Appendix

Talha Bozkus and Urbashi Mitra

I. Proofs

A. Proof of Proposition 1

By the properties of Q-learning algorithm, Q-functions and policies of each Q-learning algorithm on different VMEs converge to their optimal values, *i.e.* $\mathbf{Q}_t^{(n)} \to \mathbf{Q}^{(n)^*}$ and $\boldsymbol{\pi}^{(n)} \to \boldsymbol{\pi}^{(n)^*}$. Note that $\mathbf{Q}^* = \mathbf{Q}^{(1)^*}$ and $\boldsymbol{\pi}^* = \boldsymbol{\pi}^{(1)^*}$ as the original Q-learning algorithm operates in the original environment (1^{st} VME). By Proposition 5, the Q-functions and policies of Algorithm 1 converge to their true values: $\mathbf{Q}^{it} \to \mathbf{Q}^*$ and $\hat{\boldsymbol{\pi}} \to \boldsymbol{\pi}^*$.

The weights in Algorithm 1 are computed by comparing the $\hat{\boldsymbol{\pi}}$ with $\boldsymbol{\pi}^{(n)}$ at each iteration. Hence, weights converge by the convergence of $\hat{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}^{(n)}$.

We know that the optimal policies are solutions to the following equation [1]:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\mu}} (\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\mu}})^{-1} \mathbf{c}_{\boldsymbol{\mu}}, \tag{1}$$

where $\mathbf{P}_{\boldsymbol{\mu}}$ is the PTM under the policy $\boldsymbol{\mu}$, and $\mathbf{c}_{\boldsymbol{\mu}}$ is the cost vector under the policy $\boldsymbol{\mu}$. Thus, if we replace the PTT $\mathbf{P}$ in (1) by the transition tensors for different VMEs, which are the l_1-normalized n^{th} order co-link tensors $\tilde{\mathbf{L}}^{(n)}$, and use the explicit expression for the softmax function, we obtain the final values for the weights.

B. Proof of Proposition 2

The following analysis is valid for any (s,a); thus, the notation (s,a) is dropped. Using the update rule of Algorithm 1, the following equalities can be shown:

$$\mathbf{Q}_1^{\mathbf{it}} = u\mathbf{Q}_0^{\mathbf{it}} + (1-u)\sum_{n=1}^{K} \mathbf{w}_0^{(n)} \mathbf{Q}_0^{(n)}.$$

$$\mathbf{Q}_2^{\mathbf{it}} = u\mathbf{Q}_1^{\mathbf{it}} + (1-u)\sum_{n=1}^{K} \mathbf{w}_1^{(n)} \mathbf{Q}_1^{(n)} = (1-u)\Big[u\sum_{n=1}^{K} \mathbf{w}_0^{(n)} \mathbf{Q}_0^{(n)} + \sum_{n=1}^{K} \mathbf{w}_1^{(n)} \mathbf{Q}_1^{(n)}\Big], \tag{2}$$

which can be generalized to arbitrary t as follows:

$$\mathbf{Q}_{t+1}^{it} = (1-u)\sum_{i=0}^{t} u^{t-i} \sum_{n=1}^{K} \mathbf{w}_i^{(n)} \mathbf{Q}_i^{(n)}. \tag{3}$$

All Q-functions are initialized to 0. Hence, $\mathbf{Q}_0^{\mathbf{it}} = \mathbf{Q}_0^{(n)} = 0$ for all n. Let $\Delta_{t-1}^{it} = \mathbf{Q}_{t-1}^{\mathbf{it}} - \mathbf{Q}_t^{\mathbf{it}}$ and $\epsilon_{t-1}^{(n)} = \mathbf{w}_{t-1}^{(n)}\mathbf{Q}_{t-1}^{(n)} - \mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)}$. If we multiply the expression for $\mathbf{Q}_{t-1}^{it}$ from (3) by u, and subtract from the expression for $\mathbf{Q}_t^{it}$ from (3) side by side, we obtain the following:

$$\mathbf{Q}_t^{it} - u\mathbf{Q}_{t-1}^{it} = (1-u)\sum_{n=1}^{K} \mathbf{w}_{t-1}^{(n)} \mathbf{Q}_{t-1}^{(n)}. \tag{4}$$

