Appendix to *Weakly Coupled Deep Q-Networks*

A Proofs

A.1 Proof of Proposition 1

Proof. We prove part the first part of the proposition (weak duality) by induction. First, define

$$Q_0^*(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) \quad \text{and} \quad Q_0^\lambda(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \Big[\boldsymbol{b}(w) - \textstyle\sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\Big],$$

and suppose we run value iteration for both systems:

$$\begin{aligned} Q_{t+1}^*(\boldsymbol{s},\boldsymbol{a}) &= r(\boldsymbol{s},\boldsymbol{a}) + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}(\boldsymbol{s}')} Q_t^*(\boldsymbol{s}',\boldsymbol{a}')\big], \\ Q_{t+1}^\lambda(\boldsymbol{s},\boldsymbol{a}) &= r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \Big[\boldsymbol{b}(w) - \textstyle\sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\Big] + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}} Q_t^\lambda(\boldsymbol{s}',\boldsymbol{a}')\big]. \end{aligned}$$

It is well-known that, by the value iteration algorithm's convergence,

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = \lim_{t\to\infty} Q_t^*(\boldsymbol{s},\boldsymbol{a}) \quad \text{and} \quad Q^\lambda(\boldsymbol{s},\boldsymbol{a}) = \lim_{t\to\infty} Q_t^\lambda(\boldsymbol{s},\boldsymbol{a}).$$

Consider a state $\boldsymbol{s}\in\mathcal{S}$ and a feasible action $\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s})$. We have,

$$Q_0^\lambda(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \Big[\boldsymbol{b}(w) - \textstyle\sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\Big] \geq r(\boldsymbol{s},\boldsymbol{a}) = Q_0^*(\boldsymbol{s},\boldsymbol{a}).$$

Suppose $Q_t^\lambda(\boldsymbol{s},\boldsymbol{a}) \geq Q_t^*(\boldsymbol{s},\boldsymbol{a})$ holds for all $\boldsymbol{s}\in\mathcal{S}$ and $\boldsymbol{a}\in\mathcal{A}(\boldsymbol{s})$ for some $t>0$ (induction hypothesis). Then,

$$\begin{aligned} Q_{t+1}^\lambda(\boldsymbol{s},\boldsymbol{a}) &= r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \Big[\boldsymbol{b}(w) - \textstyle\sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\Big] + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}} Q_t^\lambda(\boldsymbol{s}',\boldsymbol{a}')\big] \\ &\geq r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \Big[\boldsymbol{b}(w) - \textstyle\sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\Big] + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}(\boldsymbol{s}')} Q_t^*(\boldsymbol{s}',\boldsymbol{a}')\big] \\ &\geq r(\boldsymbol{s},\boldsymbol{a}) + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}(\boldsymbol{s}')} Q_t^*(\boldsymbol{s}',\boldsymbol{a}')\big] = Q_{t+1}^*(\boldsymbol{s},\boldsymbol{a}). \end{aligned}$$

Thus, it follows that $Q^\lambda(\boldsymbol{s},\boldsymbol{a}) \geq Q^*(\boldsymbol{s},\boldsymbol{a})$.

For the proof of the second part of the proposition, define

$$\boldsymbol{B}_0(w) = \boldsymbol{b}(w) \quad \text{and} \quad \boldsymbol{B}_{t+1}(w) = \boldsymbol{b}(w) + \gamma\,\mathbb{E}\big[\boldsymbol{B}_t(w')\big].$$

We use an induction proof. We have for all $(\boldsymbol{s},\boldsymbol{a})\in\mathcal{S}\times\mathcal{A}$,

$$\begin{aligned} Q_0^\lambda(\boldsymbol{s},\boldsymbol{a}) &= r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \left[\boldsymbol{b}(w) - \sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\right] \\ &= \sum_{i=1}^N \Big[r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i)\Big] + \lambda^\intercal \boldsymbol{b}(w) = \sum_{i=1}^N Q_{0,i}^\lambda(s_i,a_i) + \lambda^\intercal \boldsymbol{B}_0(w), \end{aligned}$$

