

APPENDIX

Our appendix includes the following:

- Proofs of the theoretical claims presented in the main paper.
- Details of our experimental settings.
- Detailed numerical results from the ablation study investigating the impact of α on ComaDICE’s performance.
- An ablation study assessing ComaDICE’s performance with different forms of f-divergence functions.
- An ablation study comparing ComaDICE’s performance using 1-layer versus 2-layer mixing networks.

CONTENTS

A Missing Proofs	17
A.1 Proof of Proposition 4.1	17
A.2 Proof of Theorem 4.2	18
A.3 Proof of Proposition 4.3	19
B Additional Details	21
B.1 Factorization Aspect of the Learning Objective in ComaDICE	21
B.2 Offline Multi-Agent Datasets	22
B.3 Implementation Details	22
B.4 Additional Experimental Details	24
B.5 Ablation Study: Different Values of Alpha	26
B.6 Ablation Study: Different Forms of f-divergence	28
B.7 Ablation Study: Different Types of Mixer Network	33
B.8 ComaDICE on the Penalty XOR Game	36

A MISSING PROOFS

A.1 PROOF OF PROPOSITION 4.1

Proposition. *The minimax problem in 6 is equivalent to $\min_{\nu^{tot}} \left\{ \tilde{\mathcal{L}}(\nu^{tot}) \right\}$, where*

$$\tilde{\mathcal{L}}(\nu^{tot}) = (1 - \gamma) \mathbb{E}_{s \sim p_0} [\nu^{tot}(s)] + \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} \left[\alpha f^* \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right) \right].$$

where f^* is convex conjugate of f , i.e., $f^*(y) = \sup_{t \geq 0} \{ty - f(t)\}$. Moreover, if ν^{tot} is parameterized by θ , the first order derivative of $\tilde{\mathcal{L}}(\nu^{tot})$ w.r.t. θ is given as

$$\nabla_\theta \tilde{\mathcal{L}}(\nu^{tot}) = (1 - \gamma) \mathbb{E}_{s \sim p_0} [\nabla_\theta \nu^{tot}(s)] + \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} [\nabla_\theta A_\nu^{tot}(s, a) w_\nu^{tot*}(s, a)].$$

where $w_\nu^{tot*}(s, a) = \max\{0, f'^{-1}(A_\nu^{tot}(s, a)/\alpha)\}$, where $f'^{-1}(\cdot)$ is the inverse function of the first-order derivative of f .

Proof. The first part of the proof, concerning the closed-form formulation for $\tilde{\mathcal{L}}(\nu^{tot})$, follows directly from the single-agent OptDICE paper (Lee et al., 2021). While straightforward, we include it here for the sake of completeness. Our novelty begins with the derivation of the formulation for the first-order derivative of the loss function, $\nabla_\theta \tilde{\mathcal{L}}(\nu^{tot})$.

We write the Lagrange dual function as:

$$\begin{aligned} \mathcal{L}(\nu^{tot}, \rho^{\pi_{tot}}) &= \mathbb{E}_{(s, a) \sim \rho^{\pi_{tot}}} [r(s, a)] - \alpha \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} \left[f \left(\frac{\rho^{\pi_{tot}}(s, a)}{\rho^{\mu_{tot}}(s, a)} \right) \right] \\ &\quad - \sum_s \nu^{tot}(s) \left(\sum_{a'} \rho^{\pi_{tot}}(s, a') - (1 - \gamma)p_0(s) - \gamma \sum_{a', s'} \rho^{\pi_{tot}}(s', a') P(s|a', s') \right) \\ &= \sum_s \nu^{tot}(s) (1 - \gamma)p_0(s) - \alpha \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} \left[f \left(\frac{\rho^{\pi_{tot}}(s, a)}{\rho^{\mu_{tot}}(s, a)} \right) \right] \\ &\quad + \sum_{s, a} \rho^{\mu_{tot}}(s, a) (r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \nu^{tot}(s') - \nu^{tot}(s)) \\ &= (1 - \gamma) \mathbb{E}_{s \sim p_0} [\nu^{tot}(s)] + \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} [-\alpha f(w_\nu^{tot}(s, a)) + w_\nu^{tot}(s, a) A_\nu^{tot}(s, a)], \end{aligned} \quad (13)$$

where $w_\nu^{tot}(s, a) = \frac{\rho^{\pi_{tot}}(s, a)}{\rho^{\mu_{tot}}(s, a)}$. We now see that, for each (s, a) , each component $-\alpha f(w_\nu^{tot}(s, a)) + w_\nu^{tot}(s, a) A_\nu^{tot}(s, a)$ is maximized at:

$$\max_{w^{tot} \geq 0} -\alpha f(w_\nu^{tot}(s, a)) + w_\nu^{tot}(s, a) A_\nu^{tot}(s, a) = f^* \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right),$$

where f^* is the (variant) convex conjugate of the convex function f . We then obtain:

$$\max_{w^{tot} \geq 0} \mathcal{L}(\nu^{tot}, w^{tot}) = \tilde{\mathcal{L}}(\nu^{tot}) = (1 - \gamma) \mathbb{E}_{s \sim p_0} [\nu^{tot}(s)] + \mathbb{E}_{(s, a) \sim \rho^{\mu_{tot}}} \left[\alpha f^* \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right) \right].$$

Moreover, consider the maximization problem $\max_{w^{tot} \geq 0} T(w^{tot}(s, a)) = -\alpha f(w_\nu^{tot}(s, a)) + w_\nu^{tot}(s, a) A_\nu^{tot}(s, a)$. Taking its first-order derivative w.r.t $w^{tot}(s, a)$ yields:

$$-\alpha f'(w^{tot}(s, a)) + A_\nu^{tot}(s, a).$$

So, if $f'^{-1} \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right) \geq 0$, then $w^{tot*}(s, a) = f'^{-1} \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right) \geq 0$ is optimal for the maximization problem. Otherwise, if $f'^{-1} \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right) < 0$, we see that $T(w^{tot}(s, a))$ is increasing when $w^{tot}(s, a) \leq f'^{-1} \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right)$ and decreasing when $w^{tot}(s, a) \geq f'^{-1} \left(\frac{A_\nu^{tot}(s, a)}{\alpha} \right)$, implying that the maximization problem has an optimal solution at $w^{tot*}(s, a) = 0$. So, putting all together, $w_\nu^{tot*}(s, a) = \max\{0, f'^{-1}(A_\nu^{tot}(s, a)/\alpha)\}$ is optimal for the maximization problem $\max_{w^{tot} \geq 0} T(w^{tot}(s, a))$.

To get derivatives of $\tilde{\mathcal{L}}(\nu^{tot})$, we note that, for any $y \in \mathbb{R}$, $\nabla f^*(y) = t^*$, where $y^* = \text{argmax}_{t \geq 0}(ty - f(t))$. Thus, the first-order derivative of $f^*\left(\frac{A_\nu^{tot}(\mathbf{s}, \mathbf{a})}{\alpha}\right)$ can be computed as:

$$\nabla_\theta f^*\left(\frac{A_\nu^{tot}(\mathbf{s}, \mathbf{a})}{\alpha}\right) = \frac{\nabla_\theta A_\nu^{tot}(\mathbf{s}, \mathbf{a})}{\alpha} w_{\nu^{tot}}^*(\mathbf{s}, \mathbf{a}),$$

which implies:

$$\nabla_\theta \tilde{\mathcal{L}}(\nu^{tot}) = (1 - \gamma) \mathbb{E}_{\mathbf{s} \sim p_0} [\nabla_\theta \nu^{tot}(\mathbf{s})] + \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [\nabla_\theta A_\nu^{tot}(\mathbf{s}, \mathbf{a}) w_{\nu^{tot}}^*(\mathbf{s}, \mathbf{a})],$$

we complete the proof. \square

A.2 PROOF OF THEOREM 4.2

Theorem. Assume the mixing network $\mathcal{M}_\theta[\cdot]$ is constructed with non-negative weights and convex activations, then $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ is convex in $\boldsymbol{\nu}$.

Proof. We first introduce the following lemma, which is essential to validate the convexity of $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$.

Lemma A.1. *If the mixing network are multi-level feed-forward, constructed with non-negative weights and convex activations, then $\mathcal{M}_\theta[\boldsymbol{\nu}(\mathbf{s})]$ and $\mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]$ are convex in $\boldsymbol{\nu}$*

Proof. To simplify the proof, we first prove a general result stating that if $\mathcal{M}_\theta[\mathbf{X}]$ is a multi-level feed-forward network with non-negative weights and convex activations, then $\mathcal{M}_\theta[\mathbf{X}]$ is convex in \mathbf{X} . To start, we note that any N -layer feed-forward network with input \mathbf{X} can be defined recursively as

$$F^0(\mathbf{X}) = \mathbf{X} \tag{14}$$

$$F^n(\mathbf{X}) = \sigma^n(F^{n-1}(\mathbf{X})) \times W_n + b_n, \quad n = 1, \dots, N, \tag{15}$$

where σ^n is a set of activation functions applied to each element of vector $F^{n-1}(\mathbf{X})$, and W_n and b_n are the weights and biases, respectively, at layer n . Therefore, we will prove the result by induction, i.e., $F^n(\mathbf{X})$ is convex and non-decreasing in \mathbf{X} for $n = 0, \dots, N$. Here we note that $F^n(\mathbf{X})$ is a vector, so when we say “ $F^n(\mathbf{X})$ is convex and non-decreasing in \mathbf{X} ,” it means each element of $F^n(\mathbf{X})$ is convex and non-decreasing in \mathbf{X} .