If we rewrite the expression (4) with the variable change $t \to t+1$, and subtract the new expression from (4) side by side, we obtain the following:

$$u\Delta_{t-1}^{it} - \Delta_t^{it} = (1-u)\sum_{n=1}^{K} \Big[\mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)} - \mathbf{w}_{t-1}^{(n)}\mathbf{Q}_{t-1}^{(n)}\Big]. \tag{5}$$

$$\Delta_t^{it} = u\Delta_{t-1}^{it} + (1-u)\sum_{n=1}^{K} \epsilon_{t-1}^{(n)}. \tag{6}$$

$$< \Delta_{t-1}^{it} + (1-u)\sum_{n=1}^{K} \epsilon_{t-1}^{(n)}, \tag{7}$$

where (6) follows from the definition of $\epsilon_{t-1}^{(n)}$, and (7) follows from the fact that $u \in [0, 1)$. Then, the difference between consecutive updates can be bounded as follows:

$$\Delta_t^{it} - \Delta_{t-1}^{it} < (1-u) \sum_{n=1}^{K} \epsilon_{t-1}^{(n)}. \tag{8}$$

$$|\Delta_t^{it} - \Delta_{t-1}^{it}| < (1-u) \sum_{n=1}^{K} |\epsilon_{t-1}^{(n)}|, \tag{9}$$

where (9) follows from the triangle inequality. By the convergence of Q-learning on each VME, $\mathbf{Q}_{t-1}^{(n)} \xrightarrow{t\to\infty} \mathbf{Q}_t^{(n)}$, and by the convergence of weights from Proposition 1, $\mathbf{w}_{t-1}^{(n)} \xrightarrow{t\to\infty} \mathbf{w}_t^{(n)}$ for all n. Thus, $|\epsilon_{t-1}^{(n)}| \xrightarrow{t\to\infty} 0$, and $|\Delta_t^{it} - \Delta_{t-1}^{it}| \xrightarrow{t\to\infty} 0$.

C. Proof of Proposition 3

Using (5) for $t = 1, 2$, we obtain the following:

$$u\Delta_0^{it} - \Delta_1^{it} = -(1-u) \sum_{n=1}^{K} \epsilon_0^{(n)}. \tag{10}$$

$$u\Delta_1^{it} - \Delta_2^{it} = -(1-u) \sum_{n=1}^{K} \epsilon_1^{(n)}. \tag{11}$$

Multiplying (10) by u, and summing with (11), we obtain the following:

$$u^2\Delta_0^{it} - \Delta_2^{it} = -(1-u) \Big[u \sum_{n=1}^{K} \epsilon_0^{(n)} + \sum_{n=1}^{K} \epsilon_1^{(n)} \Big]. \tag{12}$$

Carrying out similar operations, we can show the following for arbitrary t:

$$u^t\Delta_0^{it} - \Delta_t^{it} = -(1-u) \sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \epsilon_{t-k-1}^{(n)}. \tag{13}$$

$$|\Delta_t^{it}| = (1-u) \Big| \sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \epsilon_{t-k-1}^{(n)} \Big|, \tag{14}$$

where (14) follows from $\Delta_0^{it} = 0$, and taking the absolute values of both sides. We choose constants $\theta^{(n)}$ for all n, t such that $|\epsilon_t^{(n)}| \leq \theta^{(n)}$, and proceed as follows:

$$|\Delta_t^{it}| \leq (1-u) \sum_{k=0}^{t-1} u^k \Big| \sum_{n=1}^{K} \epsilon_{t-k-1}^{(n)} \Big|. \tag{15}$$

$$\leq (1-u) \sum_{k=0}^{t-1} u^k \sum_{n=1}^{K} \theta^{(n)}. \tag{16}$$

$$\leq \sum_{n=1}^{K} \theta^{(n)} (1-u^t), \tag{17}$$

where (15) follows from the triangle inequality, (16) follows from the $|\epsilon_t^{(n)}| \leq \theta^{(n)}$ bound, and (17) follows from the fact that $u \in [0, 1)$. These results imply that $|\Delta_t^{it}| \leq \sum_{n=1}^{K} \theta^{(n)}$ as $t \to \infty$. Note that the constants $\theta^{(n)}$ always exist; hence, this bound is always feasible.