where $Q_{0,i}^\lambda(s_i,a_i) = r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i)$. Similarly, for all $(\boldsymbol{s},\boldsymbol{a})\in\mathcal{S}\times\mathcal{A}$,

$$\begin{aligned} Q_1^\lambda(\boldsymbol{s},\boldsymbol{a}) &= r(\boldsymbol{s},\boldsymbol{a}) + \lambda^\intercal \left[\boldsymbol{b}(w) - \sum_{i=1}^N \boldsymbol{d}(s_i,a_i)\right] + \gamma\,\mathbb{E}\big[\max_{\boldsymbol{a}'\in\mathcal{A}} Q_0^\lambda(\boldsymbol{s}',\boldsymbol{a}')\big] \\ &= \sum_{i=1}^N \Big[r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i)\Big] + \lambda^\intercal \boldsymbol{b}(w) + \gamma\,\mathbb{E}\left[\max_{\boldsymbol{a}'\in\mathcal{A}} \left\{\sum_{i=1}^N Q_{0,i}^\lambda(s_i',a_i') + \lambda^\intercal \boldsymbol{B}_0(w')\right\}\right] \\ &= \sum_{i=1}^N \Big[r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i) + \gamma\,\mathbb{E}\big[\max_{a_i'\in\mathcal{A}_i} Q_{0,i}^\lambda(s_i',a_i')\big]\Big] + \lambda^\intercal \Big(\boldsymbol{b}(w) + \gamma\,\mathbb{E}\big[\boldsymbol{B}_0(w')\big]\Big) \\ &= \sum_{i=1}^N Q_{1,i}^\lambda(s_i,a_i) + \lambda^\intercal \boldsymbol{B}_1(w). \end{aligned}$$

Continuing in this manner, we arrive at $Q_t^\lambda(\boldsymbol{s},\boldsymbol{a}) = \sum_{i=1}^N Q_{t,i}^\lambda(s_i,a_i) + \lambda^\intercal \boldsymbol{B}_t(w)$. Finally, we have

$$\begin{aligned} Q^\lambda(\boldsymbol{s},\boldsymbol{a}) &= \lim_{t\to\infty} Q_t^\lambda(\boldsymbol{s},\boldsymbol{a}) \\ &= \lim_{t\to\infty} \sum_{i=1}^N Q_{t,i}^\lambda(s_i,a_i) + \lambda^\intercal \boldsymbol{B}_t(w) = \sum_{i=1}^N Q_i^\lambda(s_i,a_i) + \lambda^\intercal \boldsymbol{B}(w), \end{aligned}$$

which follows by the convergence of value iteration. □

A.2 Proof of Theorem 1

Proof. First, we define the Bellman operator H:

$$(HQ')(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathbb{E}\left[\max_{\boldsymbol{a}'} Q'(\boldsymbol{s}',\boldsymbol{a}')\right],$$

which is known to be a γ-contraction mapping. Next we define the random noise term

$$\xi_n(\boldsymbol{s},\boldsymbol{a}) = \gamma \max_{\boldsymbol{a}'} Q'_n(\boldsymbol{s}',\boldsymbol{a}') - \gamma\mathbb{E}\left[\max_{\boldsymbol{a}'} Q'_n(\boldsymbol{s}',\boldsymbol{a}')\right]. \tag{19}$$

Analogously, for subproblem $i \in \{1,\ldots,N\}$, define the subproblem Bellman operator

$$(H_i Q_i^\lambda)(s_i,a_i) = r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i) + \gamma\mathbb{E}\left[\max_{a'_i} Q_i^\lambda(s'_i,a'_i)\right],$$

and random noise term

$$\xi_{i,n}(s_i,a_i) = \gamma \max_{a'_i} Q'_{i,n}(s'_i,a'_i) - \gamma\mathbb{E}\left[\max_{a'_i} Q'_{i,n}(s'_i,a'_i)\right]. \tag{20}$$