We first see that the claim indeed holds for $n = 0$. Now let us assume that $F^{n-1}(\mathbf{X})$ is convex and non-decreasing in \mathbf{X} ; we will prove that $F^n(\mathbf{X})$ is also convex and non-decreasing in \mathbf{X} . The non-decreasing property can be easily verified as we can see, given two vectors \mathbf{X} and \mathbf{X}' such that $\mathbf{X} \geq \mathbf{X}'$ (element-wise comparison), we have the following chain of inequalities:

$$\begin{aligned} F^{n-1}(\mathbf{X}) &\stackrel{(a)}{\geq} F^{n-1}(\mathbf{X}') \\ \sigma^n(F^{n-1}(\mathbf{X})) &\stackrel{(b)}{\geq} \sigma^n(F^{n-1}(\mathbf{X}')) \\ \sigma^n(F^{n-1}(\mathbf{X})) \times W_n + b_n &\stackrel{(c)}{\geq} \sigma^n(F^{n-1}(\mathbf{X}')) \times W_n + b_n, \end{aligned}$$

where (a) is due to the induction assumption that $F^{n-1}(\mathbf{X})$ is non-decreasing in \mathbf{X} , (b) is because σ^n is also non-decreasing, and (c) is because the weights W_n are non-negative.

To verify the convexity of $F^n(\mathbf{X})$, we will show that for any \mathbf{X}, \mathbf{X}' , and any scalar $\alpha \in (0, 1)$, the following holds:

$$\alpha F^n(\mathbf{X}) + (1 - \alpha) F^n(\mathbf{X}') \geq F^n(\alpha \mathbf{X} + (1 - \alpha) \mathbf{X}') \tag{16}$$

To this end, we write:

$$\begin{aligned} \alpha F^n(\mathbf{X}) + (1 - \alpha) F^n(\mathbf{X}') &= \left(\alpha \sigma^n(F^{n-1}(\mathbf{X})) + (1 - \alpha) \sigma^n(F^{n-1}(\mathbf{X}')) \right) \times W_n + b_n \\ &\stackrel{(d)}{\geq} \left(\sigma^n \left(\alpha F^{n-1}(\mathbf{X}) + (1 - \alpha) F^{n-1}(\mathbf{X}') \right) \right) \times W_n + b_n \\ &\stackrel{(e)}{\geq} \left(\sigma^n \left(F^{n-1}(\alpha \mathbf{X} + (1 - \alpha) \mathbf{X}') \right) \right) \times W_n + b_n \\ &= F^n(\alpha \mathbf{X} + (1 - \alpha) \mathbf{X}'). \end{aligned}$$

where (d) is due to the assumption that activation functions σ^n are convex and $W_n \geq 0$, and (e) is because $\alpha F^{n-1}(\mathbf{X}) + (1 - \alpha)F^{n-1}(\mathbf{X}') \geq F^{n-1}(\alpha\mathbf{X} + (1 - \alpha)\mathbf{X}')$ (because $F^{n-1}(\mathbf{X})$ is convex in \mathbf{X} , by the induction assumption), and the activation functions σ^n are non-decreasing and $W_n \geq 0$. So, we have:

$$\alpha F^n(\mathbf{X}) + (1 - \alpha)F^n(\mathbf{X}') \geq F^n(\alpha\mathbf{X} + (1 - \alpha)\mathbf{X}').$$

implying that $F^n(\mathbf{X})$ is convex in \mathbf{X} . We then complete the induction proof and conclude that $F^n(\mathbf{X})$ is convex and non-decreasing in \mathbf{X} for any $n = 0, \dots, N$.

From the result above, since both $\boldsymbol{\nu}(\mathbf{s})$ and $\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})$ are linear in $\boldsymbol{\nu}$, it follows that $\mathcal{M}_\theta[\boldsymbol{\nu}(\mathbf{s})]$ and $\mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]$ are convex with respect to $\boldsymbol{\nu}$. \square

We are now ready to prove the convexity of $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ with respect to $\boldsymbol{\nu}$. Directly verifying the convexity of this function is challenging, as it involves some complicated components such as $f^*\left(\frac{\mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]}{\alpha}\right)$, which is difficult to analyze. However, we recall that:

$$\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta) = \max_{w^{tot} \geq 0} \mathcal{L}(\boldsymbol{\nu}, \theta, w^{tot}),$$

where

$$\begin{aligned} \mathcal{L}(\boldsymbol{\nu}, \theta, w^{tot}) &= (1 - \gamma) \mathbb{E}_{\mathbf{s} \sim p_0} [\mathcal{M}_\theta[\boldsymbol{\nu}(\mathbf{s})]] \\ &\quad + \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\boldsymbol{\mu}_{tot}}} [-\alpha f(w_\nu^{tot}(\mathbf{s}, \mathbf{a})) + w_\nu^{tot}(\mathbf{s}, \mathbf{a}) \mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]]. \end{aligned}$$

From Lemma A.1, we know that $\mathcal{M}_\theta[\boldsymbol{\nu}(\mathbf{s})]$ and $\mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]$ are convex in $\boldsymbol{\nu}$, thus $\mathcal{L}(\boldsymbol{\nu}, \theta, w^{tot})$ is also convex in $\boldsymbol{\nu}$. We now follow the standard approach to verify the convexity of $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ as follows. Let $\boldsymbol{\nu}^1$ and $\boldsymbol{\nu}^2$ be two feasible value functions. Given any $\beta \in (0, 1)$, we will prove that:

$$\beta \tilde{\mathcal{L}}(\boldsymbol{\nu}^1, \theta) + (1 - \beta) \tilde{\mathcal{L}}(\boldsymbol{\nu}^2, \theta) \geq \tilde{\mathcal{L}}(\beta \boldsymbol{\nu}^1 + (1 - \beta) \boldsymbol{\nu}^2, \theta). \quad (17)$$

To see why this should hold, we recall that $\mathcal{L}(\boldsymbol{\nu}, \theta, w^{tot})$ is convex in $\boldsymbol{\nu}$ and $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta) = \max_{w^{tot} \geq 0} \mathcal{L}(\boldsymbol{\nu}, \theta, w^{tot})$, leading to the following chain of inequalities:

$$\begin{aligned} \beta \tilde{\mathcal{L}}(\boldsymbol{\nu}^1, \theta) + (1 - \beta) \tilde{\mathcal{L}}(\boldsymbol{\nu}^2, \theta) &= \beta \max_{w^{tot}} \mathcal{L}(\boldsymbol{\nu}^1, \theta, w^{tot}) + (1 - \beta) \max_{w^{tot}} \mathcal{L}(\boldsymbol{\nu}^2, \theta, w^{tot}) \\ &\geq \max_{w^{tot}} \{\beta \mathcal{L}(\boldsymbol{\nu}^1, \theta, w^{tot}) + (1 - \beta) \mathcal{L}(\boldsymbol{\nu}^2, \theta, w^{tot})\} \\ &\geq \max_{w^{tot}} \{\mathcal{L}(\beta \boldsymbol{\nu}^1 + (1 - \beta) \boldsymbol{\nu}^2, \theta, w^{tot})\} \\ &= \tilde{\mathcal{L}}(\beta \boldsymbol{\nu}^1 + (1 - \beta) \boldsymbol{\nu}^2, \theta). \end{aligned}$$

The last inequality directly confirms Eq. 17, implying the convexity of $\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ in $\boldsymbol{\nu}$, as desired. \square

A.3 PROOF OF PROPOSITION 4.3

Proposition. *Let π_i^* be the optimal solution to the local weighted BC 9. Then $\boldsymbol{\pi}_{tot}^*(\mathbf{a}|\mathbf{s}) = \prod_{i \in \mathcal{N}} \pi_i^*(a_i|s_i)$ is also optimal for the global weighted BC problem 8.*

Proof. To prove that $\boldsymbol{\pi}_{tot}^*(\mathbf{a}|\mathbf{s}) = \prod_{i \in \mathcal{N}} \pi_i^*(a_i|s_i)$ is optimal for the global WBC problem 8, we need to verify that

$$\mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\boldsymbol{\mu}_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \boldsymbol{\pi}_{tot}(\mathbf{a}|\mathbf{s})] \leq \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\boldsymbol{\mu}_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \boldsymbol{\pi}_{tot}^*(\mathbf{a}|\mathbf{s})]$$

for any global policy $\boldsymbol{\pi}_{tot} \in \Pi_{tot}$.

