A Detailed Proof

A.1 Proof of Theorem 4.1

Proof. Similar to the proof of Theorem 3.2 in Kumar et al. [4], we first prove this theorem in the absence of sampling error, and then incorporate sampling error at the end. By set the derivation of the objective in Eq. 4 to zero, we can compute the Q-function update induced in the exact, tabular setting($\mathcal{T}^{\boldsymbol{\pi}} = \hat{\mathcal{T}}^{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}_\beta(\mathbf{a}|s) = \hat{\boldsymbol{\pi}}_\beta(\mathbf{a}|s)$).

$$\forall\, s, \boldsymbol{a}, k,\ \hat{Q}^{k+1}(s,\boldsymbol{a}) = \mathcal{T}^{\boldsymbol{\pi}} \hat{Q}^k(s,\boldsymbol{a}) - \alpha \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i}{\pi_\beta^i} - 1 \right] \tag{A.1}$$

Then, the value of the policy, $\hat{V}^{k+1}$ can be proved to be underestimated, since:

$$\hat{V}^{k+1}(s) = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}(\boldsymbol{a}|s)} \left[\hat{Q}^{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right] = \mathcal{T}^{\boldsymbol{\pi}} \hat{V}^k(s) - \alpha \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}(\boldsymbol{a}|s)} \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i}{\pi_\beta^i} - 1 \right] \tag{A.2}$$

Next, we will show that $D_{CQL}^{CF}(s) = \Sigma_a \boldsymbol{\pi}(\boldsymbol{a}|s) \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i(a^i|s)}{\hat{\pi}_\beta^i(a^i|s)} - 1 \right]$ is always positive, when $\mu^i(a^i|s) = \pi^i(a^i|s)$:

$$\begin{aligned}
D_{CQL}^{CF}(s) &= \Sigma_a \boldsymbol{\pi}(\boldsymbol{a}|s) \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i(a^i|s)}{\pi_\beta^i(a^i|s)} - 1 \right] && \text{(A.3)} \\
&= \Sigma_{i=1}^n \lambda_i \left[\Sigma_{a^i} \pi^i(a^i|s) \left[\frac{\mu^i(a^i|s)}{\pi_\beta^i(a^i|s)} - 1 \right] \right] && \text{(A.4)} \\
&= \Sigma_{i=1}^n \lambda_i \left[\Sigma_{a^i} (\pi^i(a^i|s) - \pi_\beta^i(a^i|s) + \pi_\beta^i(a^i|s)) \left[\frac{\mu^i(a^i|s)}{\pi_\beta^i(a^i|s)} - 1 \right] \right] && \text{(A.5)} \\
&= \Sigma_{i=1}^n \lambda_i \left[\Sigma_{a^i} (\pi^i(a^i|s) - \pi_\beta^i(a^i|s)) \left[\frac{\pi^i(a^i|s) - \pi_\beta^i(a^i|s)}{\pi_\beta^i(a^i|s)} \right] + \Sigma_{a^i} \pi_\beta^i(a^i|s)) \left[\frac{\mu^i(a^i|s)}{\pi_\beta^i(a^i|s)} - 1 \right] \right] && \text{(A.6)} \\
&= \Sigma_{i=1}^n \lambda_i \left[\Sigma_{a^i} \left[\frac{(\pi^i(a^i|s) - \pi_\beta^i(a^i|s))^2}{\pi_\beta^i(a^i|s)} \right] + 0 \right] \ \textit{since}, \forall i, \Sigma_{a^i} \pi^i(a^i|s) = \Sigma_{a^i} \pi_\beta^i(a^i|s) = 1 && \text{(A.7)} \\
&\geq 0 && \text{(A.8)}
\end{aligned}$$

As shown above, the $D_{CQL}^{CF}(s) \geq 0$, and $D_{CQL}^{CF}(s) = 0$, iff $\pi^i(a^i|s) = \pi_\beta^i(a^i|s)$. This implies that each value iterate incurs some underestimation, i.e. $\hat{V}^{k+1}(s) \leq \mathcal{T}^{\boldsymbol{\pi}} \hat{V}^k(s)$.

We can compute the fixed point of the recursion in Equation A.2 and get the following estimated policy value:

$$\hat{V}^{\boldsymbol{\pi}}(s) = V^{\boldsymbol{\pi}}(s) - \alpha \left[(I - \gamma P^{\boldsymbol{\pi}})^{-1} \Sigma_a \boldsymbol{\pi}(\boldsymbol{a}|s) \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i(a^i|s)}{\hat{\pi}_\beta^i(a^i|s)} - 1 \right] \right] (s) \tag{A.9}$$

Because the $(I - \gamma P^{\boldsymbol{\pi}})^{-1}$ is non negative and the $D_{CQL}^{CF}(s) \geq 0$, it's easily to prove that in the absence of sampling error, Theorem 4.1 gives a lower bound.