We can also find the time (or iteration) when the magnitude of Q-function updates of Algorithm 1 will be at most some positive constant β. By setting $\beta = \sum_{n=1}^{K} \theta^{(n)} (1-u^t)$, we obtain the following:

$$t = \frac{log(1 - \frac{\beta}{\sum_{n=1}^{K} \theta^{(n)}})}{log(u)}. \tag{18}$$

D. Proof of Proposition 4

If there exist constants $\phi^{(n)} \in (0,1)$ such that $|\epsilon_t^{(n)}| \leq \phi^{(n)}|\epsilon_{t-1}^{(n)}|$ for all n, t, then we choose $\phi = \max_n \phi^{(n)}$, and proceed as follows:

$$|\Delta_t^{it}| \leq (1-u)\sum_{k=0}^{t-1} u^k \Big|\sum_{n=1}^{K} \epsilon_{t-k-1}^{(n)}\Big|. \tag{19}$$

$$\leq (1-u)\sum_{k=0}^{t-1} u^k \Big|\sum_{n=1}^{K} \epsilon_0^{(n)}\phi^{t-k-1}\Big|. \tag{20}$$

$$\leq (1-u)\sum_{k=0}^{t-1} u^k \phi^{t-k-1}\Big|\sum_{n=1}^{K} \epsilon_0^{(n)}\Big|, \tag{21}$$

where (19) follows from (15), (20) follows from $\phi = \max_n \phi^{(n)}$ and $|\epsilon_t^{(n)}| \leq \phi^t|\epsilon_0^{(n)}|$. Let $u = \phi$. Then:

$$|\Delta_t^{it}| \leq (1-u)\sum_{k=0}^{t-1} \phi^{(t-1)}\Big|\sum_{n=1}^{K} \epsilon_0^{(n)}\Big|. \tag{22}$$

$$\leq (1-u)t\phi^{(t-1)}\Big|\sum_{n=1}^{K} \epsilon_0^{(n)}\Big|. \tag{23}$$

As $t \to \infty$, $t\phi^{(t-1)} \to 0$. Thus, $|\Delta_t^{it}| \to 0$. Note that the constants $\phi^{(n)} \in (0,1)$ may not exist; hence, this bound may not always be attainable.

E. Proof of Proposition 5

The following analysis is valid for any (s,a); thus, the notation (s,a) is dropped. Recall the main probabilistic assumption:

$$\mathcal{X}_t^{(n)} \overset{\text{def}}{=} \mathbf{Q}_t^{(n)}(s,a) - \mathbf{Q}^*(s,a) \sim \text{unif}[-\lambda_n, \lambda_n],\ \forall n, t, s, a. \tag{24}$$

$$\mathbb{E}[\mathcal{X}_t^{(n)}] = 0,\ \forall n, t. \tag{25}$$

$$\mathbb{V}[\mathcal{X}_t^{(n)}] = \frac{\lambda_n^2}{3},\ \forall n, t. \tag{26}$$

$$\lim_{t\to\infty} \mathcal{E}_t = \lim_{t\to\infty} (\mathbf{Q}_t^{it} - \mathbf{Q}^*). \tag{27}$$

$$= \lim_{t\to\infty} (1-u)\Big[u^t \sum_n \mathbf{w}_0^{(n)}\mathbf{Q}_0^{(n)} + u^{t-1}\sum_n \mathbf{w}_1^{(n)}\mathbf{Q}_1^{(n)} + ... + u^0 \sum_n \mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)} - \mathbf{Q}^*\Big]. \tag{28}$$

$$= \lim_{t\to\infty} (1-u)\Big[u^t \sum_n \mathbf{w}_0^{(n)}(\mathbf{Q}_0^{(n)} - \mathbf{Q}^*) + u^{t-1}\sum_n \mathbf{w}_1^{(n)}(\mathbf{Q}_1^{(n)} - \mathbf{Q}^*) + ... + u^0 \sum_n \mathbf{w}_t^{(n)}(\mathbf{Q}_t^{(n)} - \mathbf{Q}^*)\Big], \tag{29}$$