Since $\boldsymbol{\pi}_{tot}$ is decomposable, there exist local policies π_i such that

$$\boldsymbol{\pi}_{tot}(\mathbf{a}|\mathbf{s}) = \prod_{i \in \mathcal{N}} \pi_i(a_i|s_i).$$

As a result, we have the following inequalities:

$$\begin{aligned}
\mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \pi_{tot}(\mathbf{a}|\mathbf{s})] &= \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} \left[w^{tot*}(\mathbf{s}, \mathbf{a}) \sum_{i \in \mathcal{N}} \log \pi_i(a_i|s_i) \right] \\
&= \sum_{i \in \mathcal{N}} \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \pi_i(a_i|s_i)] \\
&\leq \sum_{i \in \mathcal{N}} \max_{\pi'_i} \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \pi'_i(a_i|s_i)] \\
&= \sum_{i \in \mathcal{N}} \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \pi_i^*(a_i|s_i)] \\
&= \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\mu_{tot}}} [w^{tot*}(\mathbf{s}, \mathbf{a}) \log \pi_{tot}^*(\mathbf{a}|\mathbf{s})],
\end{aligned}$$

which directly implies that π_{tot}^* is optimal for the global WBC problem 8.

□

B ADDITIONAL DETAILS

B.1 FACTORIZATION ASPECT OF THE LEARNING OBJECTIVE IN COMADICE

In this section, we delve into the main learning objective function of ComaDICE to explore its factorization aspect. Specifically, we show that, under certain conditions on the mixing network and the f-divergence function, optimizing the objective function $\tilde{\mathcal{L}}$ is approximately equivalent to optimizing factorized occupancy ratios.

To see this, let us consider the main learning objective with mixing networks:

$$\tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta) = (1 - \gamma)\mathbb{E}_{\mathbf{s} \sim p_0}[\mathcal{M}_\theta(\boldsymbol{\nu}(\mathbf{s})]] + \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \rho^{\boldsymbol{\mu}_{tot}}} \left[\alpha f^* \left(\frac{\mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})]}{\alpha} \right) \right],$$

where \mathcal{M}_θ is the mixing network.

Assume that the mixing structure is linear in its inputs, i.e.,

$$\mathcal{M}_\theta(\boldsymbol{\nu}(\mathbf{s})) = \sum_i \beta_i \nu_i(s_i), \quad \mathcal{M}_\theta[\mathbf{q}(\mathbf{s}, \mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})] = \sum_i \beta_i (q_i(s_i, a_i) - \nu_i(s_i)),$$

where β_i are non-negative weights of the mixing network. Moreover, assume that the f-divergence function is chi-square. Under these assumptions, the learning objective can be written as:

$$\begin{aligned} \tilde{\mathcal{L}}(\boldsymbol{\nu}, \theta) &= \sum_i \beta_i (1 - \gamma) \mathbb{E}_{s_i \sim p_{i0}} [\nu_i(s_i)] + \sum_i \mathbb{E}_{(s_i, a_i) \sim \rho^{\boldsymbol{\mu}_{tot}}} \left[\alpha f^* \left(\sum_i \beta_i \frac{q_i(s_i, a_i) - \nu_i(s_i)}{\alpha} \right) \right], \\ &\stackrel{(a)}{\approx} \sum_i \beta_i \mathcal{L}_i(\nu_i), \end{aligned}$$

where

$$\mathcal{L}_i(\nu_i) = \beta_i (1 - \gamma) \mathbb{E}_{s_i \sim p_{i0}} [\nu_i(s_i)] + \mathbb{E}_{(s_i, a_i) \sim \rho^{\boldsymbol{\mu}_{tot}}} \left[\alpha f^* \left(\frac{q_i(s_i, a_i) - \nu_i(s_i)}{\alpha} \right) \right].$$

Here, the approximation holds because the mixing network is linear in ν_i , and the f-divergence is chi-square, where $(f'_{\chi^2})^{-1}(x) = x + 1$.

We now see that minimizing the local function $\mathcal{L}_i(\nu_i)$ is equivalent to:

$$\max_{\pi_i} \mathbb{E}_{(s_i, a_i) \sim \rho^{\pi_i}} [r_i(s_i, a_i)] - \alpha D^f(\rho^{\pi_i} \parallel \rho^{\mu_i}),$$

which is essentially solving the OptDICE learning problem for each individual agent.

Thus, the above discussion implies that optimizing the main objective function $\tilde{\mathcal{L}}$ of ComaDICE, under the setting of a linear mixing network and chi-square divergence, is approximately equivalent to optimizing factorized policies. This further implies the global-local consistency property mentioned in the main paper. It is also worth noting that the setting of a linear mixing network and chi-square divergence is exactly what we employ in our experiments, yielding the best performance compared to the variants.

When using a two-layer mixing network, the equivalence becomes harder to achieve. However, since the mixing network in our setting consists of non-negative weights, minimizing the global training objective $\tilde{\mathcal{L}}$ is expected to behave similarly to minimizing each local function $\tilde{\mathcal{L}}_i$, partially indicating the global-local consistency and the factorization aspect of ComaDICE.

In comparison with other DICE-based approaches such as AlberDICE and OptDICE, ComaDICE takes a distinctive approach by learning a global occupancy ratio and employing a factorization method to decompose the global learning variables into local ones, leveraging local information. This design captures the contribution of each local agent to the global objective, enabling ComaDICE to effectively model the interconnections between agents. Furthermore, during the policy extraction phase, local policies are optimized using a shared global occupancy ratio, which incorporates aspects of credit assignment across agents—an important feature not present in AlberDICE.

B.2 OFFLINE MULTI-AGENT DATASETS

Instances		Trajectories	Samples	Agents	State dim	Obs dim	Action dim	Average returns
2c_vs_64zg	poor	0.3K	21.7K	2	675	478	70	8.9±1.0
	medium	1.0K	75.9K	2	675	478	70	13.0±1.4
	good	1.0K	118.4K	2	675	478	70	19.9±1.3
5m_vs_6m	poor	1.0K	113.7K	5	156	124	12	8.5±1.2
	medium	1.0K	138.6K	5	156	124	12	11.0±0.6
	good	1.0K	138.7K	5	156	124	12	20.0±0.0
6h_vs_8z	poor	1.0K	145.5K	6	213	172	14	9.1±0.8
	medium	1.0K	177.1K	6	213	172	14	12.0±1.3
	good	1.0K	228.2K	6	213	172	14	17.8±2.1
corridor	poor	1.0K	307.6K	6	435	346	30	4.9±1.7
	medium	1.0K	756.1K	6	435	346	30	13.1±1.3
	good	1.0K	601.0K	6	435	346	30	19.9±1.0
Protoss	5_vs_5	1.0K	60.8K	5	130	92	11	16.8±6.3
	10_vs_10	1.0K	68.3K	10	310	182	16	15.7±5.2
	10_vs_11	1.0K	62.9K	10	327	191	17	15.3±5.7
	20_vs_20	1.0K	76.7K	20	820	362	26	16.2±4.7
	20_vs_23	1.0K	65.0K	20	901	389	29	14.0±4.5
Terran	5_vs_5	1.0K	47.6K	5	120	82	11	15.2±7.2
	10_vs_10	1.0K	56.4K	10	290	162	16	14.7±6.2
	10_vs_11	1.0K	52.5K	10	306	170	17	12.1±5.7
	20_vs_20	1.0K	63.0K	20	780	322	26	14.0±6.0
	20_vs_23	1.0K	51.3K	20	858	346	29	11.7±5.7
Zerg	5_vs_5	1.0K	27.5K	5	120	82	11	10.4±5.0
	10_vs_10	1.0K	31.9K	10	290	162	16	14.7±6.0
	10_vs_11	1.0K	30.9K	10	306	170	17	12.0±5.1
	20_vs_20	1.0K	35.4K	20	780	322	26	12.3±4.2
	20_vs_23	1.0K	32.8K	20	858	346	29	10.8±4.0
Hopper	expert	1.5K	999K	3	42	14	1	2452.0±1097.9
	medium	4.0K	915K	3	42	14	1	723.6±211.7
	m-replay	4.2K	1311K	3	42	14	1	746.4±671.9
	m-expert	5.5K	1914K	3	42	14	1	1190.6±973.4
Ant	expert	1.0K	1000K	2	226	113	4	2055.1±22.1
	medium	1.0K	1000K	2	226	113	4	1418.7±37.0
	m-replay	1.8K	1750K	2	226	113	4	1029.5±141.3
	m-expert	2.0K	2000K	2	226	113	4	1736.9±319.6
Half Cheetah	expert	1.0K	1000K	6	138	23	1	2785.1±1053.1
	medium	1.0K	1000K	6	138	23	1	1425.7±520.1
	m-replay	1.0K	1000K	6	138	23	1	655.8±590.4
	m-expert	2.0K	2000K	6	138	23	1	2105.4±1073.2

Table 4: Overview of datasets used in experiments, including details of trajectories, samples, agent counts, and state, observation, and action space dimensions across SMACv1, SMACv2, and MaMu-joco environments, with average returns indicating performance levels.