Incorporating sampling error. According to the conclusion in Kumar et al. [4], we can directly write down the result with sampling error as follows:

$$\hat{V}^{\boldsymbol{\pi}}(s) \leq V^{\boldsymbol{\pi}}(s) - \alpha \left[(I - \gamma P^{\boldsymbol{\pi}})^{-1} \Sigma_a \boldsymbol{\pi}(\boldsymbol{a}|s) \left[\Sigma_{i=1}^n \lambda_i \frac{\mu^i(a^i|s)}{\hat{\pi}_\beta^i(a^i|s)} - 1 \right] \right] (s) + \left[(I - \gamma P^{\boldsymbol{\pi}})^{-1} \frac{C_{r,T,\sigma} R_{max}}{(1-\gamma)\sqrt{|D|}} \right] \tag{A.10}$$

So, the statement of Theorem 4.1 with sampling error is proved. Please refer to the Sec.D.3 in Kumar et al. [4] For detailed proof. Besides, the choice of α in this case to prevent overestimation is given by:

$$\alpha \geq \max_{s,\boldsymbol{a}\in D} \frac{C_{r,T,\sigma} R_{max}}{(1-\gamma)\sqrt{|D|}} \cdot \max_{s\in D} \left[\Sigma_a \boldsymbol{\pi}(\boldsymbol{a}|s) \left[\Sigma_{i=1}^{n} \lambda_i \frac{\mu^i(a^i|s)}{\hat{\pi}_\beta^i(a^i|s)} - 1 \right] \right]^{-1} \tag{A.11}$$

□

A.2 Proof of Theorem 4.2

Proof. According to the definition, we can get the formulation of $D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s)$ and $D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s)$ as follow:

$$D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s) = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}(\cdot|s)} \left(\left[\sum_{i=1}^{n} \lambda_i \frac{\pi^i(a^i|s)}{\beta^i(a^i|s)} \right] - 1 \right) \tag{A.12}$$

$$= \sum_{i=1}^{n} \lambda_i \left(\sum_{a^i} \frac{\pi^i(a^i|s) * \pi^i(a^i|s)}{\beta^i(a^i|s)} \right) - 1 \geq 0 \tag{A.13}$$

$$D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s) = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}(\cdot|s)} \left(\left[\frac{\boldsymbol{\pi}(\boldsymbol{a}|s)}{\boldsymbol{\beta}(\boldsymbol{a}|s)} \right] - 1 \right) \tag{A.14}$$

$$= \prod_{i=1}^{n} \left(\sum_{a^i} \frac{\pi^i(a^i|s) * \pi^i(a^i|s)}{\beta^i(a^i|s)} \right) - 1 \geq 0 \tag{A.15}$$

Then, by taking the logarithm of $D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s)$, we get:

$$\ln(D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s) + 1) = \sum_{i=1}^{n} \ln \left(\mathbb{E}_{a^i\sim\pi^i(\cdot|s)} \frac{\pi^i(a^i|s)}{\beta^i(a^i|s)} \right) \tag{A.16}$$

As $\Sigma_i \lambda_i = 1$, it's obvious that

$$\ln(D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s) + 1) \leq \ln \left(\sum_{a^j} \frac{\pi^j(a^j|s) * \pi^j(a^j|s)}{\beta^j(a^j|s)} \right), where\ j = \arg\max_k \mathbb{E}_{\pi^k} \frac{\pi^k}{\beta^k} \tag{A.17}$$

By combining equation A.16 and inequation A.17, we get

$$\frac{D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s) + 1}{D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s) + 1} \geq \exp \left(\sum_{i=1, i\neq j}^{n} \ln \left(\mathbb{E}_{a^i\sim\pi^i(\cdot|s)} \frac{\pi^i(a^i|s)}{\beta^i(a^i|s)} \right) \right) \tag{A.18}$$

$$\geq \exp \left(\sum_{i=1, i\neq j}^{n} KL(\pi^i(s)||\beta^i(s)) \right), where\ j = \arg\max_k \mathbb{E}_{\pi^k} \frac{\pi^k}{\beta^k} \tag{A.19}$$

the second inequality is derived from the Jensen's inequality. As the Kullback-Leibler Divergence is non-negative, it's obvious that $D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s) \geq D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s)$, then we can simplify the left-hand side of this inequality:

$$\frac{D_{CQL}(\boldsymbol{\pi},\boldsymbol{\beta})(s)}{D_{CQL}^{CF}(\boldsymbol{\pi},\boldsymbol{\beta})(s)} \geq \exp \left(\sum_{i=1, i\neq j}^{n} KL(\pi^i(s)||\beta^i(s)) \right), where\ j = \arg\max_k \mathbb{E}_{\pi^k} \frac{\pi^k}{\beta^k} \tag{A.20}$$

□

A.3 Proof of Equation 6

Proof. Similar to the proof of Lemma D.3.1 in CQL [4], $\bar{Q}$ is obtained by solving a recursive Bellman fixed point equation in the empirical MDP $\hat{M}$, with an altered reward, $r(s,a)-\alpha\left[\sum_i \lambda_i \frac{\pi^i(a^i|s)}{\beta^i(a^i|s)} - 1\right]$, hence the optimal policy $\boldsymbol{\pi}^*(\boldsymbol{a}|s)$ obtained by optimizing the value under the CFCQL Q-function equivalently is characterized via Eq. 6. □