The update rules of WCQL can then be written as

$$\begin{aligned} Q_{n+1}(\boldsymbol{s},\boldsymbol{a}) &= (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))\,Q'_n(\boldsymbol{s},\boldsymbol{a}) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\ \left[(HQ'_n)(\boldsymbol{s},\boldsymbol{a}) + \xi_{n+1}(\boldsymbol{s},\boldsymbol{a})\right], \\ Q_{i,n+1}^\lambda(s_i,a_i) &= Q_{i,n}^\lambda(s_i,a_i) + \beta_n(s_i,a_i)\left[(H_iQ'_{i,n})(s_i,a_i) + \xi_{i,n+1}(s_i,a_i)\right], \qquad (21) \\ Q_{n+1}^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}) &= \min_{\lambda\in\Lambda}\ \lambda^\intercal \boldsymbol{B}_n(w) + \sum_{i=1}^N Q_{i,n}^\lambda(s_i,a_i), \\ Q'_{n+1}(\boldsymbol{s},\boldsymbol{a}) &= \min(Q_{n+1}(\boldsymbol{s},\boldsymbol{a}), Q_{n+1}^{\lambda^*}(\boldsymbol{s},\boldsymbol{a})). \qquad (22) \end{aligned}$$

Parts (i) and (ii). By the iteration described in (21), we know that for a fixed λ, we are running Q-learning on an auxiliary MDP with Bellman operator H_i, which encodes a reward $r_i(s_i,a_i) - \lambda^\intercal \boldsymbol{d}(s_i,a_i)$ and the transition dynamics for subproblem i. By the standard result for asymptotic convergence of Q-learning [6], we have

$$\lim_{n\to\infty} Q_{i,n}^\lambda(s_i,a_i) = Q_i^\lambda(s_i,a_i). \tag{23}$$

We now prove the result in (ii): $\lim_{n\to\infty} Q_n^\lambda(\boldsymbol{s},\boldsymbol{a}) \ge Q^*(\boldsymbol{s},\boldsymbol{a})$. Recall that

$$Q_n^\lambda(\boldsymbol{s},\boldsymbol{a}) = \lambda^\intercal \boldsymbol{B}_n(w) + \sum_{i=1}^N Q_{i,n}^\lambda(s_i,a_i).$$

By standard stochastic approximation theory, $\lim_{n\to\infty} \boldsymbol{B}_n(w) = \boldsymbol{B}(w)$ for all w [38]. Combining this with (23), we have $\lim_{n\to\infty} Q_n^\lambda(\boldsymbol{s},\boldsymbol{a}) = Q^\lambda(\boldsymbol{s},\boldsymbol{a})$ for all $(\boldsymbol{s},\boldsymbol{a})$, and to conclude that this limit is an upper bound on $Q^*(\boldsymbol{s},\boldsymbol{a})$, we apply Proposition 1.

Part (iii). Assume without loss of generality that $Q^*(\boldsymbol{s},\boldsymbol{a}) = 0$ for all state-action pairs $(\boldsymbol{s},\boldsymbol{a})$. This can be established by shifting the origin of the coordinate system. We also assume that $\alpha_n(\boldsymbol{s},\boldsymbol{a}) \le 1$ for all $(\boldsymbol{s},\boldsymbol{a})$ and n. We proceed via induction. Note that the iterates $Q'_n(\boldsymbol{s},\boldsymbol{a})$ are bounded in the sense that there exists a constant $D_0 = R_{\max}/(1-\gamma)$, $R_{\max} = \max_{(\boldsymbol{s},\boldsymbol{a})}|r(\boldsymbol{s},\boldsymbol{a})|$, such that $|Q'_n(\boldsymbol{s},\boldsymbol{a})| \le D_0$ for all $(\boldsymbol{s},\boldsymbol{a})$ and iterations n [17]. Define the sequence $D_{k+1} = (\gamma+\epsilon)\,D_k$, such that $\gamma+\epsilon < 1$ and $\epsilon > 0$. Clearly, $D_k \to 0$. Suppose that there exists a random variable n_k, representing an iteration threshold such that for all $(\boldsymbol{s},\boldsymbol{a})$,

$$-D_k \le Q'_n(\boldsymbol{s},\boldsymbol{a}) \le \min\{D_k, Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a})\}, \quad \forall\, n \ge n_k.$$

We will show that there exists some iteration n_{k+1} such that

$$-D_{k+1} \le Q'_n(\boldsymbol{s},\boldsymbol{a}) \le \min\{D_{k+1}, Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a})\} \quad \forall\,(s,a),\; n \ge n_{k+1},$$

which implies that $Q'_n(\boldsymbol{s},\boldsymbol{a})$ converges to $Q^*(\boldsymbol{s},\boldsymbol{a}) = 0$ for all $(\boldsymbol{s},\boldsymbol{a})$.