B.3 IMPLEMENTATION DETAILS

Our experiments were implemented using PyTorch and executed in parallel on a single NVIDIA® H100 NVL Tensor Core GPU. Our study required running a large number of sub-tasks, specifically 1,365 in total (i.e., 39 instances across 7 algorithms with 5 different random seeds each).

Algorithm 1 ComaDICE: Offline Cooperative MARL with Stationary DIstribution Correction Estimation

- 1: **Input:** Parameters $\theta, \psi_q, \psi_\nu, \eta_i$ and the corresponding learning rates $\lambda_\theta, \lambda_{\psi_q}, \lambda_{\psi_\nu}, \lambda_\eta$, respectively. Offline data \mathcal{D} .
- 2: **Output:** Local optimized policies π_i .
- 3: # Training the occupancy ratio $w^{\text{tot}*}$
- 4: **for** a certain number of training steps **do**
- 5: $\psi_q = \psi_q - \lambda_{\psi_q} \nabla_{\psi_q} \mathcal{L}(\psi_q)$ # Update Q-function towards the MSE in 10
- 6: $\theta = \theta - \lambda_\theta \nabla_\theta \tilde{\mathcal{L}}(\psi_\nu, \theta)$ # Update θ to minimize the loss in 11
- 7: $\psi_\nu = \psi_\nu - \lambda_{\psi_\nu} \nabla_{\psi_\nu} \tilde{\mathcal{L}}(\psi_\nu, \theta)$ # Update ψ_ν to minimize the loss in 11
- 8: **end for**
- 9: # Training local policy
- 10: **for** a certain number of training steps **do**
- 11: $\eta_i = \eta_i + \lambda_\eta \nabla_{\eta_i} \mathcal{L}_\pi(\eta_i)$ # Update the local policy by optimizing 12
- 12: **end for**
- 13: Return $\pi_i(a_i|o_i; \eta_i), i = 1, \dots, n$

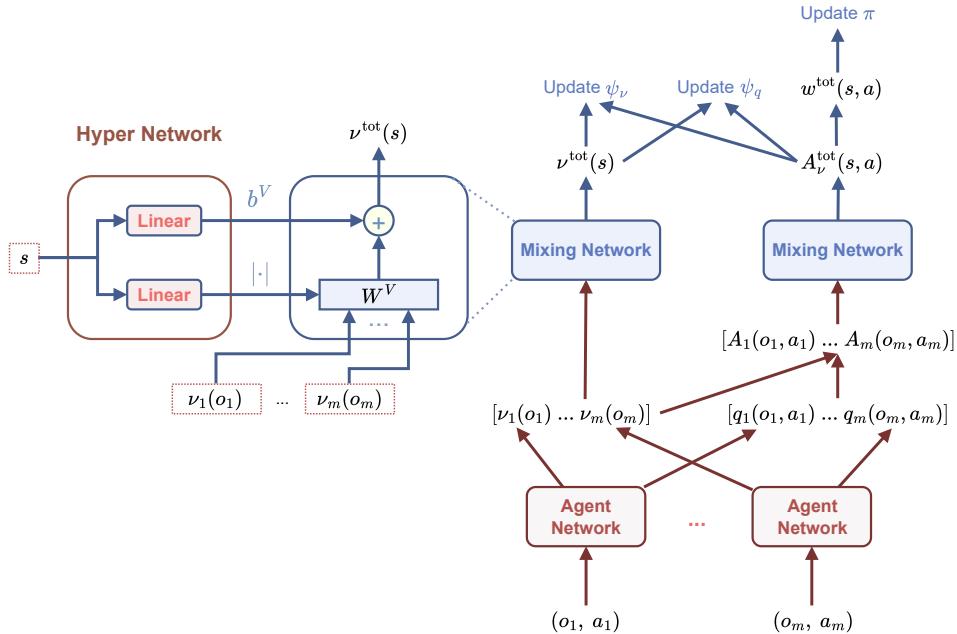


Figure 2: Our ComaDICE model architecture.

The offline datasets for each instance are substantial, reaching sizes of up to 7.4 GB. To manage this, we developed a preprocessing step designed to optimize data handling and improve computational efficiency. This process involves reading all transitions from each dataset and combining individual trajectory files into a single large NumPy object that contains batches of trajectories. In this step, we define the data type for each element, such as states (float32), actions (int64), and dones (bool), ensuring consistent and efficient data storage. The processed data is then saved into a compressed NumPy file, which significantly boosts computing performance.

Despite these optimizations, loading the entire dataset still requires a large amount of RAM. By leveraging parallel processing and efficient data management strategies, we effectively managed the extensive computational and memory demands of our experiments. This approach allowed us to handle the large-scale data and complex computations necessary for our study.

B.3.1 HYPER-PARAMETERS

Hyperparameter	Value
Optimizer	Adam
Learning rate (Q-value and policy networks)	1×10^{-4}
Tau (τ)	0.005
Gamma (γ)	0.99
Batch size	128
Agent hidden dimension	256
Mixer hidden dimension	64
Number of seeds	5
Number of episodes per evaluation step	32
Number of evaluation steps	100
Lambda scale (λ)	1.0
Alpha (α)	10
f-divergence	soft- χ^2

Table 5: Hyperparameters for our algorithm

In our study, we developed two versions of our algorithm: a continuous version for MaMujoco using Gaussian distributions (`torch.distributions.Normal`), and a discrete version for SMACv1 and SMACv2 using Categorical distributions (`torch.distributions.Categorical`). In the discrete setting, action probabilities are computed using softmax over available actions only, ensuring zero probability for unavailable actions, which enhances the accuracy of log likelihood calculations. Key hyperparameters are listed on the Table 5. Experiments were conducted with 5 seeds, 32 episodes per evaluation step, and 100 evaluation steps.

B.4 ADDITIONAL EXPERIMENTAL DETAILS

We evaluate the performance of our ComaDICE algorithm using two key metrics: mean and standard deviation (std) of returns and winrates. Returns measure the average rewards accumulated by agents, calculated across five random seeds to ensure robustness, while winrates, applicable only to competitive environments like SMACv1 and SMACv2, indicate the success rate against other agents. For cooperative settings such as MaMujoco, winrates are not applicable. We also include figures showing evaluation curves, highlighting how each method’s performance evolves during training with offline datasets. These metrics and visualizations provide a comprehensive overview of our algorithm’s effectiveness and consistency in various MARL tasks.

B.4.1 RETURNS

Tables 6, 7, 8, 9, and 10 present the returns from our experimental results across the SMACv1, SMACv2, and Multi-Agent MuJoCo environments, highlighting the performance of our proposed algorithm, ComaDICE, alongside baseline methods such as BC, BCQ, CQL, ICQ, OMAR, OMIGA, OptDICE and AlberDICE. Our results demonstrate that ComaDICE consistently achieves superior returns, particularly excelling in more complex difficulty tasks. Figures 3, 4, and 5 illustrate the learning curves for these algorithms, showing that ComaDICE not only outperforms other algorithms in terms of mean returns but also exhibits lower standard deviation, indicating robust and stable performance. This suggests that ComaDICE effectively handles distributional shifts in offline settings. These findings underscore our algorithm’s adaptability and effectiveness in diverse multi-agent coordination scenarios, setting a new benchmark in offline MARL.

Instances		BC	BCQ	CQL	ICQ	OMAR	OMIGA	OptDICE	AlberDICE	ComaDICE
2c_vs_64zg	poor	11.6 ± 0.4	12.5 ± 0.2	10.8 ± 0.5	12.6 ± 0.2	11.3 ± 0.5	13.0 ± 0.7	10.8 ± 0.4	11.0 ± 0.2	12.1 ± 0.5
	medium	13.4 ± 1.9	15.6 ± 0.4	12.8 ± 1.6	15.6 ± 0.6	10.2 ± 0.2	16.0 ± 0.2	11.2 ± 0.8	15.2 ± 0.5	16.3 ± 0.7
	good	17.9 ± 1.3	19.1 ± 0.3	18.5 ± 1.0	18.8 ± 0.2	17.3 ± 0.8	19.1 ± 0.3	14.9 ± 1.2	17.9 ± 0.6	20.3 ± 0.1
5m_vs_6m	poor	7.0 ± 0.5	7.6 ± 0.4	7.4 ± 0.1	7.3 ± 0.2	7.3 ± 0.4	7.5 ± 0.2	7.1 ± 0.2	5.7 ± 1.2	8.1 ± 0.5
	medium	7.0 ± 0.8	7.6 ± 0.1	7.8 ± 0.1	7.8 ± 0.3	7.1 ± 0.5	7.9 ± 0.6	5.9 ± 1.3	7.7 ± 0.4	8.7 ± 0.4
	good	7.0 ± 0.5	7.8 ± 0.1	8.1 ± 0.2	7.9 ± 0.3	7.4 ± 0.6	8.3 ± 0.4	5.8 ± 1.5	6.5 ± 0.6	8.7 ± 0.5
6h_vs_8z	poor	8.6 ± 0.8	10.8 ± 0.2	10.8 ± 0.5	10.6 ± 0.1	10.6 ± 0.2	11.3 ± 0.2	9.8 ± 0.3	10.6 ± 0.3	11.4 ± 0.6
	medium	9.5 ± 0.3	11.8 ± 0.2	11.3 ± 0.3	11.1 ± 0.3	10.4 ± 0.2	12.2 ± 0.2	10.8 ± 0.6	12.3 ± 0.4	12.8 ± 0.2
	good	10.0 ± 1.7	12.2 ± 0.2	10.4 ± 0.2	11.8 ± 0.1	9.9 ± 0.3	12.5 ± 0.2	9.1 ± 0.7	10.0 ± 0.3	13.1 ± 0.5
corridor	poor	2.9 ± 0.6	4.5 ± 0.9	4.1 ± 0.6	4.5 ± 0.3	4.3 ± 0.5	5.6 ± 0.3	6.3 ± 0.5	5.0 ± 0.5	6.4 ± 0.5
	medium	7.4 ± 0.8	10.8 ± 0.9	7.0 ± 0.7	11.3 ± 1.6	7.3 ± 0.7	11.7 ± 1.3	11.2 ± 0.7	9.3 ± 0.3	12.9 ± 0.6
	good	10.8 ± 2.6	15.2 ± 1.2	5.2 ± 0.8	15.5 ± 1.1	6.7 ± 0.7	15.9 ± 0.9	13.4 ± 2.1	14.4 ± 1.2	18.0 ± 0.1