A.4 Proof of Theorem 4.3

Proof. Similar to Eq. 6, $\boldsymbol{\pi}^*_{MA}$ is equivalently obtained by solving:

$$\boldsymbol{\pi}^*_{MA}(\boldsymbol{a}|s) \leftarrow \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \hat{M}) - \alpha \frac{1}{1-\gamma} \mathbb{E}_{s\sim d^{\boldsymbol{\pi}}_{\hat{M}}(s)}[D_{CQL}(\boldsymbol{\pi}, \boldsymbol{\beta})(s)]. \tag{A.21}$$

Recall that $\forall s, \boldsymbol{\pi}, \boldsymbol{\beta}, D_{CQL}(\boldsymbol{\pi}, \boldsymbol{\beta})(s) \geq 0$. We have

$$\begin{aligned} J(\boldsymbol{\pi}^*_{MA}, \hat{M}) \geq & J(\boldsymbol{\pi}^*_{MA}, \hat{M}) - \alpha \frac{1}{1-\gamma} \mathbb{E}_{s\sim d^{\boldsymbol{\pi}^*_{MA}}_{\hat{M}}(s)}[D_{CQL}(\boldsymbol{\pi}^*_{MA}, \boldsymbol{\beta})(s)] \\ \geq & J(\boldsymbol{\pi}^*, \hat{M}) - \alpha \frac{1}{1-\gamma} \mathbb{E}_{s\sim d^{\boldsymbol{\pi}^*}_{\hat{M}}(s)}[D_{CQL}(\boldsymbol{\pi}^*, \boldsymbol{\beta})(s)]. \end{aligned} \tag{A.22}$$

Then we give an upper bound of $\mathbb{E}_{s\sim d^{\boldsymbol{\pi}^*}_{\hat{M}}(s)}[D_{CQL}(\boldsymbol{\pi}^*, \boldsymbol{\beta})(s)]$. Due to the assumption that β^i is greater than ϵ anywhere, we have

$$\begin{aligned} D_{CQL}(\boldsymbol{\pi}, \boldsymbol{\beta})(s) = & \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|s)[\frac{\boldsymbol{\pi}(\boldsymbol{a}|s)}{\boldsymbol{\beta}(\boldsymbol{a}|s)} - 1] = \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|s)[\frac{\boldsymbol{\pi}(\boldsymbol{a}|s)}{\prod_{i=1}^n \beta^i(a^i|s)} - 1] \\ \leq & \left(\frac{1}{\epsilon^n} \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|s)[\boldsymbol{\pi}(\boldsymbol{a}|s)]\right) - 1 \leq \frac{1}{\epsilon^n} - 1. \end{aligned} \tag{A.23}$$

Combining Eq. A.22 and Eq. A.23, we can get

$$J(\boldsymbol{\pi}^*_{MA}, \hat{M}) \geq J(\boldsymbol{\pi}^*, \hat{M}) - \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon^n} - 1) \tag{A.24}$$

Recall the sampling error proved in [4] and referred to above in (A.10), we can use it to bound the performance difference for any $\boldsymbol{\pi}$ on true and empirical MDP by

$$|J(\boldsymbol{\pi}, M) - J(\boldsymbol{\pi}, \hat{M})| \leq \frac{C_{r,T,\delta} R_{max}}{(1-\gamma)^2} \sum_s \frac{\rho(s)}{\sqrt{|D(s)|}}, \tag{A.25}$$

then let $sampling\ error := 2 \cdot \frac{C_{r,T,\sigma} R_{max}}{(1-\gamma)^2} \sum_s \frac{\rho(s)}{\sqrt{|D(s)|}}$, and incorporate it into (A.24) , we get

$$J(\boldsymbol{\pi}^*_{MA}, M) \geq J(\boldsymbol{\pi}^*, M) - \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon^n} - 1) - \text{sampling error} \tag{A.26}$$

where $sampling\ error$ is a constant dependent on the MDP itself and D. Note that during the proof we do not take advantage of the nature of $\boldsymbol{\pi}^*$. Actually $\boldsymbol{\pi}^*$ can be replaced by any policy $\boldsymbol{\pi}$. The reason we use $\boldsymbol{\pi}^*$ is that it can give that largest lower bound, resulting in the best policy improvement guarantee. Similarly, D^{CF}_{CQL} can be bounded by $\frac{1}{\epsilon} - 1$:

$$\begin{aligned} D^{CF}_{CQL}(\boldsymbol{\pi}, \boldsymbol{\beta})(s) = & \sum_{i=1}^n \lambda_i \sum_{a^i} \pi^i(a^i|s)[\frac{\pi^i(a^i|s)}{\beta^i(a^i|s)} - 1] \\ \leq & \left(\frac{1}{\epsilon} \sum_{i=1}^n \lambda_i \sum_{a^i} \pi^i(a^i|s)[\pi^i(a^i|s)]\right) - 1 \\ \leq & \frac{1}{\epsilon}\left(\sum_{i=1}^n \lambda_i\right) - 1 = \frac{1}{\epsilon} - 1. \end{aligned} \tag{A.27}$$

□

A.5 Proof of Theorem 4.4

We first show the theorem of safe policy improvement guarantee for MACQL and CFCQL, separately. Then we compare these two gaps.