where (28) follows from the explicit expression of $\mathbf{Q}_t^{it}$ from (3), (29) follows from the fact that the weights sum to 1, *i.e.* $\sum_n w_t^{(n)} = 1$ for all t, and $(1-u)\sum_t u^t = 1$ as $t \to \infty$. Taking the expectation of both sides, we obtain the following:

$$\lim_{t\to\infty} \mathbb{E}[\mathcal{E}_t] = \lim_{t\to\infty} \mathbb{E}[\mathbf{Q}_t^{it} - \mathbf{Q}^*]. \tag{30}$$

$$= \lim_{t\to\infty} \mathbb{E}\Big[(1-u)\big[u^t \sum_n \mathbf{w}_0^{(n)}\mathbf{Q}_0^{(n)} + u^{t-1}\sum_n \mathbf{w}_1^{(n)}\mathbf{Q}_1^{(n)} + ... + u^0 \sum_n \mathbf{w}_t^{(n)}\mathbf{Q}_t^{(n)} - \mathbf{Q}^*\big]\Big]. \tag{31}$$

$$= \lim_{t\to\infty} (1-u)\Big[u^t \sum_n \mathbf{w}_0^{(n)}\mathbb{E}[\mathbf{Q}_0^{(n)} - \mathbf{Q}^*] + u^{t-1}\sum_n \mathbf{w}_1^{(n)}\mathbb{E}[\mathbf{Q}_1^{(n)} - \mathbf{Q}^*] + ... + u^0 \sum_n \mathbf{w}_t^{(n)}\mathbb{E}[\mathbf{Q}_t^{(n)} - \mathbf{Q}^*]\Big]. \tag{32}$$

$$= 0, \tag{33}$$

where (32) follows from the linearity of expectation, and (33) follows from (24) and (25).

We now derive the upper bound on the variance.

$$\lim_{t\to\infty}\mathbb{V}[\mathcal{E}_t]=\lim_{t\to\infty}\mathbb{V}\Big[(1-u)\big[u^t\sum_n\mathbf{w}_0^{(n)}(\mathbf{Q}_0^{(n)}-\mathbf{Q}^*)+u^{t-1}\sum_n\mathbf{w}_1^{(n)}(\mathbf{Q}_1^{(n)}-\mathbf{Q}^*)+...+u^0\sum_n\mathbf{w}_t^{(n)}(\mathbf{Q}_t^{(n)}-\mathbf{Q}^*)\big]\Big]. \tag{34}$$

$$\begin{aligned}
&=\lim_{t\to\infty}(1-u)^2\Big[u^{2t}\big[\sum_n(\mathbf{w}_0^{(n)})^2\mathbb{V}[\mathbf{Q}_0^{(n)}-\mathbf{Q}^*]+2\sum_n\sum_{m\neq n}\mathbf{w}_0^{(n)}\mathbf{w}_0^{(m)}\mathrm{Cov}(\mathbf{Q}_0^{(n)}-\mathbf{Q}^*,\mathbf{Q}_0^{(m)}-\mathbf{Q}^*)\big]\\
&\qquad+u^{2(t-1)}\big[\sum_n(\mathbf{w}_1^{(n)})^2\mathbb{V}[\mathbf{Q}_1^{(n)}-\mathbf{Q}^*]+2\sum_n\sum_{m\neq n}\mathbf{w}_1^{(n)}\mathbf{w}_1^{(m)}\mathrm{Cov}(\mathbf{Q}_1^{(n)}-\mathbf{Q}^*,\mathbf{Q}_1^{(m)}-\mathbf{Q}^*)\big]\\
&\qquad\quad+...\\
&\qquad\quad+u^0\big[\sum_n(\mathbf{w}_t^{(n)})^2\mathbb{V}[\mathbf{Q}_t^{(n)}-\mathbf{Q}^*]+2\sum_n\sum_{m\neq n}\mathbf{w}_t^{(n)}\mathbf{w}_t^{(m)}\mathrm{Cov}(\mathbf{Q}_t^{(n)}-\mathbf{Q}^*,\mathbf{Q}_t^{(m)}-\mathbf{Q}^*)\big]\Big].
\end{aligned}\tag{35}$$