Table 6: Comparison of average returns for ComaDICE and baselines on SMACv1 benchmarks.

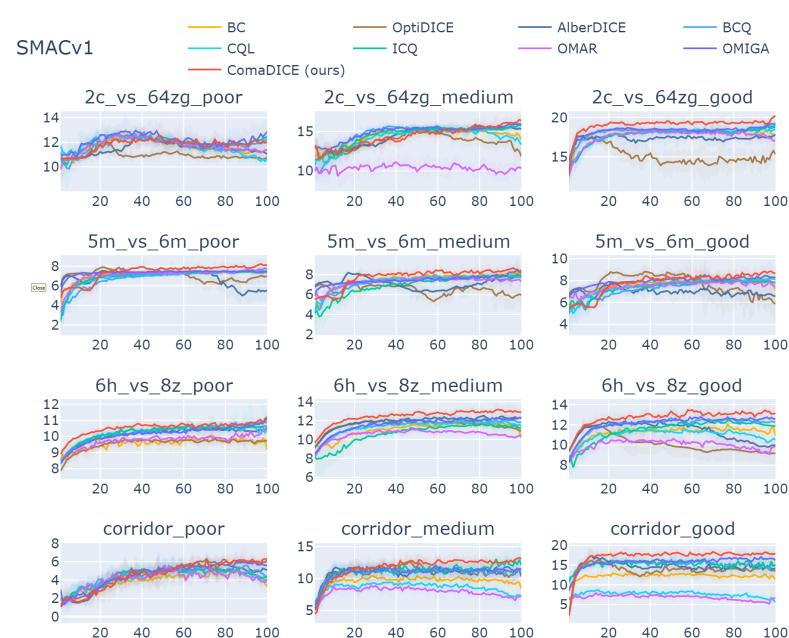


Figure 3: Evaluation of SMACv1 tasks comparing the returns achieved by ComaDICE and baselines.

Instances		BC	BCQ	CQL	ICQ	OMAR	OMIGA	OptDICE	AlberDICE	ComaDICE (ours)
Protoss	5_vs_5	13.2 ± 0.7	6.8 ± 1.6	9.3 ± 1.6	10.7 ± 1.2	8.9 ± 0.8	14.3 ± 1.4	10.8 ± 1.2	12.6 ± 0.9	14.4 ± 1.1
	10_vs_10	12.0 ± 1.9	7.7 ± 1.3	11.3 ± 0.9	10.4 ± 1.6	8.8 ± 0.6	14.2 ± 1.5	9.5 ± 0.8	11.8 ± 0.9	14.6 ± 1.8
	10_vs_11	11.2 ± 0.5	5.2 ± 1.4	7.9 ± 0.8	10.3 ± 0.7	8.0 ± 0.3	12.1 ± 0.5	10.0 ± 0.5	9.8 ± 0.3	13.2 ± 0.9
	20_vs_20	13.1 ± 0.5	4.8 ± 0.6	10.5 ± 0.9	11.8 ± 0.5	9.1 ± 0.5	14.0 ± 0.9	10.0 ± 2.0	10.1 ± 0.6	14.8 ± 1.0
	20_vs_23	11.2 ± 0.5	3.5 ± 0.6	5.6 ± 0.7	10.2 ± 0.7	7.4 ± 0.7	13.0 ± 1.1	8.1 ± 1.4	8.8 ± 0.8	13.3 ± 0.9
Terran	5_vs_5	10.8 ± 1.4	6.4 ± 1.1	6.5 ± 0.9	6.8 ± 0.6	6.9 ± 0.6	10.5 ± 1.2	6.4 ± 1.1	8.1 ± 1.4	10.7 ± 1.5
	10_vs_10	10.3 ± 0.3	4.6 ± 0.4	6.8 ± 0.6	8.7 ± 1.4	7.6 ± 1.0	10.1 ± 0.6	6.0 ± 1.6	8.2 ± 1.0	11.8 ± 0.9
	10_vs_11	9.0 ± 0.7	3.6 ± 1.1	5.5 ± 0.2	5.5 ± 0.9	5.9 ± 0.7	8.8 ± 1.4	4.8 ± 1.2	6.2 ± 0.9	9.4 ± 0.9
	20_vs_20	10.8 ± 0.8	3.9 ± 0.6	4.3 ± 0.6	8.3 ± 0.3	7.3 ± 0.4	10.5 ± 0.7	6.3 ± 1.8	5.9 ± 1.2	11.8 ± 0.5
	20_vs_23	7.2 ± 1.0	1.2 ± 1.0	1.6 ± 0.2	5.3 ± 0.5	5.1 ± 0.3	7.9 ± 0.6	4.4 ± 0.7	3.9 ± 0.8	8.2 ± 0.7
Zerg	5_vs_5	10.5 ± 2.2	6.6 ± 0.2	6.7 ± 0.5	6.5 ± 0.9	7.7 ± 0.9	8.9 ± 1.1	8.2 ± 1.8	9.5 ± 0.8	10.7 ± 2.0
	10_vs_10	11.0 ± 0.8	7.3 ± 1.0	7.2 ± 0.3	7.7 ± 1.1	7.5 ± 0.8	11.8 ± 1.6	7.8 ± 1.0	8.5 ± 0.3	11.5 ± 1.0
	10_vs_11	9.2 ± 1.1	7.6 ± 0.9	6.7 ± 0.4	6.8 ± 1.0	6.5 ± 1.0	9.5 ± 1.2	7.2 ± 0.7	9.1 ± 0.5	11.0 ± 0.9
	20_vs_20	9.3 ± 0.5	3.7 ± 0.4	4.7 ± 0.3	6.9 ± 0.5	6.9 ± 0.8	9.2 ± 0.5	7.3 ± 0.7	8.3 ± 0.5	9.4 ± 1.2
	20_vs_23	8.5 ± 0.7	3.3 ± 0.3	4.1 ± 0.6	6.9 ± 0.5	5.7 ± 0.4	9.8 ± 0.6	7.1 ± 1.2	8.8 ± 0.5	10.5 ± 0.8

Table 7: Comparison of average returns for ComaDICE and baselines on SMACv2 tasks.

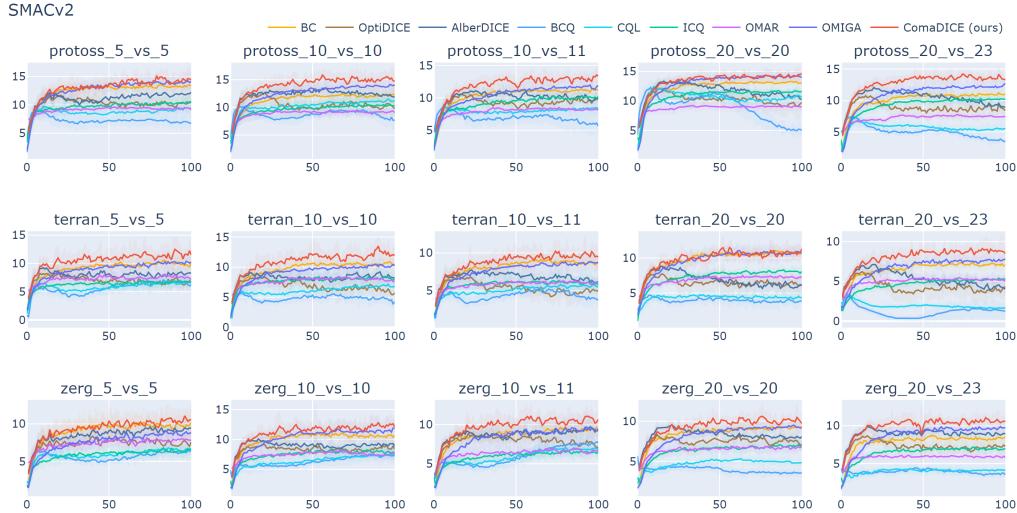


Figure 4: Evaluation of SMACv2 tasks comparing the returns achieved by ComaDICE and baselines.