MACQL has a safe policy improvement guarantee related to the number of agents n:

Theorem A.1. *Given the discounted marginal state-distribution $d^{\boldsymbol{\pi}}_{\hat{M}}$, we define $\mathcal{B}(\boldsymbol{\pi}, D) = \mathbb{E}_{s\sim d^{\boldsymbol{\pi}}_{\hat{M}}}[\sqrt{D\left(\boldsymbol{\pi}, \boldsymbol{\beta}\right)(s)+1}]$. The policy $\boldsymbol{\pi}^*_{MA}(\boldsymbol{a}|s)$ is a ζ^{MA}-safe policy improvement over $\boldsymbol{\beta}$ in the actual MDP M, i.e., $J(\boldsymbol{\pi}^*_{MA}, M) \geq J(\boldsymbol{\beta}, M) - \zeta^{MA}$, where $\zeta^{MA} = 2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{\max} C_{T,\delta}}{(1-\gamma)^2}\right) \cdot \frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}\mathcal{B}(\boldsymbol{\pi}^*_{MA}, D_{CQL}) + \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon^n} - 1) - (J(\boldsymbol{\pi}^*, \hat{M}) - J(\hat{\boldsymbol{\beta}}, \hat{M}))$.*

Proof. We can first get a $J(\boldsymbol{\pi}^*_{MA}, \hat{M})$-related policy improvement guarantee following the proof of Theorem 3.6 in Kumar et al. [4]:

$$\begin{aligned} J(\boldsymbol{\pi}^*_{MA}, M) \geq & J(\boldsymbol{\beta}, M) - \left(2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{\max} C_{T,\delta}}{(1-\gamma)^2}\right) \cdot \frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}\mathcal{B}(\boldsymbol{\pi}^*_{MA}, D_{CQL}) \right. \\ & \left. - (J(\boldsymbol{\pi}^*_{MA}, \hat{M}) - J(\hat{\boldsymbol{\beta}}, \hat{M}))\right) \end{aligned} \tag{A.28}$$

According to Eq. A.21, $\boldsymbol{\pi}^*_{MA}$ is obtained by optimizing $J(\boldsymbol{\pi}, \hat{M})$ with a D_{CQL}-related regularizer. And Theorem 4.3 shows that D_{CQL} can be extremely large when the team size expands, which may severely change the optimization objective and affects the shape of the optimization plane. Therefore, $J(\boldsymbol{\pi}^*_{MA}, \hat{M})$ may be extremely low, and keeping $J(\boldsymbol{\pi}^*_{MA}, \hat{M})$ in Eq. A.28 results in a mediocre policy improvement guarantee. To bound $J(\boldsymbol{\pi}^*_{MA}, \hat{M})$, we introduce Eq. A.24 into Eq. A.28, we get the following:

$$\begin{aligned} J(\boldsymbol{\pi}^*_{MA}, M) \geq & J(\boldsymbol{\beta}, M) - \left(2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{\max} C_{T,\delta}}{(1-\gamma)^2}\right) \cdot \frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}\mathcal{B}(\boldsymbol{\pi}^*_{MA}, D_{CQL}) \right. \\ & \left. + \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon^n} - 1) - (J(\boldsymbol{\pi}^*, \hat{M}) - J(\hat{\boldsymbol{\beta}}, \hat{M}))\right) \end{aligned} \tag{A.29}$$

This complete the proof. □

We can get a similar ζ^{CF} satisfying $J(\boldsymbol{\pi}^*_{CF}, M) \geq J(\boldsymbol{\beta}, M) - \zeta^{CF}$ for CFCQL, which is independent of n:

$$\zeta^{CF} = 2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{\max} C_{T,\delta}}{(1-\gamma)^2}\right) \cdot \frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}\mathcal{B}(\boldsymbol{\pi}^*_{CF}, D^{CF}_{CQL}) + \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon} - 1) - (J(\boldsymbol{\pi}^*, \hat{M}) - J(\hat{\boldsymbol{\beta}}, \hat{M})) \tag{A.30}$$

Then we can prove Theorem 4.4.

Proof. Subtract ζ^{CF} from ζ^{MA}, and we get:

$$\zeta^{MA} - \zeta^{CF} = 2\left(\frac{C_{r,\delta}}{1-\gamma} + \frac{\gamma R_{\max} C_{T,\delta}}{(1-\gamma)^2}\right) \frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}} \left(\mathcal{B}(\boldsymbol{\pi}^*_{MA}, D_{CQL}) - \mathcal{B}(\boldsymbol{\pi}^*_{CF}, D^{CF}_{CQL})\right) + \frac{\alpha}{1-\gamma}(\frac{1}{\epsilon^n} - \frac{1}{\epsilon}) \tag{A.31}$$