$$\begin{aligned}
&=\lim_{t\to\infty}(1-u)^2\Big[u^{2t}\big[\sum_n(\mathbf{w}_0^{(n)})^2\mathbb{V}[\mathcal{X}_0^{(n)}]+2\sum_n\sum_{m\neq n}\mathbf{w}_0^{(n)}\mathbf{w}_0^{(m)}\mathrm{Cov}(\mathcal{X}_0^{(n)},\mathcal{X}_0^{(m)})\big]\\
&\qquad+u^{2(t-1)}\big[\sum_n(\mathbf{w}_1^{(n)})^2\mathbb{V}[\mathcal{X}_1^{(n)}]+2\sum_n\sum_{m\neq n}\mathbf{w}_1^{(n)}\mathbf{w}_1^{(m)}\mathrm{Cov}(\mathcal{X}_1^{(n)},\mathcal{X}_1^{(m)})\big]\\
&\qquad\quad+...\\
&\qquad\quad+u^0\big[\sum_n(\mathbf{w}_t^{(n)})^2\mathbb{V}[\mathcal{X}_t^{(n)}]+2\sum_n\sum_{m\neq n}\mathbf{w}_t^{(n)}\mathbf{w}_t^{(m)}\mathrm{Cov}(\mathcal{X}_t^{(n)},\mathcal{X}_t^{(m)})\big]\Big].
\end{aligned}\tag{36}$$

$$\begin{aligned}
&\le\lim_{t\to\infty}(1-u)^2\Big[u^{2t}\big[\sum_n\mathbf{w}_0^{(n)}\mathbb{V}[\mathcal{X}_0^{(n)}]+2\sum_n\sum_m\mathbf{w}_0^{(n)}\mathbf{w}_0^{(m)}\sqrt{\mathbb{V}[\mathcal{X}_0^{(n)}]\mathbb{V}[\mathcal{X}_0^{(m)}]}\big]\\
&\qquad\quad+u^{2(t-1)}\big[\sum_n\mathbf{w}_1^{(n)}\mathbb{V}[\mathcal{X}_1^{(n)}]+2\sum_n\sum_m\mathbf{w}_1^{(n)}\mathbf{w}_1^{(m)}\sqrt{\mathbb{V}[\mathcal{X}_1^{(n)}]\mathbb{V}[\mathcal{X}_1^{(m)}]}\big]\\
&\qquad+...\\
&\qquad\quad+u^0\big[\sum_n\mathbf{w}_t^{(n)}\mathbb{V}[\mathcal{X}_t^{(n)}]+2\sum_n\sum_m\mathbf{w}_t^{(n)}\mathbf{w}_t^{(m)}\sqrt{\mathbb{V}[\mathcal{X}_t^{(n)}]\mathbb{V}[\mathcal{X}_t^{(m)}]}\big]\Big].
\end{aligned}\tag{37}$$

$$\begin{aligned}
&\le\lim_{t\to\infty}(1-u)^2\Big[u^{2t}\big[\sum_n\mathbf{w}_0^{(n)}\frac{\lambda^2}{3}+2\sum_n\sum_m\mathbf{w}_0^{(n)}\mathbf{w}_0^{(m)}\frac{\lambda^2}{3}\big]\\
&\qquad+u^{2(t-1)}\big[\sum_n\mathbf{w}_1^{(n)}\frac{\lambda^2}{3}+2\sum_n\sum_m\mathbf{w}_1^{(n)}\mathbf{w}_1^{(m)}\frac{\lambda^2}{3}\big]\\
&\qquad+...\\
&\qquad\quad+u^0\big[\sum_n\mathbf{w}_t^{(n)}\frac{\lambda^2}{3}+2\sum_n\sum_m\mathbf{w}_t^{(n)}\mathbf{w}_t^{(m)}\frac{\lambda^2}{3}\big]\Big].
\end{aligned}\tag{38}$$