Method	Hopper-v2			
	expert	medium	medium-replay	medium-expert
BC	209.8±191.1	511.9±7.4	133.3±53.5	155.3±111.5
BCQ	77.9±58.0	44.6±20.6	26.5±24.0	54.3±23.7
CQL	159.1±313.8	401.3±199.9	31.4±15.2	64.8±123.3
ICQ	754.7±806.3	501.8±14.0	195.4±103.6	355.4±373.9
OMAR	2.4±1.5	21.3±24.9	3.3±3.2	1.4±0.9
OMIGA	859.6±709.5	1189.3±544.3	774.2±494.3	709.0±595.7
OptiDICE	655.9±120.1	204.1±41.9	257.8±55.3	400.9±132.5
AlberDICE	844.6±556.5	216.9±35.3	419.2±243.5	515.1±303.4
ComaDICE (ours)	2827.7±62.9	822.6±66.2	906.3±242.1	1362.4±522.9

Table 8: Comparison of average returns on Hopper-v2 of MaMujoco benchmarks.

B.4.2 WINRATES

In this section, we analyze the winrates of our ComaDICE algorithm across various multi-agent reinforcement learning scenarios. Winrates are crucial in competitive environments like SMACv1 and SMACv2, as they measure the algorithm’s success against other agents. Our results demonstrate that ComaDICE consistently achieves higher winrates compared to baseline methods. Notably, ComaDICE performs well across both simple and complex tasks, reflecting its robustness and adaptability. As shown in Tables 1 and 2, as well as Figures 6 and 7, ComaDICE not only excels in average winrates but also exhibits lower variance, indicating stable performance across different trials. These findings highlight ComaDICE’s ability to effectively manage distributional shifts and the OOD issue.

B.5 ABLATION STUDY: DIFFERENT VALUES OF ALPHA

We provide more experimental details for ablation study assessing the impact of varying the regularization parameter alpha (α) on the performance of our ComaDICE.

B.5.1 RETURNS

Our results, in Tables 11, 12, and 13, show that the performance of ComaDICE is sensitive to the choice of α . Lower values of α tend to prioritize imitation learning, leading to suboptimal performance in terms of returns, whereas higher values facilitate better adaptation to the offline data,

Method	Ant-v2			
	expert	medium	medium-replay	medium-expert
BC	2046.3±6.2	1421.1±7.9	994.0±20.3	1561.7±64.8
BCQ	1317.7±286.3	1059.6±91.2	950.8±48.8	1020.9±242.7
CQL	1042.4±2021.6	533.9±1766.4	234.6±1618.3	800.2±1621.5
ICQ	2050.0±11.9	1412.4±10.9	1016.7±53.5	1590.2±85.6
OMAR	312.5±297.5	-1710.0±1589.0	-2014.2±844.7	-2992.8±7.0
OMIGA	2055.5±1.6	1418.4±5.4	1105.1±88.9	1720.3±110.6
OptDICE	1717.2±27.0	1199.0±26.8	869.4±62.6	1293.2±183.1
AlberDICE	1896.8±33.7	1304.3±2.6	1042.8±80.8	1780.0±23.6
ComaDICE (ours)	2056.9±5.9	1425.0±2.9	1122.9±61.0	1813.9±68.4

Table 9: Comparison of average returns on Ant-v2 of MaMujoco benchmarks.

Method	HalfCheetah-v2			
	expert	medium	medium-replay	medium-expert
BC	3251.2±386.8	2280.3±178.2	1886.2±390.8	2451.9±783.0
BCQ	2992.7±629.7	2590.5±1110.4	-333.6±152.1	3543.7±780.9
CQL	1189.5±1034.5	1011.3±1016.9	1998.7±693.9	1194.2±1081.0
ICQ	2955.9±459.2	2549.3±96.3	1922.4±612.9	2834.0±420.3
OMAR	-206.7±161.1	-265.7±147.0	-235.4±154.9	-253.8±63.9
OMIGA	3383.6±552.7	3608.1±237.4	2504.7±83.5	2948.5±518.9
OptDICE	2601.6±461.9	305.3±946.8	-912.9±1363.9	-2485.8±2338.4
AlberDICE	3356.4±546.9	522.4±315.5	440.0±528.0	2288.2±759.5
ComaDICE (ours)	4082.9±45.7	2664.7±54.2	2855.0±242.2	3889.7±81.6

Table 10: Comparison of average returns on HalfCheetah-v2 of MaMujoco benchmarks.

achieving superior returns. Notably, an α value of 10 consistently yielded the best results across most tasks, indicating an optimal balance between exploration and exploitation in offline settings. This ablation study underscores the importance of selecting an appropriate α to enhance the algorithm’s robustness and effectiveness in handling distributional shifts in offline multi-agent reinforcement learning scenarios.

Instances		$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$
2c_vs_64zg	poor	10.6±0.5	11.1±0.4	11.1±0.1	12.1±0.5	11.8±0.2
	medium	9.6±0.5	13.1±0.8	12.5±2.4	16.3±0.7	16.0±0.3
	good	11.1±1.4	9.6±2.7	17.4±0.5	20.3±0.1	19.9±0.1
5m_vs_6m	poor	5.7±0.1	5.1±0.3	7.1±0.7	8.1±0.5	7.7±0.3
	medium	5.6±0.1	5.3±0.2	7.8±0.8	8.7±0.4	8.5±0.7
	good	5.7±0.1	5.7±0.2	7.8±0.5	8.7±0.5	8.8±0.8
6h_vs_8z	poor	8.5±0.2	9.6±0.3	10.0±0.3	11.4±0.6	10.7±0.4
	medium	8.5±0.6	10.5±0.8	10.7±0.5	12.8±0.2	12.3±0.3
	good	7.9±0.1	9.5±0.6	11.3±0.6	13.1±0.5	12.8±0.4
corridor	poor	2.1±0.4	3.7±1.0	6.1±0.8	6.4±0.5	5.0±1.1
	medium	1.7±1.0	2.2±1.7	11.3±0.3	12.9±0.6	13.3±0.1
	good	4.7±2.4	3.8±5.0	15.7±0.3	18.0±0.1	17.4±0.1

Table 11: Impact of alpha on returns for ComaDICE and baselines in SMACv1.

B.5.2 WINRATES

In the A.4.2 section of the appendix, we investigate the impact of varying α on winrates across different multi-agent reinforcement learning environments. We observe that an intermediate α value of 10 consistently yields optimal results, suggesting it strikes an effective balance between conservative policy adherence and exploration of the offline dataset. This section underscores the

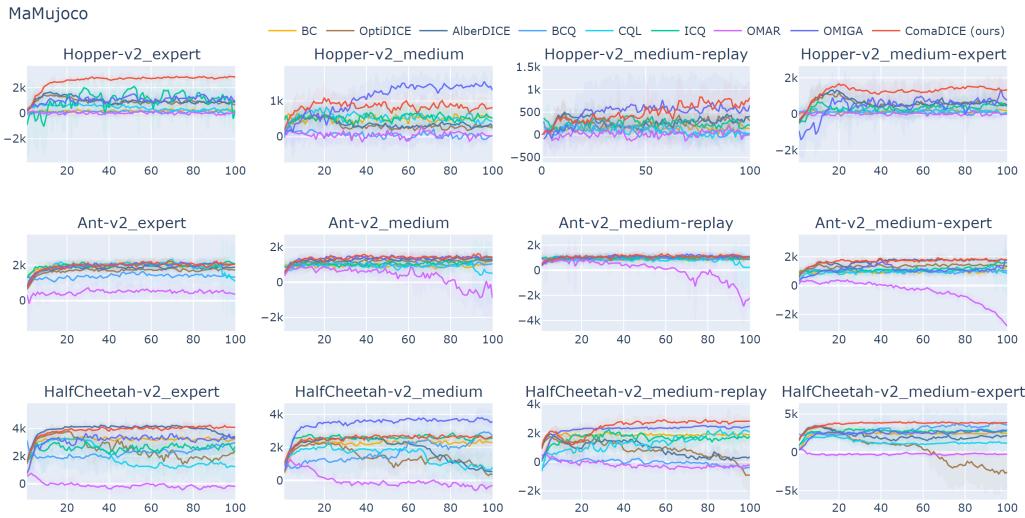


Figure 5: Evaluation of MaMujoco tasks comparing the returns achieved by ComaDICE and baselines.

