithm, $|\Delta_{(t-1)}| \xrightarrow{t\to\infty} 0$. Thus, $|\Delta^{it}_{(t)} - \Delta^{it}_{(t-1)}| \xrightarrow{t\to\infty} 0$. This result implies that the magnitude of the difference between consecutive updates for Q^{it} gets stabilized.

2) **Proof of 3.2.2***:* Assume $n = 1$. Using (18), (19) and simple manipulations, the following equalities can be derived:

$$u\Delta^{it}_{(0)} - \Delta^{it}_{(1)} = -(1-u)\Delta_{(0)}. \tag{24}$$

$$u^2\Delta^{it}_{(0)} - \Delta^{it}_{(2)} = -(1-u)(u\Delta_{(0)} + \Delta_{(1)}). \tag{25}$$

$$u^3\Delta^{it}_{(0)} - \Delta^{it}_{(3)} = -(1-u)(u^2\Delta_{(0)} + u\Delta_{(1)} + \Delta_{(2)}). \tag{26}$$

Using the fact that $\Delta^{it}_{(0)} = 0$, the following equalities can also be shown:

$$|\Delta^{it}_{(1)}| = (1-u)|\Delta_{(0)}|. \tag{27}$$

$$|\Delta^{it}_{(2)}| = (1-u)|(u\Delta_{(0)} + \Delta_{(1)})|. \tag{28}$$

$$|\Delta^{it}_{(3)}| = (1-u)|(u^2\Delta_{(0)} + u\Delta_{(1)} + \Delta_{(2)})|, \tag{29}$$

which can be generalized for any t as follows:

$$|\Delta^{it}_{(t)}| = (1-u)|\sum_{k=0}^{t-1} u^k \Delta_{(t-k-1)}|. \tag{30}$$

If there is a constant θ such that each of the Q-learning update is bounded above by θ, *i.e.* $|\Delta_{(t)}| \leq \theta$ (in other words, the biggest update during the Q-learning algorithm is θ), then:

$$|\Delta^{it}_{(t)}| < (1-u)\sum_{k=0}^{t-1} u^k |\Delta_{(t-k-1)}|, \tag{31}$$

$$< \theta(1-u)\sum_{k=0}^{t-1} u^k, \tag{32}$$

$$= \theta(1-u^t), \tag{33}$$

where (31) follows from the triangle inequality, (32) follows from the $|\Delta_{(t)}| \leq \theta$ bound, and (33) follows from the fact that $u \in [0, 1)$. These results imply that $|\Delta^{it}_{(t)}| < \theta$ as $t \to \infty$.

3) **Proof of 3.2.3***:* Note that the convergence of the proposed algorithm heavily depends on the converge of the Q-learning algorithm. Thus, depending on the one-step updates of the Q-learning, several tight bounds can be derived. If there exists a constant $\phi \in (0, 1)$ such that each of the Q-learning update satisfies the following condition: $|\Delta_{(t)}| \leq \phi|\Delta_{(t-1)}|, \forall t$, then,

$$|\Delta^{it}_{(t)}| < (1-u)\sum_{k=0}^{t-1} u^k |\Delta_{(t-k-1)}|, \tag{34}$$

$$< (1-u)|\Delta_{(0)}|\sum_{k=0}^{t-1} u^k \phi^{(t-k-1)}, \tag{35}$$

where (35) follows from the fact that $|\Delta_{(t)}| \leq \phi^t|\Delta_{(0)}|$, which can be obtained from the update condition on the Q-learning algorithm. Let $u = \phi$. Then,

$$|\Delta^{it}_{(t)}| < (1-\phi)|\Delta_{(0)}|\sum_{k=0}^{t-1} \phi^{(t-1)}, \tag{36}$$

$$< (1-\phi)|\Delta_{(0)}|t\phi^{(t-1)}, \tag{37}$$

where 36 follows from $u = \phi$. As $t \to \infty$, $t\phi^{(t-1)} \to 0$. Thus, $|\Delta^{it}_{(t)}| \to 0$.

D. **Time complexity reduction**

In this section, we provide an estimate of the time-complexity of the proposed algorithm, and compare it with that of Q-learning algorithm.

The number of different state-action pairs visited during a trajectory of length l changes every-time. The minimum possible different visits is 1, and let the maximum be l_1, where $l_1 \leq l$ and l_1 is a function of l and ϵ. Thus, on average, $\frac{(l_1+1)}{2}$ different state-action pairs are visited.

Then, the minimum number of iterations for a sufficient exploration of the whole system is $\frac{(|\mathcal{S}||\mathcal{A}|)^R v}{\frac{(l_1+1)}{2}}$ because at state-action pair must be visited at least v times. During each trajectory of length l, there are exactly l state-action pairs are visited. Assume each visit and corresponding Q-function update takes a unit time. The time for a sufficient exploration then can be given by:

$$t_1 = \frac{(|\mathcal{S}||\mathcal{A}|)^R v}{\frac{(l_1+1)}{2}} l, \tag{38}$$

which corresponds to the following time-complexity:

$$\approx O\left(\frac{(|\mathcal{S}||\mathcal{A}|)^R v}{l_1} l\right) \approx O\left((|\mathcal{S}||\mathcal{A}|)^R v f(l,\epsilon)\right), \tag{39}$$

where $f(l,\epsilon)$ is some function of the trajectory length l and ϵ (exploration-exploitation constant). In the proposed algorithm, there are K different cases. During a trajectory of length l, the minimum possible different visits is 1, and let the maximum be l_n for n^{th} system, where $l_n \leq l$ and l_n is a function of l and ϵ for all $n \in \{1, 2, ...K\}$. Thus, on average, $\frac{(l_n+1)}{2}$ different state-action pairs are visited in the n^{th} system. Because of the increasing monotonicity in the transition matrix of the n^{th} system, $l_1 \ll l_2 \ll ... \ll l_n$.