Let the right side ≥ 0, and we can get

$$n \geq \log_{\frac{1}{\epsilon}}\left[\max\left(1, \frac{1}{\epsilon} + \frac{2}{\alpha}\frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}\left(C_{r,\delta} + \frac{\gamma R_{\max} C_{T,\delta}}{1-\gamma}\right) \cdot \left[\mathcal{B}\left(\boldsymbol{\pi}^*_{CF}, D^{CF}_{CQL}\right) - \mathcal{B}\left(\boldsymbol{\pi}^*_{MA}, D_{CQL}\right)\right]\right)\right] \tag{A.32}$$

According to Theorem 4.3,

$$\mathcal{B}\left(\boldsymbol{\pi}^*_{CF}, D^{CF}_{CQL}\right) = \mathbb{E}_{s\sim d^{\boldsymbol{\pi}^*_{CF}}_{\hat{M}}}[\sqrt{D^{CF}_{CQL}(\boldsymbol{\pi}^*_{CF}, \boldsymbol{\beta})(s)+1}] \leq \mathbb{E}_{s\sim d^{\boldsymbol{\pi}^*_{CF}}_{\hat{M}}}[\sqrt{\frac{1}{\epsilon} - 1 + 1}] = \frac{1}{\sqrt{\epsilon}} \tag{A.33}$$

In the meantime, we have

$$\mathcal{B}\left(\boldsymbol{\pi}_{CF}^{*}, D_{CQL}^{CF}\right) = \mathbb{E}_{s\sim d_{\hat{M}}^{\boldsymbol{\pi}_{MA}^{*}}}[\sqrt{D_{CQL}(\boldsymbol{\pi}_{MA}^{*}, \boldsymbol{\beta})(s)+1}] \geq \mathbb{E}_{s\sim d_{\hat{M}}^{\boldsymbol{\pi}_{MA}^{*}}}[\sqrt{D_{CQL}(\boldsymbol{\beta}, \boldsymbol{\beta})(s)+1}] = 1 \tag{A.34}$$

Therefore, we can relax the lower bound of n to a constant that

$$n \geq \log_{\frac{1}{\epsilon}}\left(\frac{1}{\epsilon} + \frac{2}{\alpha}\frac{\sqrt{|A|}}{\sqrt{|\mathcal{D}(s)|}}(C_{r,\delta} + \frac{\gamma R_{\max} C_{T,\delta}}{1-\gamma})\cdot(\frac{1}{\sqrt{\epsilon}}-1)\right) \tag{A.35}$$

□

B Implement Details

B.1 Derivation of the Update Rule

To utilize the Eq. 4 for policy optimization, following the analysis in the Section 3.2 in Kumar et al. [4], we formally define optimization problems over each $\mu^i(a^i|s)$ by adding a regularizer $R(\mu^i)$. As shown below, we mark the modifications from the Eq. 4 in red.

$$\begin{aligned}\min_{Q}\max_{\boldsymbol{\mu}} \alpha\left[\sum_{i=1}^{n}\lambda_i\mathbb{E}_{s\sim\mathcal{D},a^i\sim\mu^i,\boldsymbol{a}^{-i}\sim\boldsymbol{\beta}^{-i}}[Q(s,\boldsymbol{a})] - \mathbb{E}_{s\sim\mathcal{D},\boldsymbol{a}\sim\boldsymbol{\beta}}[Q(s,\boldsymbol{a})]\right] \\ + \frac{1}{2}\mathbb{E}_{s,\boldsymbol{a},s'\sim\mathcal{D}}\left[(Q(s,\boldsymbol{a}) - \hat{\mathcal{T}}^{\boldsymbol{\pi}}\bar{Q}_k(s,\boldsymbol{a}))^2\right] + \sum_{i=1}^{n}\lambda_i R(\mu^i),\end{aligned} \tag{B.36}$$

By choosing different regularizer, there are a variety of instances within CQL family. As recommended in Kumar et al. [4], we choose $R(\mu^i)$ to be the KL-divergence against a Uniform distribution over action space, i.e., $R(\mu^i) = -D_{KL}(\mu^i, Unif(a^i))$, then it's easily to derive the following variant of Eq. B.36 called CFCQL(H) which is the update rule we used:

$$\begin{aligned}\min_{Q}\ \alpha\mathbb{E}_{s\sim\mathcal{D}}\left[\sum_{i=1}^{n}\lambda_i\mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\beta}^{-i}}[\log\sum_{a^i}\exp(Q(s,\boldsymbol{a}))] - \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\beta}}[Q(s,\boldsymbol{a})]\right] \\ + \frac{1}{2}\mathbb{E}_{s,\boldsymbol{a},s'\sim\mathcal{D}}\left[(Q(s,\boldsymbol{a}) - \hat{\mathcal{T}}^{\boldsymbol{\pi}_k}\bar{Q}_k(s,\boldsymbol{a}))^2\right].\end{aligned} \tag{B.37}$$

B.2 Details for Computing λ

To compute λ, we need an explicit expression of π^i and β^i. In the setting of discrete action space, as we use Q-learning, π^i can be expressed by the Boltzman policy, i.e.

$$\pi^i(a_j^i) = \frac{\exp\left(\mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\beta}^{-i}}Q(s,a_j^i,\boldsymbol{a^{-i}})\right)}{\sum_k \exp\left(\mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\beta}^{-i}}Q(s,a_k^i,\boldsymbol{a^{-i}})\right)} \tag{B.38}$$

We use behaviour cloning to pre-train a parameterized $\boldsymbol{\beta}(s)$ with a three-level fully-connected network and MLE(Maximum Likelihood Estimation) loss.