By part (ii), we know that for all $\eta > 0$, with probability 1, there exists some finite iteration n_0 such that for all $n \ge n_0$,

$$Q^*(\boldsymbol{s},\boldsymbol{a}) - \eta \le Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}). \tag{24}$$

Now, we define an accumulated noise process started at n_k by $W_{n_k,n_k}(\boldsymbol{s},\boldsymbol{a}) = 0$, and

$$W_{n+1,n_k}(\boldsymbol{s},\boldsymbol{a}) = (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))\, W_{n,n_k}(\boldsymbol{s},\boldsymbol{a}) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\, \xi_{n+1}(\boldsymbol{s},\boldsymbol{a}), \quad \forall\, n \ge n_k, \tag{25}$$

where $\xi_n(\boldsymbol{s},\boldsymbol{a})$ is as defined in (19). Let $\mathcal{F}_n$ be the entire history of the algorithm up to the point where the step sizes at iteration n are selected. Using Corollary 4.1 in [6] which states that under Assumption 2 on the step size $\alpha_n(\boldsymbol{s},\boldsymbol{a})$, and if $\mathbb{E}[\xi_n(\boldsymbol{s},\boldsymbol{a}) \,|\, \mathcal{F}_n] = 0$ and $\mathbb{E}[\xi_n^2(\boldsymbol{s},\boldsymbol{a}) \,|\, \mathcal{F}_n] \le A_n$, where the random variable A_n is bounded with probability 1, the sequence $W_{n+1,n_k}(\boldsymbol{s},\boldsymbol{a})$ defined in (25) converges to zero, with probability 1. From our definition of the stochastic approximation noise $\xi_n(\boldsymbol{s},\boldsymbol{a})$ in (19), we have

$$\mathbb{E}[\xi_n(\boldsymbol{s},\boldsymbol{a}) \,|\, \mathcal{F}_n] = 0 \quad \text{and} \quad \mathbb{E}[\xi_n^2(\boldsymbol{s},\boldsymbol{a}) \,|\, \mathcal{F}_n] \le C(1 + \max_{\boldsymbol{s}',\boldsymbol{a}'} Q_n'^2(\boldsymbol{s}',\boldsymbol{a}')),$$

where C is a constant. Then, it follows that

$$\lim_{n\to\infty} W_{n,n_k}(\boldsymbol{s},\boldsymbol{a}) = 0 \quad \forall\,(\boldsymbol{s},\boldsymbol{a}),\; n_k.$$

We use the following lemma from [6] to bound the accumulated noise.

Lemma A.1 (Lemma 4.2 in [6])**.** *For every $\delta > 0$, with probability one, there exists some n' such that $|W_{n,n'}(\boldsymbol{s},\boldsymbol{a})| \le \delta$, for all $n \ge n'$.*

Now, by Lemma A.1, let $n_{k'} \ge \max(n_k, n_0)$ such that, for all $n \ge n_{k'}$ we have

$$|W_{n,n_{k'}}(\boldsymbol{s},\boldsymbol{a})| \le \gamma\epsilon D_k < \gamma D_k.$$

Let $\nu_k \ge n_{k'}$ such that, for all $n \ge \nu_k$, by (24) we have

$$\gamma\epsilon D_k - \gamma D_k \le Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}).$$

Define another sequence Y_n that starts at iteration ν_k.

$$Y_{\nu_k}(\boldsymbol{s},\boldsymbol{a}) = D_k \quad \text{and} \quad Y_{n+1}(\boldsymbol{s},\boldsymbol{a}) = (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))\, Y_n(\boldsymbol{s},\boldsymbol{a}) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\, \gamma\, D_k \tag{26}$$

Note that it is easy to show that the sequence $Y_n(\boldsymbol{s},\boldsymbol{a})$ in (26) is decreasing, bounded below by γD_k, and converges to γD_k as $n \to \infty$. Now we state the following lemma.