As explained in the paper, the minimum number of visit v requirement is significantly alleviated due to the different exploration capability of different Markov chains. Assume the minimum number of visit requirement becomes $v' \ll v$. We can approximately say $v' \approx \frac{v}{K}$.

Then, the total time t for sufficient explorations for all n systems is given by:

$$t = t_1 + t_2 + ... + t_n = \sum_n \frac{(|\mathcal{S}||\mathcal{A}|)^R v'}{\frac{(l_n+1)}{2}} l \tag{40}$$

Assume the whole size of state-action space can be reduced by $\tau \in (0, 1]$ through the state-action aggregation idea. Then, the equation for t can be updated as follows:

$$t = t_1 + t_2 + ... + t_n = \sum_n \frac{(|\mathcal{S}||\mathcal{A}|)^{R\tau} v'}{\frac{(l_n+1)}{2}} l \tag{41}$$

Using the fact that $v' \approx \frac{v}{K} \ll v$, and $l_1 \ll l_2 \ll ... \ll l_n$, we have the following time-complexity:

$$\approx O\left(\frac{(|\mathcal{S}||\mathcal{A}|)^{R\tau} v}{l_1 K} l\right) \approx O\left(\frac{(|\mathcal{S}||\mathcal{A}|)^{R\tau} v}{K} f(l,\epsilon)\right) \tag{42}$$

If we compare the (39) and (42), we see that:

- The dependence of the time-complexity on $|\mathcal{S}|$, $|\mathcal{A}|$, R is considerably reduced. Thus, as the system parameters gets larger, the proposed algorithm outperforms the Q-learning algorithm.
- The time complexity is a possibly non-monotonic function of l, *i.e.*, increasing l does not always increase performance.
- The number of different Markov chains is inversely proportional to the time complexity.

For the double Q-learning (DQ) and weighted double Q-learning (WQ) algorithms, a similar analysis show that the complexity of these algorithms are worse than Q-learning because (i) there are 2 different Q-tables to be updated. For DQN, it is often necessary to do hyper-parameter optimization, and training. For NQ, finding a good weighting function also poses similar challenge.

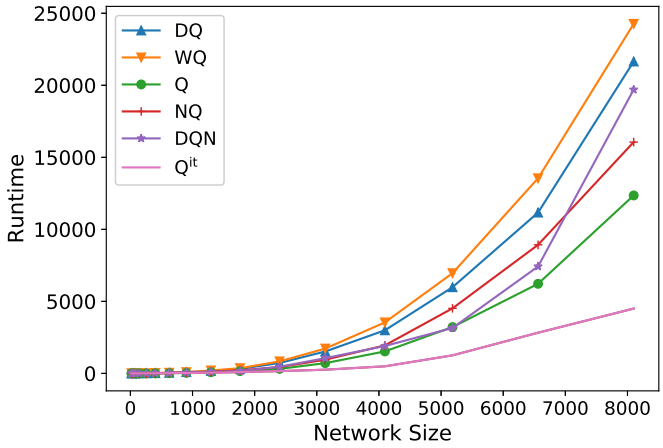

Fig. 1: Runtime complexity of different Q-learning methods

E. Runtime complexity of different Q-learning algorithms

In this section, we provide a simulation result on the runtime complexity of the Q-learning algorithms given in Table 1. The methods are run until they converge; thus, the number of iterations and the corresponding runtime are different for each method.

As can be seen from Fig.1, the proposed method achieves %70 reduction in runtime complexity for large state-spaces ($|\mathcal{S}| \approx 8000$). Q-learning is a trivial algorithm; thus, it is relatively faster than the others. DQN suffers from the training and hyper-parameter tuning stage, and DQ and WQ suffer from the use of double estimators.

F. Hyper-parameters for different Q-learning algorithms

For NQ, (13) in [1] is implemented with a weighting function having equal weights for only neighboring nodes. The implementation of DQN is similar to the one in [2], which uses the following hyper-parameters with an arbitrary number of transmitters (R): 2^R input nodes in the input layer, 2^R output nodes in the final layer, $R+1$ hidden layers each of which having 48 nodes. Each layer is followed by a ReLU function except for the final layer which uses a softmax activation function. The initial learning rate is chosen as 0.001 with an exponential decay with a decay rate of 0.995. The batch size is 20, the number of epochs is 20, and the number of simulations carried out is 50. No drop-out is used. L2 regularization is employed with a coefficient of 0.001.

REFERENCES

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

[2] 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.