With the explicit expression of π^i and β^i, we can directly compute λ with Eq. 8 and Eq. 9. While, in practice, we find the $\mathbb{E}_{\pi^i}\frac{\pi^i(s)}{\beta^i(s)}$ may introduce extreme variance as its large scale and fluctuations, which will hurt the performance. Instead, we take the logarithm of it and further reduced it to the Kullback-Leibler Divergence as follow:

$$\forall i,s,\lambda_i(s) = \frac{\exp\left(-\tau D_{KL}(\pi^i(s)||\beta^i(s))\right)}{\sum_{j=1}^{n}\exp\left(-\tau D_{KL}(\pi^j(s)||\beta^j(s))\right)}, \tag{B.39}$$

For continuous action space, we use the deterministic policy like in MADDPG, whose policy distribution can be regared as a Dirac delta function. Therefore, we approximate $\mathbb{E}_{\pi^j} \frac{\pi^j(s)}{\beta^j(s)}$ by the following:

$$\mathbb{E}_{\pi^j} \frac{\pi^j(s)}{\beta^j(s)} \approx \frac{1}{\beta^j(\pi^j(s)|s)} \tag{B.40}$$

Then we need to obtain an explicit expression of β^i. We first train a VAE [3] from the dataset to obtain the lower bound of β^i. Let $p_\phi(a, z|s)$ and $q_\varphi(z|a, s)$ be the decoder and the encoder of the trained VAE, respectively. According to Wu et al. [13], $\beta^j(a^j|s)$ can be explicitly estimated by (We omit the superscript j for brevity):

$$\begin{aligned} \log \beta_\phi(a \mid s) &= \log \mathbb{E}_{q_\varphi(z|a,s)} \left[\frac{p_\phi(a, z \mid s)}{q_\varphi(z \mid a, s)} \right] \\ &\approx \mathbb{E}_{z^{(l)} q_\varphi(z|a,s)} \left[\log \frac{1}{L} \sum_{l=1}^{L} \frac{p_\phi\left(a, z^{(l)} \mid s\right)}{q_\varphi\left(z^{(l)} \mid a, s\right)} \right] \\ &\stackrel{\text{def}}{=} \widehat{\log \pi_\beta}(a \mid s; \varphi, \phi, L). \end{aligned} \tag{B.41}$$

Therefore, we can sample from the VAE L times to estimate β^i. The sampling error reduces as L increases.

C Experimental Details

C.1 Tasks

$Equal\_Line$ is a multi-agent task which we design by simplify the space shape of $Equal\_Space$ to one-dimension. There are n agents and they are randomly initialized to the interval $[0, 2]$. The state space is a a one-dimensional bounded region in $[0, \max(10, 2 * n)]$ and the local action space is a discrete, eleven-dimensional space, i.e. $[0, -0.01, -0.05, -0.1, -0.5, -1, 0.01, 0.05, 0.1, 0.5, 1]$, which represents the moving direction and distance at each step. The reward is shared by the agents and formulated as $10 * (n-1) \frac{min\_dis - last\_step\_min\_dis}{line\_length}$, which will spur the agents to cooperate to spread out and keep the same distance between each other.

For Multi-agent Particle Environment and Multi-agent Mujoco, we adopt the open-source implementations from Lowe et al. [5][1] and Peng et al. [8][2] respectively. And we use the datasets and the adversary agents provided by Pan et al. [7].

For StarCraft II Micromanagement Benchmark, we use the open-source implementation from Samvelyan et al. [10][3] and choose four maps with different difficulty and number of agents as the experimental scenarios, which is summarized in Table 1. We construct our own datasets with QMIX [9] by collecting training or evaluating data.

Table 1: The details of tested maps in the StarCraft II micromanagement benchmark

Maps	**Agents**	**Enemies**	**Difficulty**
2s3z	2 Stalkers & 3 Zealots	2 Stalkers & 3 Zealots	Easy
3s_vs_5z	3 Stalkers	5 Zealots	Easy
5m_vs_6m	5 Marines	6 Marines	Hard
6h_vs_8z	6 Hydralisks	8 Zealots	Super Hard

[1] `https://github.com/openai/multiagent-particle-envs`
[2] `https://github.com/schroederdewitt/multiagent_mujoco`
[3] `https://github.com/oxwhirl/smac`

C.2 StarCraft II datasets collection

The datasets are made based on the training process or trained model of QMIX[9]. Specially, the $Medium$ or $Expert$ datasets are sampled by executing a partially-pretrained policy with a medium performance level or a fully-pretrained policy. The $Medium-Replay$ datasets are exactly the replay buffer during training until the policy reaches the medium performance. The $Mixed$ datasets are the equal mixture of $Medium$ and $Expert$ datasets. All datasets contain five thousand trajectories, except for the $Medium - Replay$.