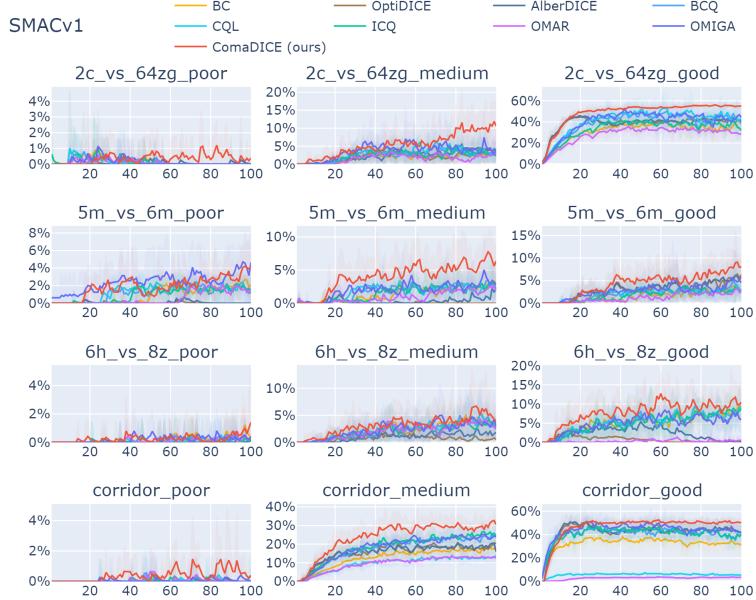


Figure 6: Evaluation of SMACv1 tasks comparing the winrates achieved by ComaDICE and baselines.

importance of fine-tuning α to enhance the robustness and efficacy of the ComaDICE algorithm in managing distributional shifts within competitive multi-agent settings.

B.6 ABLATION STUDY: DIFFERENT FORMS OF F-DIVERGENCE

We conduct an ablation study to examine the effects of different functions of f -divergence on the performance of our ComaDICE algorithm across various multi-agent reinforcement learning environments. The study specifically evaluates three types of f -divergence: Kullback-Leibler (KL), χ^2 , and Soft- χ^2 .

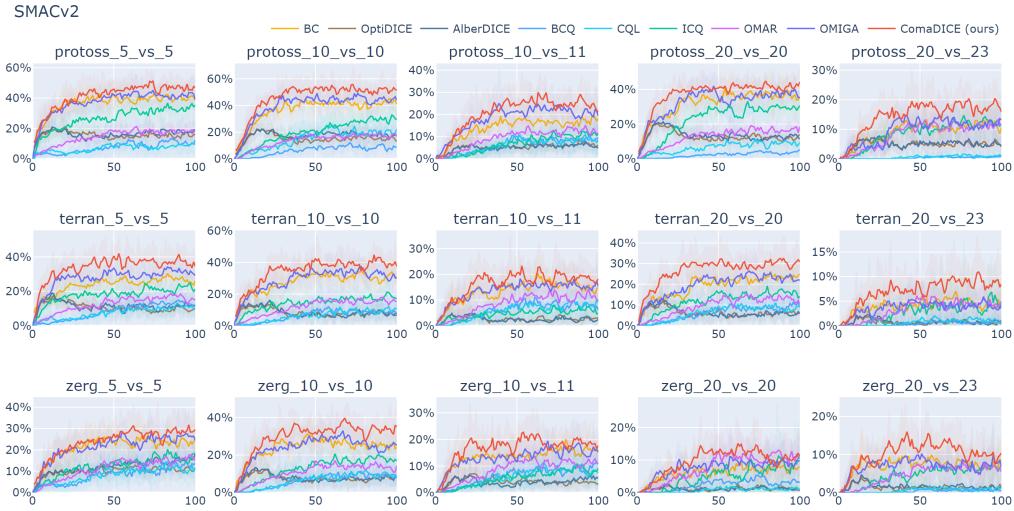


Figure 7: Evaluation of SMACv2 tasks comparing the winrates achieved by ComaDICE and baselines.

Instances	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$
Protoss	5_vs_5	12.2±1.0	13.1±1.3	13.2±1.1	14.4±1.1
	10_vs_10	12.8±0.9	14.0±0.8	13.4±1.2	14.6±1.8
	10_vs_11	9.9±1.1	11.1±0.8	11.3±1.2	13.2±0.9
	20_vs_20	10.3±0.5	11.1±1.0	12.2±0.9	14.8±1.0
	20_vs_23	8.0±2.3	11.2±1.2	11.7±0.6	13.3±0.9
Terran	5_vs_5	11.1±1.8	10.1±1.2	9.0±1.0	10.7±1.5
	10_vs_10	8.5±0.8	10.3±0.7	10.4±1.1	11.8±0.9
	10_vs_11	7.5±0.7	8.6±2.1	8.5±1.6	9.4±0.9
	20_vs_20	6.2±1.1	6.4±1.7	9.1±0.7	11.8±0.5
	20_vs_23	5.5±1.1	6.5±1.6	6.5±0.8	8.2±0.7
Zerg	5_vs_5	7.9±0.6	9.3±0.9	10.5±1.4	10.7±2.0
	10_vs_10	10.9±1.5	11.4±1.5	11.8±0.7	11.5±1.0
	10_vs_11	10.1±2.5	9.1±1.2	10.0±1.2	11.0±0.9
	20_vs_20	8.0±0.5	9.2±1.3	9.2±1.0	9.4±1.2
	20_vs_23	9.1±1.1	10.0±0.7	10.4±0.6	10.5±0.8

Table 12: Impact of alpha on returns for ComaDICE and baselines in SMACv2.

KL-Divergence: This is a well-known measure of how one probability distribution diverges from a second, expected probability distribution. It is defined as:

$$f_{\text{KL}}(x) = x \log x - x + 1$$

The corresponding inverse derivative, which is used in optimization, is:

$$(f'_{\text{KL}})^{-1}(x) = \exp(x - 1)$$

KL-divergence can lead to numerical instability due to the exponential function, especially when the values become large.

χ^2 -Divergence: This divergence measures the difference between two probability distributions by considering the square of the differences. It is expressed as:

$$f_{\chi^2}(x) = \frac{1}{2}(x - 1)^2$$

The inverse derivative is:

$$(f'_{\chi^2})^{-1}(x) = x + 1$$

While this function avoids the exponential instability seen in KL-divergence, it may suffer from zero gradients for negative values, which can slow down or halt training.

Instances	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$
Hopper	expert	147.3 \pm 67.9	107.9 \pm 65.5	545.7 \pm 820.6	2827.7 \pm 62.9
	medium	149.6 \pm 96.8	107.5 \pm 66.9	244.7 \pm 267.5	822.6 \pm 66.2
	m-replay	165.6 \pm 104.1	109.6 \pm 38.7	155.6 \pm 61.6	906.3 \pm 242.1
	m-expert	119.1 \pm 77.1	95.6 \pm 69.5	58.8 \pm 26.1	1362.4 \pm 522.9
Ant	expert	1016.4 \pm 196.5	1179.0 \pm 273.7	1927.7 \pm 174.1	2056.9 \pm 5.9
	medium	907.3 \pm 32.2	1000.0 \pm 90.4	1424.3 \pm 3.1	1425.0 \pm 2.9
	m-replay	969.1 \pm 21.9	978.4 \pm 39.6	944.6 \pm 28.9	1122.9 \pm 61.0
	m-expert	915.8 \pm 364.1	1132.9 \pm 282.2	738.5 \pm 250.2	1813.9 \pm 68.4
Half Cheetah	expert	1068.9 \pm 635.2	935.2 \pm 905.9	3637.0 \pm 80.9	4082.9 \pm 45.7
	medium	575.9 \pm 724.8	445.2 \pm 403.9	2690.0 \pm 92.4	2664.7 \pm 54.2
	m-replay	412.3 \pm 310.5	233.5 \pm 270.1	861.6 \pm 173.5	2855.0 \pm 242.2
	m-expert	-107.5 \pm 298.1	-275.9 \pm 544.5	1136.9 \pm 1608.3	3889.7 \pm 81.6

Table 13: Impact of alpha on returns for ComaDICE and baselines in MaMujoco.

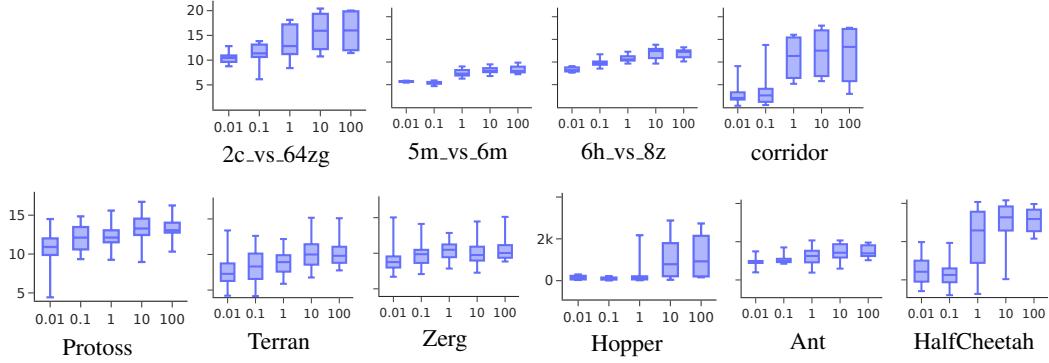


Figure 8: Impact of alpha on returns for ComaDICE and baselines.