Lemma A.2. *For all state-action pairs $(\boldsymbol{s},\boldsymbol{a})$ and iterations $n \ge \nu_k$, it holds that:*

$$-Y_n(\boldsymbol{s},\boldsymbol{a}) + W_{n,\nu_k}(\boldsymbol{s},\boldsymbol{a}) \le Q'_n(\boldsymbol{s},\boldsymbol{a}) \le \min\{Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}), Y_n(\boldsymbol{s},\boldsymbol{a}) + W_{n,\nu_k}(\boldsymbol{s},\boldsymbol{a})\}. \tag{27}$$

Proof. We focus on the right hand side inequality, the left hand side can be proved similarly. For the base case $n = \nu_k$, the statement holds because $Y_{\nu_k}(\boldsymbol{s},\boldsymbol{a}) = D_k$ and $W_{\nu_k,\nu_k}(\boldsymbol{s},\boldsymbol{a}) = 0$. We assume it is true for n and show that it continues to hold for $n+1$:

$$\begin{aligned}
Q_{n+1}(\boldsymbol{s},\boldsymbol{a}) &= (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))Q'_n(\boldsymbol{s},\boldsymbol{a}) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\; [(HQ'_n)(\boldsymbol{s},\boldsymbol{a}) + \xi_{n+1}(\boldsymbol{s},\boldsymbol{a})] \\
&\le (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))\min\{Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}), Y_n(\boldsymbol{s},\boldsymbol{a}) + W_{n,\nu_k}(\boldsymbol{s},\boldsymbol{a})\} \\
&\qquad\qquad + \alpha_n(\boldsymbol{s},\boldsymbol{a})\,(HQ'_n)(\boldsymbol{s},\boldsymbol{a}) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\,\xi_{n+1}(\boldsymbol{s},\boldsymbol{a}) \\
&\le (1-\alpha_n(\boldsymbol{s},\boldsymbol{a}))\,(Y_n(\boldsymbol{s},\boldsymbol{a}) + W_{n,\nu_k}(\boldsymbol{s},\boldsymbol{a})) + \alpha_n(\boldsymbol{s},\boldsymbol{a})\,\gamma D_k + \alpha_n(\boldsymbol{s},\boldsymbol{a})\,\xi_{n+1}(\boldsymbol{s},\boldsymbol{a}) \\
&\le Y_{n+1}(\boldsymbol{s},\boldsymbol{a}) + W_{n+1,\nu_k}(\boldsymbol{s},\boldsymbol{a}),
\end{aligned}$$

where we used $(HQ'_n) \le \gamma\|Q'_n\| \le \gamma D_k$. Now, we have

$$\begin{aligned}
Q'_{n+1}(\boldsymbol{s},\boldsymbol{a}) &= \min(Q_{n+1}^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}), Q_{n+1}(\boldsymbol{s},\boldsymbol{a})) \\
&\le \min\{Q_{n+1}^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}), Y_{n+1}(\boldsymbol{s},\boldsymbol{a}) + W_{n+1,\nu_k}(\boldsymbol{s},\boldsymbol{a})\}.
\end{aligned}$$

The inequality holds because

$$Q_{n+1}(\boldsymbol{s},\boldsymbol{a}) \le Y_{n+1}(\boldsymbol{s},\boldsymbol{a}) + W_{n+1,\nu_k}(\boldsymbol{s},\boldsymbol{a}),$$

which completes the proof. □

Since $Y_n(\boldsymbol{s},\boldsymbol{a}) \to \gamma D_k$ and $W_{n,\nu_k}(\boldsymbol{s},\boldsymbol{a}) \to 0$, we have

$$\limsup_{n\to\infty} \|Q'_n\| \le \gamma D_k < D_{k+1}.$$

Therefore, there exists some time n_{k+1} such that

$$-D_{k+1} \le Q'_n(\boldsymbol{s},\boldsymbol{a}) \le \min\{D_{k+1}, Q_n^{\lambda^*}(\boldsymbol{s},\boldsymbol{a})\} \ \ \forall\,(\boldsymbol{s},\boldsymbol{a}),\, n \ge n_{k+1},$$

which completes the induction.