C.3 Baselines

BC: behavior cloning. In discrete action space, we train a three-level MLP network with MLE loss. In continuous action space, we use the method of explicit estimation of behavior density in Wu et al. [13], which is modified from a VAE [3] estimator. **TD3-BC**[1]: One of the SOTA single agent offline algorithm, simply adding the BC term to TD3 [2]. We use the open-source implementation[4] and modify it to a CTDE version with centralised critic. **MACQL**:naive extension of conservative Q-learning, as proposed in Sec. 3.3. We implement it based on the open-source implementation[5]. As the joint action space is enormous, we sample N actions for the logsumexp operation. **MAICQ**[14]:multi-agent version of implicit constraint Q-learning by propose the decomposed multi-agent joint-policy under implicit constraint. We use the open-source implementation[6] in discrete action space and cite the experimental results in continuous action space from Pan et al. [7]. **OMAR**[7]:uses zeroth-order optimization for better coordination among agents' policies, based on independent CQL (**ICQL**). We cite the experimental results in continuous action space from Pan et al. [7] and implement a version in discrete action space based on the open-source implementation[7]. **MADTKD**[12]:uses decision transformer to represent each agent's policy and trains with knowledge distillation. As lack of open-source implementation, We implement it based on the open-source implementation[8] of another Decision Transformer based method **MADT**[6].

C.4 Resources

We use 2 servers to run all the experiments. Each one has 8*NVIDIA RTX 3090 GPUs, and 2*AMD 7H12 CPUs. Each setting is repeated for 5 seeds. For one seed in SC2, it takes about 1.5 hours. For MPE, 10 minutes is enough. The experiments on MaMuJoCo cost the most, about 5 hours for each seed.

C.5 Code, Hyper-parameters and Reproducibility

Please refer to our submitted anonymous repository[9] for the code and the hyper-parameters of our method. For each dataset number 0, 1, 2, 3, 4, we use the seed 0, 1, 2, 3, 4, respectively.

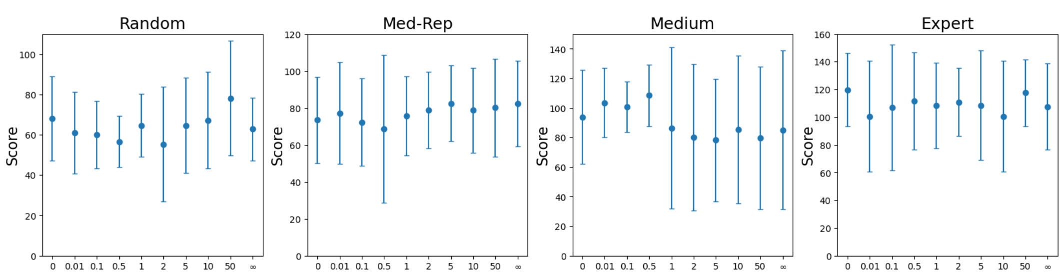

Figure 1: Ablations of τ on World.

[4]https://github.com/sfujim/TD3_BC
[5]https://github.com/aviralkumar2907/CQL
[6]https://github.com/YiqinYang/ICQ
[7]https://github.com/ling-pan/OMAR
[8]https://github.com/ReinholdM/Offline-Pre-trained-Multi-Agent-Decision-Transformer
[9]https://anonymous.4open.science/r/CFCQL-7272

Table 2: Complete results on Multi-agent Particle Environment.

Env	Dataset	MAICQ	MATD3-BC	ICQL	OMAR	MACQL	CFCQL
CN	Random	6.3±3.5	9.8±4.9	24.0±9.8	34.4±5.3	45.6±8.7	**62.2±8.1**
	Medium-replay	13.6±5.7	15.4±5.6	20.0±8.4	37.9±12.3	25.5±5.9	**52.2±9.6**
	Medium	29.3±5.5	29.3±4.8	34.1±7.2	47.9±18.9	14.3±20.2	**65.0±10.2**
	Expert	104.0±3.4	108.3±3.3	98.2±5.2	**114.9±2.6**	12.2±31	112±4
PP	Random	2.2±2.6	5.7±3.5	5.0±8.2	11.1±2.8	25.2±11.5	**78.5±15.6**
	Medium-replay	34.5±27.8	28.7±20.9	24.8±17.3	47.1±15.3	11.9±9.2	**71.1±6**
	Medium	63.3±20.0	65.1±29.5	61.7±23.1	66.7±23.2	55±43.2	**68.5±21.8**
	Expert	113.0±14.4	115.2±12.5	93.9±14.0	116.2±19.8	108.4±21.5	**118.2±13.1**
World	Random	1.0±3.2	2.8±5.5	0.6±2.0	5.9±5.2	11.7±11	**68±20.8**
	Medium-replay	12.0±9.1	17.4±8.1	29.6±13.8	42.9±19.5	13.2±16.2	**73.4±23.2**
	Medium	71.9±20.0	73.4±9.3	58.6±11.2	74.6±11.5	67.4±48.4	**93.8±31.8**
	Expert	109.5±22.8	110.3±21.3	71.9±28.1	110.4±25.7	99.7±31	**119.7±26.4**