$$\le\lim_{t\to\infty}(1-u)^2\Big[u^{2t}\lambda^2+u^{2(t-1)}\lambda^2+...+u^0\lambda^2\Big]. \tag{39}$$

$$\le\lim_{t\to\infty}\frac{(1-u)}{(1+u)}\lambda^2. \tag{40}$$

where (35) follows from the properties of variance operator, (36) follows from (24), (37) follows from the fact that $|\mathbf{w}_t^{(n)}|\le 1$, and Cauchy-Schwarz inequality for the covariance operator, (38) follows from $\lambda=\max_n\lambda_n$ and (26), (39) follows from $\sum_n\mathbf{w}_t^{(n)}=1$ for all t, (40) follows from the infinite geometric sum formula.

If we use a time-varying update ratio in the following structure: $u_t=1-e^{\frac{-t}{c_4}}$ with $c_4>0$, $u_t\xrightarrow{t\to\infty}1$. This is in line with the intuition that u should be small enough to exploit multiple VMEs in the beginning, and then increase to utilize previously obtained samples. This is similar to the exploration-exploitation trade-off with the difference that exploration here refers to collecting samples from different VMEs, and exploitation refers to using the cumulative Q-function output. Therefore,

$$\lim_{t\to\infty}\mathbb{V}[\mathcal{E}_t]=0. \tag{41}$$

Algorithm	Objective	Strategy	
Simple Q (Q)	-	Single	-
Speedy Q (SQ)	Convergence rate	Single	[2]
Neural Fitted Q (NQ)	Data efficiency	Single	[3]
Double Q (DQ)	Bias	Multi	[4]
MaxMin Q (MMQ)	Bias & variance	Multi	[5]
Ensemble Bootst. Q (EBQ)	Bias	Multi	[6]
Averaged DQN (ADQN)	Stability & variance	Multi	[7]
Bootstrapped DQN (BDQN)	Learning speed	Multi	[8]

TABLE I: Q-learning algorithm and variants

Note that (41) can also be shown by starting the derivations with the time-varying update ratio u and using an upper bound on u_t for all t.

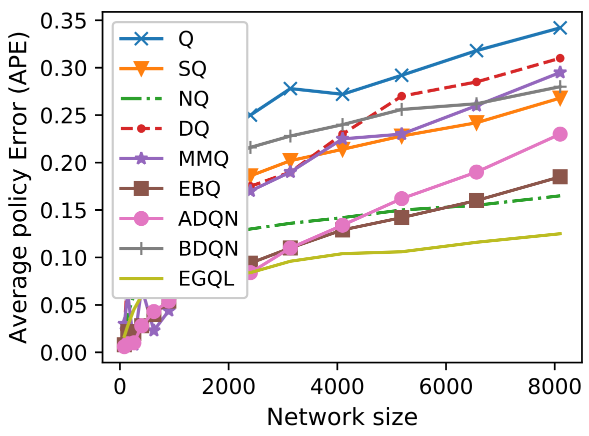
II. Simulation results with more benchmarks

In this section, we carry out the same simulations with more benchmark algorithms as given in Table I. In addition to the algorithms considered in the paper, we also employ Double Q-learning [4], Speedy Q-learning [2] and MaxMin Q-learning [5]. The APE performances for three different models are given in Fig.1a, Fig.1b, and Fig.1c, and the common runtime complexity result is shown in Fig. 1d. We observe that the new benchmarks also have inferior APE performances and runtime complexities compared to EGQL.

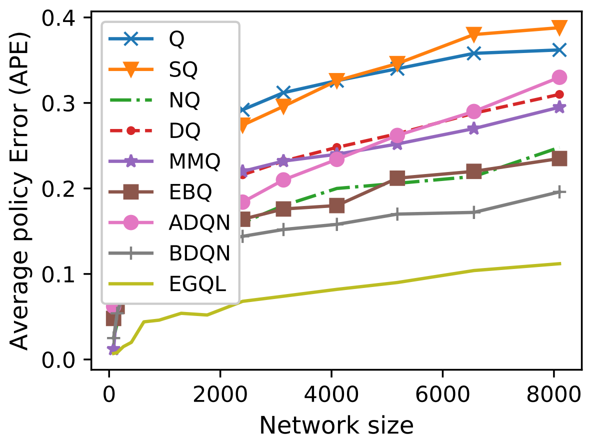
III. Hyper-parameter Tuning and other details

We herein explain how to select and optimize the hyper-parameter for different Q-learning algorithms. Since we carry out the simulations across different network sizes, there is no single set of hyper-parameters that fit all simulations. Hence, we give a reasonable range for each parameter across different settings and find the optimal one by trial-error.