□

B Weakly Coupled Q-learning Algorithm Description

Algorithm 2 Weakly Coupled Q-learning

```
1:  Input: Lagrange multiplier set $\Lambda$, initial state distribution $S_0$.
2:  Initialize Q-table estimates $Q_0$, $\{Q_{0,i}\}_{i=1}^N$. Set $Q'_0 = Q_0$.
3:  for $n = 0, 1, 2, \ldots$ do
4:      Take an $\epsilon$-greedy behavioral action $\boldsymbol{a}_n$ with respect to $Q'_n(\boldsymbol{s}_n, \boldsymbol{a})$.
5:      // Estimate upper bound by combining subagents
6:      for $i = 1, 2, \ldots, N$ do
7:          Update each subproblem value functions $Q_{i,n+1}^{\lambda}$ according to (7).
8:      end for
9:      Update right-hand-side estimate $\boldsymbol{B}_{n+1}(w_n)$ according to (8).
10:     Using (9) and (10), combine subproblems to obtain $Q_{n+1}^{\lambda^*}(\boldsymbol{s}_n, \boldsymbol{a})$ for all $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{s}_n)$.
11:     // Main agent standard update, followed by projection
12:     Do standard Q-learning update using (11) to obtain $Q_{n+1}$.
13:     Perform upper bound projection step: $Q'_{n+1}(\boldsymbol{s},\boldsymbol{a}) = Q_{n+1}^{\lambda^*}(\boldsymbol{s},\boldsymbol{a}) \wedge Q_{n+1}(\boldsymbol{s},\boldsymbol{a})$
14: end for
```

C Weakly Coupled DQN Algorithm Implementation

In our implementation of WCDQN, the subproblem Q_i^{λ}-network in Algorithm 1 follows the standard network architecture as in [43], where given an input state s_i the network predicts the Q-values for all actions. This mandates that all the subproblems have the same number of actions. To address different subproblem action spaces, we can change the network architecture to receive the state-action pair (s_i, a_i) as input and output the predicted Q-value. This simple change does not interfere or affect WCDQN's main idea.

Our code is available at `https://github.com/ibrahim-elshar/WCDQN_NeurIPS`.

D Numerical Experiment Details

A discount factor of 0.9 is used for the EV charging problem and 0.99 for the multi-product inventory and online stochastic ad matching problems. In the tabular setting, we use a polynomial learning rate that depends on the state-action pairs visitation given by $\alpha_n(\boldsymbol{s},\boldsymbol{a}) = 1/\nu_n(\boldsymbol{s},\boldsymbol{a})^r$, where $\nu_n(\boldsymbol{s},\boldsymbol{a})$ represent the number of times $(\boldsymbol{s},\boldsymbol{a})$ has been visited up to iteration n, and $r = 0.4$. We also use an ϵ-greedy exploration policy, given by $\epsilon(s) = 1/\nu(\boldsymbol{s})^e$, where $\nu(\boldsymbol{s})$ is the number of times the state $\boldsymbol{s}$ has been visited. We set $e = 0.4$. In the function approximation setting, we use an ϵ-greedy policy that decays ϵ from 1 to 0.05 after $30{,}000$ steps. All state-action value functions are initialized randomly. Experiments were ran on a shared memory cluster with dual 12-core Skylake CPU (Intel Xeon Gold 6126 2.60 GHz) and 192 GB RAM/node.