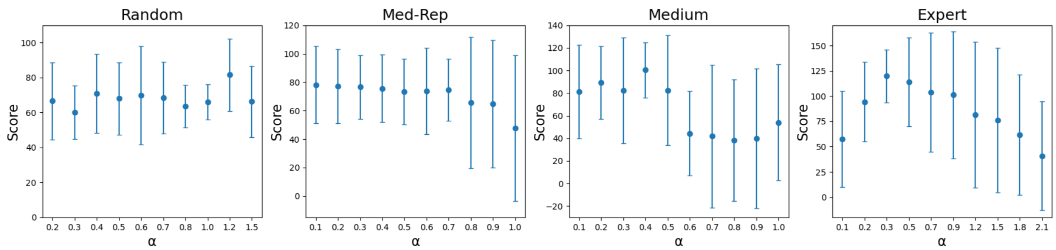

Figure 2: Ablations of α on World.

D More results

D.1 Complete Results on MPE

Table 2 shows the complete results of our methods and more baselines on Multi-agent Particle Environment. Some results are cited from Pan et al. [7].

D.2 Temperature Coefficient in Continuous Action Space

We carry out ablations of τ on MPE's map World in Fig. 1. We find that although. the best τ differs in different datasets, the overall performance is not sensitive to τ, which verifies the theoretical analysis that any simplex of λ that $\sum_{i=1}^{n} \lambda_i = 1$ can induce an underestimated value function.

D.3 Ablation on CQL α

We carry out ablations of α on MPE's map World in Fig. 2. We find that α plays a more important role for team performance on narrow distributions (e.g., $Expert$ and $Medium$) than that on wide distributions (e.g., $Random$ and $Medium-Replay$).

D.4 Component Analysis on Counterfactual style

In the environment MaMuJo, except for the counterfactual Q function, we also analyze whether the conuterfactual treatment in CFCQL can be incorporated in other components and help further improvement in Table 3. We find that the counterfactual policy improvement is critical for this environment. With CF_P, the method shows great performance gain on narrow data distribution, e.g., the $Expert$ dataset.

Table 3: Component Analysis on MaMuJoCo. CF_T: computing target Q by $\mathbb{E}_{i\sim\text{Unif}(1,n)}\mathbb{E}_{s',\boldsymbol{a}^{-i}\sim\mathcal{D},a^i\sim\pi^i}Q_{\hat{\theta}}(s,\boldsymbol{a})$. CF_P: the policy improvement (PI) by Eq. 10, otherwise using MADDPG's PI.

Dataset	Default	+CF_T	-CF_P	MACQL
Random	39.7±4.0	**48.7±1.8**	23.9±9.2	5.3±0.5
Med-Rep	**59.5±8.2**	58.9±9.6	43.5±5.6	36.7±7.1
Medium	**80.5±9.6**	76.2±12.1	43.8±7.8	51.5±26.7
Expert	**118.5±4.9**	118.1±6.9	3.7±3.1	50.1±20.1

E Discussions

E.1 Broader Impacts

Our proposed method holds potential for application in real-world multi-agent systems, such as intelligent warehouse management or medical treatment. However, directly implementing the derived policy might entail risks due to the domain gap between the training virtual datasets and real-world scenarios. To mitigate potential hazards, it is crucial for practitioners to operate the policy under human supervision, ensuring that undesirable outcomes are avoided by limiting the available options.

E.2 Limitations

Here we discuss some limitations about CFCQL. In the case of discrete action space, since CFCQL uses QMIX as the backbone, it inherits the Individual-global-max principle [11], which means it cannot solve tasks that are not factorizable. On continuous action space, the counterfactual policy update used in CFCQL allows for updating only one agent's policy for each sample, which may lead to lower convergence speed compared to methods with independent learning.

References

[1] Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[2] Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

[3] Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[5] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[6] Meng, L., Wen, M., Yang, Y., Le, C., Li, X., Zhang, W., Wen, Y., Zhang, H., Wang, J., and Xu, B. Offline pre-trained multi-agent decision transformer: One big sequence model conquers all starcraftii tasks. *arXiv preprint arXiv:2112.02845*, 2021.

[7] Pan, L., Huang, L., Ma, T., and Xu, H. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International Conference on Machine Learning*, pp. 17221–17237. PMLR, 2022.

[8] Peng, B., Rashid, T., Schroeder de Witt, C., Kamienny, P.-A., Torr, P., Böhmer, W., and Whiteson, S. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

[9] Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

[10] Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[11] Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

[12] Tseng, W.-C., Wang, T.-H., Lin, Y.-C., and Isola, P. Offline multi-agent reinforcement learning with knowledge distillation. In *Advances in Neural Information Processing Systems*, 2022.

[13] Wu, J., Wu, H., Qiu, Z., Wang, J., and Long, M. Supported policy optimization for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[14] Yang, Y., Ma, X., Li, C., Zheng, Z., Zhang, Q., Huang, G., Yang, J., and Zhao, Q. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021.