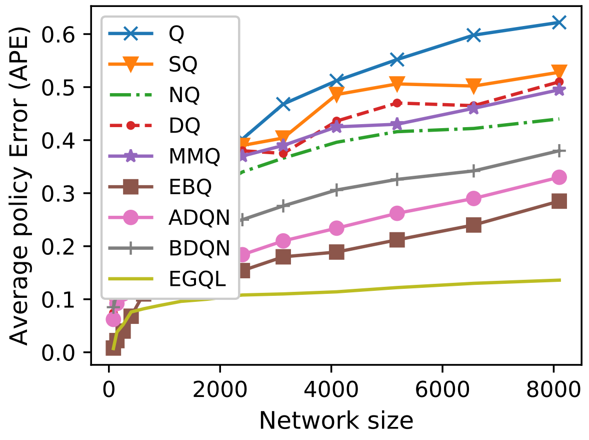
- The probability parameter (p) is used in model (iii) and set to $p = 0.8$.
- The discount factor is used in all algorithms and set to γ = 0.9.
- We define the following networks: small-sized networks: $|\mathcal{S}| \leq$ 1000, modest-sized networks: $|\mathcal{S}| \in$ [1000,10000], and large networks: $|\mathcal{S}| \geq$ 10000. The parameters v and l in EGQL are optimized through cross-validation from the following sets: for small networks: $v, l \in [5, 10]$, for modest-sized networks: $v, l \in [10, 20]$, and for large networks: $v, l \in [15, 25]$.
- The parameters learning rate (α_t) and exploration probability for the epsilon-greedy policy (ϵ_t) have common structures for all algorithms: $\alpha_t = \frac{1}{1+\frac{t}{c_1}}$, where $c_1 > 0$ determines the decay rate. Note that the form of learning rate obeys the conditions for the convergence of Q-learning. $\epsilon_t = \max((c_2)^t, c_3)$, where the constant $c_2 > 0$ adjusts the decay rate, and $0 < c_3 \ll 1$ determines the minimum exploration probability. In general, c_1 and c_2 should increase as the system parameters increase. In addition, c_3 should be small but positive so that the algorithm can still explore with a very small probability when the policy is nearly converged. The parameters c_1, c_2 and c_3 are optimized for different network sizes from the following sets: $c_1 \in \{1, 5 \cdot 10^1, 10^2, 5 \cdot 10^2, 10^3, 5 \cdot 10^3, 10^4\}$, $c_2 \in \{0.9, 0.95, 0.99, 0.995, 0.999\}$ and $c_3 \in \{10^{-2}, 10^{-1}\}$.
- The following parameters are common to most of the algorithms, and selected via cross-validation from the following sets: batch size: $\{8, 16, 32, 64, 128\}$, replay buffer memory: $\{5 \cdot 10^3, 10^4, 2 \cdot 10^4\}$, target network update frequency: $\{500, 1000, 5000\}$.
- The number of estimators for MMQ, EBQ, ADQN, BDQN is selected via cross-validation for different network sizes as follows: for small networks: $\{2, 3, 4\}$, for modest-sized networks: $\{3, 4, 5, 6\}$, and for large networks: $\{5, 6, 7, 8, 9, 10\}$.
- For NQ, ADQN, and BDQN, we employ the following models for the neural network implementations: For small networks: 3-layer fully connected NN with 48 nodes in each layer followed by ReLU, for modest-sized networks: 4-layer fully connected NN with 48 nodes in each layer followed by ReLU, for large networks: 4-layer fully connected NN with 96 nodes in each layer followed by ReLU functions. We use the normalized set of buffer and channel states for R different transmitters as input to the network as in [9] so that the number of nodes in the input layer is $2R$. The output layer has 2^R nodes followed by a softmax function, where each node represents the probability of a specific action being the optimal one. We use a dropout with a probability 0.25 after each layer except the final layer, and no