D.1 EV charging deadline scheduling [63]

In this problem, there are in total three charging spots $N = 3$. Each spot represents a subproblem with state $(c_t, B_{t,i}, D_{t,i})$, where $c_t \in \{0.2, 0.5, 0.8\}$ is the exogenous electric cost, $B_{t,i} \le 2$ is the amount of charge required and $D_{t,i} \leq 4$ is the remaining time until the EV leaves the system. The state space size is 36 for each subproblem. At a given period t, the action of each subproblem is whether to charge an EV occupying the charging spot $a_{t,i} = 1$ or not $a_{t,i} = 0$. A feasible action is given by $\sum_{i=1}^{N} a_{t,i} \leq b(c_t)$, where $b(0.2) = 3, b(0.5) = 2$, and $b(0.8) = 1$. The reward of each subproblem is given by

$$r_i\big((c_t, B_{t,i}, D_{t,i}), a_{t,i}\big) = \begin{cases} (1 - c_t)\, a_{t,i} \quad \text{if } B_{t,i} > 0, D_{t,i} > 1, \\ (1 - c_t)\, a_{t,i} - F(B_{t,i} - a_{t,i}) \text{ if } B_{t,i} > 0, D_{t,i} = 1, \\ 0, \quad \text{otherwise,} \end{cases}$$

where $F(B_{t,i} - a_{t,i}) = 0.2\,(B_{t,i} - a_{t,i})^2$ is a penalty function for failing to complete charging the EV before the deadline. The endogenous state of each subproblem evolves such that $(B_{t+1,i}, D_{t+1,i}) = (B_{t,i} - a_{t,i}, D_{t,i} - 1)$ if $D_{t,i} > 1$, and $(B_{t+1,i}, D_{t+1,i}) = (B, D)$ with probability $q(D, B)$ if $D_{t,i} \leq 1$, where $q(0, 0) = 0.3$ and $q(B, D) = 0.7/11$ for all $B > 0$ and $D > 0$. The exogenous state c_t evolves following the transition probabilities given by:

$$q(c_{t+1} \,|\, c_t) = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.5 & 0.3 \\ 0.6 & 0.2 & 0.2 \end{pmatrix}.$$

D.2 Multi-product inventory control with an exogenous production rate [27]

We consider manufacturing $N = 10$ products. The exogenous demand $D_{t,i}$ for each product $i \in \{1, 2, \ldots, 10\}$ follows a Poisson distribution with mean value μ_i. The maximum storage capacity and the maximum number of allowable backorders (after which lost sales costs incur) for product i are given by R_i and M_i, respectively.

The state for subproblem i is given by $(x_{t,i}, p_t)$, where $x_{t,i} \in X_i = \{-M_i, -M_i + 1, \ldots, R_i\}$ is the inventory level for product i, and p_t is an exogenous and Markovian noise with support $[0.8, 1]$. A negative stock level corresponds to the number of backorders. For subproblem i, the action $a_{t,i}$ is the number of resources allocated to the product i. The maximum number of resources available for all products is $U = 3$, so feasible actions must satisfy $\sum_i a_{t,i} \leq 3$.

Allocating a resource level $a_{t,i}$ yields a production rate $\rho_i(a_{t,i}, p_t) = (12\,p_t\, a_{t,i})/(5.971 + a_{t,i})$. The cost function for product i is $c_i(p_t, x_{t,i}, a_{t,i})$ and represents the sum of the holding, backorders, and lost sales costs. We let h_i, b_i, and l_i denote the per-unit holding, backorder, and lost sale costs, respectively. The cost function $c_i(x_{t,i}, p_t, a_{t,i})$ is given by,

$$\begin{aligned} c_i(x_{t,i}, p_t, a_{t,i}) = h_i(x_{t,i} + \rho_i(a_{t,i}, p_t))_+ + b_i(-x_{t,i} - \rho_i(a_{t,i}, p_t))_+ \\ + l_i((D_{t,i} - x_{t,i} - \rho_i(a_{t,i}, p_t))_+ - M_i)_+, \end{aligned}$$

where $(.)_+ = \max(., 0)$. We summarize the cost parameters and the mean demand for each product in Table 1. Finally, the transition for the inventory state of subproblem i is given by

$$x_{t+1,i} = \max\big(\min(x_{t,i} + \rho_i(a_{t,i}, p_t) - D_{t,i}, R_i), -M_i\big),$$

where the exogenous noise p_t evolves according to a transition matrix sampled from a Dirichlet distribution whose parameters are each sampled (once per replication) from a Uniform$(1, 5)$ distribution.