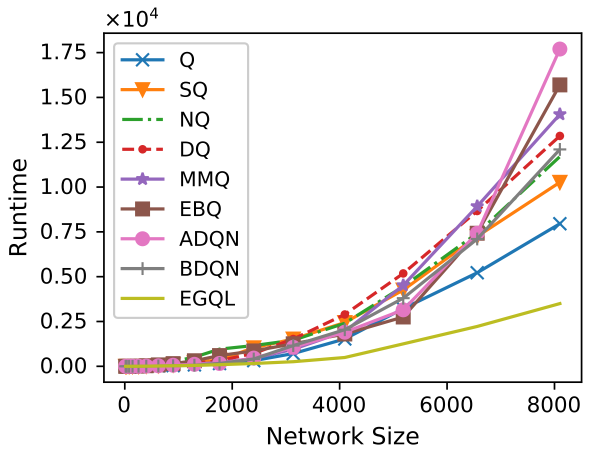
(a) APE for model (i)



(b) APE for model (ii)



(c) APE for model (iii)



(d) Runtime

Fig. 1: Performance results across different network models

regularization is used. The Adam optimizer is used to update the weight of neural networks. The other small parameters closely follow the ones given in [10].

- For the proposed algorithm (EBQL), we assume the following form for the time-varying update ratio: $u_t = 1 - e^{\frac{-t}{c_4}}$ with constant $c_4 > 0$ determining the grow rate. As the system parameters increase, c_4 should increase to ensure adequate exploitation of multiple VMEs. In particular, c_4 is selected from the following set, and optimized via cross-validation: $\{10^2, 5 \cdot 10^2, 10^3, 5 \cdot 10^3, 10^4\}$. On the other hand, the number of VMEs (K) is selected from the following sets via cross-validation: for small networks: $K \in [2, 5]$, for modest-sized networks: $K \in [3, 8]$, and for large networks: $K \in [5, 12]$.
- For practical purposes, we carry out extensive simulations and provided some of the near-optimal parameters in Table II, which can be directly used to achieve near-optimal performance. The simulations are carried out across different network sizes as shown in Fig.2 and Fig.3, where the network size is obtained as follows: the interval [0, 2000] is formed by increasing N, M comparably with $R = 1$, [2000, 4000] is formed similarly with $R = 2$, [4000, 800] is formed similarly with $R = 3$. In addition, all simulations are carried out with the optimal parameters 100 times, and the APE and runtime results are averaged over.
- To make a fair performance comparison between the different models, we choose $|\mathcal{A}|$ comparable between three different models. In particular, $|\mathcal{A}| = 4$ is fixed for model (ii). For model (i), we also choose $|\mathcal{A}| = 4$. For model (iii), $|\mathcal{A}| = 2^R$. Thus, we choose $R \in \{1, 2\}$.

Parameters	Structure	Small networks	Modest-sized networks	Large networks
α_t	$\frac{1}{1+\frac{t}{c_1}}$	$c_1 = 10^3$	$c_1 = 5 \cdot 10^3$	$c_1 = 10^4$
ϵ_t	$\max((c_2)^t, c_3)$	$c_2 = 0.9$ $c_3 = 0.1$	$c_2 = 0.99$ $c_3 = 0.01$	$c_2 = 0.999$ $c_3 = 0.01$
u_t	$1 - e^{\frac{-t}{c_4}}$	$c_4 = 10^2$	$c_4 = 10^3$	$c_4 = 5 \cdot 10^3$

TABLE II: Some of the near-optimal hyper-parameters

References

[1] Dimitri Bertsekas. Dynamic programming and optimal control 3rd edition, volume ii. 04 2010.

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

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

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

[5] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

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

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

[8] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[9] T. Bozkus and U. Mitra. Link analysis for solving multiple-access mdps with large state spaces. https://github.com/talhabozkus/Link-Analysis-For-Solving-Multiple-Access-MDPs-With-Large-State-Spaces, 2022.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.