Table 1: Multi-product inventory environment parameters

Product i	1	2	3	4	5	6	7	8	9	10
Storage capacity R_i	20	30	10	15	10	10	25	30	15	10
Maximum backorders M_i	5	5	5	5	5	5	5	5	5	5
Mean demand μ_i	0.3	0.7	0.5	1.0	1.4	0.9	1.1	1.2	0.3	0.6
Holding cost h_i	0.1	0.2	0.05	0.3	0.2	0.5	0.3	0.4	0.15	0.12
Backorder cost b_i	3.0	1.2	5.15	1.3	1.1	1.1	10.3	1.05	1.	3.1
Lost sales cost l_i	30.1	3.3	10.05	3.9	3.7	3.6	40.3	4.5	12.55	44.1

D.3 Online stochastic ad matching [18]

In this problem, a platform needs to match $N = 6$ advertisers to arriving impressions [18]. An impression $e_t \in E = \{1, 2, \dots, 5\}$ arrives according to a discrete time Markov chain with transition probabilities given by $q(e_{t+1} \mid e_t)$, where each row of the transition matrix q is sampled from a Dirichlet distribution whose parameters are sampled (once per replication) from Uniform$(1, 20)$.

The action $a_{t,i} \in \{0, 1\}$ is whether to assign impression e_t to advertiser i or not. The platform can assign an impression to at most one advertiser: $\sum_{i=1}^{N} a_{i,t} = 1$.

The state of advertiser i, $x_{i,t}$ gives the number of remaining ads to display and evolves according to $x_{t+1,i} = x_{t,i} - a_{t,i}$. The initial state is $x_0 = (10, 11, 12, 10, 14, 9)$. The reward obtained from advertiser i in state $s_{t,i} = (x_{t,i}, e_t)$ is $r_i(s_{t,i}, a_{t,i}) = l_{i,e_t} \min(x_{t,i}, a_{t,i})$, where the parameters l_{i,e_t} are sampled (once per replication) from Uniform$(1, 4)$.

D.4 Training parameters

Each method was trained for 6,000 episodes for the EV charging problem, 5,000 for the multi-product inventory control problem, and 10,000 episodes for the online stochastic ad matching problem. The episode lengths for the EV charging, online ad stochastic ad matching, and multi-product inventory control problems are 50, 30, and 25, respectively. We performed 5 independent replications.

We use a neural network architecture that consists of two hidden layers, with 64 and 32 hidden units respectively, for all algorithms. A rectified linear unit (ReLU) is used as the activation function for each hidden layer. The Adam optimizer [36] with a learning rate of 1.0×10^{-4} was used. For OTDQN, we use the same parameter settings as in He et al. [25].

For WCDQN, we use a Lagrangian multiplier $\lambda \in [0, 10]$, with a 0.01 discretization. We also used an experience buffer of size $100{,}000$ and initialized it with $10{,}000$ experience tuples that were obtained using a random policy. For the WCDQN algorithm, we set the penalty coefficient κ_U to 10, after performing a small amount of manual hyperparameter tuning on the set $\{1, 2, 4, 10\}$.

D.5 Sensitivity analysis of WCQL with respect to the number of subproblems

We study the performance improvement from WCQL over vanilla Q-learning as the number of subproblems increases for the EV charging problem. We only vary the number of subproblems (from 2 to 5) and keep all other settings as defined in Appendix D.1. The results, given in Table 2, show that the benefits of WCQL become larger as the number of subproblems increases. This provides some additional evidence for the practicality of our approach, especially in regimes where standard methods fail.

Table 2: Cumulative reward and percent improvement of WCQL over QL on the EV-charging problem with a different number of subproblems.

Algorithm	Number of Subproblems			
	2	3	4	5
QL	5.39	6.7	5.2	3.26
WCQL	5.35	7.14	6.28	4.66
Percent improvement	-0.7%	6.6%	20.8%	42.9%

E Limitations and Future Work

One interesting direction to explore for future work is to address the limitation of learning the Lagrangian upper bound using a fixed and finite set Λ. Instead, one can imagine the ability to learn the optimal value of λ and concentrate the computational effort towards learning the Lagrangian upper bound for this particular λ, which could potentially lead to tighter bounds. A possible approach is to apply subgradient descent on λ, similar to what is done in Hawkins [24].