Soft- χ^2 Divergence: This function combines the forms of KL and χ^2 divergences to mitigate both numerical instability and the dying gradient problem. It is defined piecewise as:

$$f_{\text{Soft-}\chi^2}(x) = \begin{cases} x \log x - x + 1 & \text{if } 0 < x < 1 \\ \frac{1}{2}(x-1)^2 & \text{if } x \geq 1 \end{cases}$$

The inverse derivative is:

$$(f'_{\text{Soft-}\chi^2})^{-1}(x) = \begin{cases} \exp(x) & \text{if } x < 0 \\ x + 1 & \text{if } x \geq 0 \end{cases}$$

This choice provides a stable optimization process by maintaining non-zero gradients and avoiding large exponential values, making it suitable for reinforcement learning tasks.

We assess their impact on both returns and winrates in environments such as SMACv1, SMACv2, and MaMujoco. Our results, detailed in Tables 16-20, reveal that the choice of f -divergence function significantly influences the algorithm's effectiveness. For instance, the Soft- χ^2 divergence consistently yields superior returns and competitive winrates across most scenarios, suggesting its robustness in managing distributional shifts in offline settings. Conversely, while Soft- χ^2 divergence also performs well, particularly in environments with higher complexity, KL divergence shows varying results, indicating its sensitivity to specific task dynamics. This comprehensive analysis underscores the importance of selecting an appropriate f -divergence function to optimize ComaDICE's performance in diverse multi-agent reinforcement learning contexts.

Instances		$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$
2c_vs_64zg	poor	0.0±0.0	0.0±0.0	0.0±0.0	0.6±1.3	0.6±1.3
	medium	0.0±0.0	1.9±3.8	5.0±5.1	8.8±7.0	8.8±4.6
	good	0.6±1.2	0.0±0.0	40.6±4.0	55.0±1.5	51.9±1.5
5m_vs_6m	poor	0.0±0.0	0.0±0.0	4.4±4.7	4.4±4.2	1.9±1.5
	medium	0.0±0.0	0.0±0.0	8.1±6.4	7.5±2.5	7.5±3.8
	good	0.0±0.0	0.0±0.0	6.2±4.4	8.1±3.2	10.0±6.1
6h_vs_8z	poor	0.0±0.0	0.0±0.0	1.9±3.8	1.9±3.8	0.6±1.3
	medium	0.0±0.0	0.6±1.3	1.9±1.5	3.1±2.0	3.1±2.0
	good	0.0±0.0	0.0±0.0	7.5±5.8	11.2±5.4	7.5±7.3
corridor	poor	0.0±0.0	0.6±1.2	0.0±0.0	0.6±1.3	1.2±1.5
	medium	0.0±0.0	0.0±0.0	30.0±5.1	27.3±3.4	34.4±2.8
	good	0.0±0.0	4.4±8.8	48.8±4.7	48.8±2.5	49.4±3.6

Table 14: Impact of alpha on winrates for ComaDICE and baselines in SMACv1.

Instances		$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 100$
Protoss	5_vs_5	20.6±10.0	31.9±6.1	50.0±2.8	46.2±6.1	46.2±8.5
	10_vs_10	19.4±6.1	25.0±3.4	45.0±11.1	50.6±8.7	51.2±7.6
	10_vs_11	0.0±0.0	6.2±9.7	18.8±8.1	20.0±4.2	29.4±8.3
	20_vs_20	1.2±1.5	8.8±7.8	28.1±8.6	47.5±7.8	40.6±6.2
	20_vs_23	0.0±0.0	1.9±2.5	9.4±6.6	13.8±5.8	17.5±5.1
Terran	5_vs_5	25.6±4.6	22.5±7.2	30.6±4.1	30.6±8.2	41.2±4.6
	10_vs_10	15.0±8.7	28.7±7.2	33.8±9.4	32.5±5.8	43.8±7.1
	10_vs_11	3.8±2.3	13.8±9.2	14.4±9.2	19.4±5.4	16.2±10.3
	20_vs_20	0.6±1.2	2.5±3.6	18.8±2.0	29.4±3.8	21.9±3.4
	20_vs_23	0.6±1.3	2.5±3.6	2.5±3.6	9.4±5.2	6.2±2.0
Zerg	5_vs_5	10.0±4.6	20.0±5.8	28.7±4.6	31.2±7.7	25.0±8.6
	10_vs_10	13.8±9.0	20.6±8.3	29.4±9.0	33.8±11.8	31.9±6.7
	10_vs_11	9.4±9.5	12.5±6.8	16.9±3.2	19.4±3.6	17.5±9.2
	20_vs_20	0.0±0.0	1.9±1.5	6.9±6.1	9.4±6.2	12.5±4.0
	20_vs_23	1.2±1.5	3.8±2.3	12.5±4.0	11.2±4.2	11.9±6.1

Table 15: Impact of alpha on winrates for ComaDICE and baselines in SMACv2.

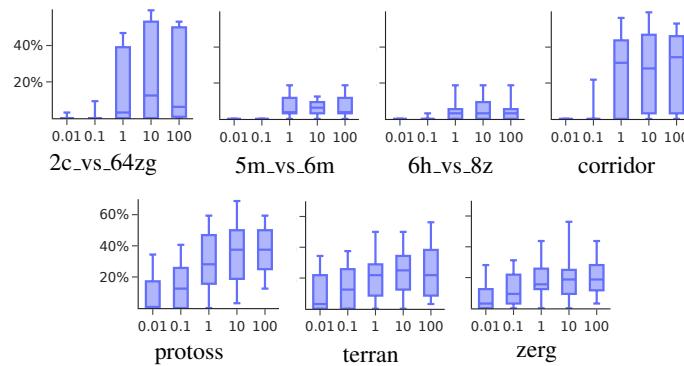


Figure 9: Impact of alpha on winrates for ComaDICE and baselines.

B.6.1 RETURNS

Instances		$f_{\chi^2}(x)$	$f_{KL}(x)$	$f_{Soft-\chi^2}(x)$
2c_vs_64zg	poor	11.6±0.2	11.1±0.3	12.1±0.5
	medium	16.1±0.6	15.7±0.3	16.3±0.7
	good	19.7±0.1	19.3±0.1	20.3±0.1
5m_vs_6m	poor	7.8±0.4	7.5±0.5	8.1±0.5
	medium	8.1±0.5	7.7±0.4	8.7±0.4
	good	8.7±0.6	8.1±0.4	8.7±0.5
6h_vs_8z	poor	10.5±0.3	10.0±0.2	11.4±0.6
	medium	12.9±0.4	12.4±0.5	12.8±0.2
	good	12.7±0.4	12.4±0.5	13.1±0.5
corridor	poor	6.5±0.5	6.1±0.4	6.4±0.5
	medium	12.7±0.7	12.0±0.7	12.9±0.6
	good	17.3±0.1	16.9±0.1	18.0±0.1

Table 16: Impact of f -divergence on returns for ComaDICE and baselines in SMACv1.

Instances		$f_{\chi^2}(x)$	$f_{KL}(x)$	$f_{Soft-\chi^2}(x)$
Protoss	5_vs_5	14.6±0.5	13.6±0.9	14.4±1.1
	10_vs_10	14.7±1.3	13.7±1.6	14.6±1.8
	10_vs_11	12.8±1.0	11.4±1.7	13.2±0.9
	20_vs_20	12.7±0.3	13.1±0.7	14.8±1.0
	20_vs_23	12.4±0.9	12.5±0.7	13.3±0.9
Terran	5_vs_5	11.1±1.2	12.7±2.0	10.7±1.5
	10_vs_10	9.8±0.9	10.7±1.3	11.8±0.9
	10_vs_11	8.9±0.8	8.9±1.0	9.4±0.9
	20_vs_20	10.5±0.5	10.2±0.7	11.8±0.5
	20_vs_23	8.2±0.4	7.4±0.7	8.2±0.7
Zerg	5_vs_5	10.0±0.8	9.6±1.5	10.7±2.0
	10_vs_10	12.4±1.2	10.3±1.1	11.5±1.0
	10_vs_11	8.9±0.4	9.1±1.1	11.0±0.9
	20_vs_20	9.0±0.8	9.0±0.6	9.4±1.2
	20_vs_23	10.2±1.0	9.3±0.8	10.5±0.8

Table 17: Impact of f -divergence on returns for ComaDICE and baselines in SMACv2.

Instances		$f_{\chi^2}(x)$	$f_{KL}(x)$	$f_{Soft-\chi^2}(x)$
Hopper	expert	2625.0±191.3	2018.7±972.0	2827.7±62.9
	medium	794.4±69.2	295.5±227.1	822.6±66.2
	m-replay	221.3±58.0	129.9±55.0	906.3±242.1
	m-expert	1294.1±520.4	105.5±103.9	1362.4±522.9
Ant	expert	1945.2±2.8	1884.1±27.8	2056.9±5.9
	medium	1359.2±3.2	1346.2±49.8	1425.0±2.9
	m-replay	1111.1±57.8	987.5±33.9	1122.9±61.0
	m-expert	1655.9±42.8	1182.5±405.1	1813.9±68.4
Half Cheetah	expert	3860.6±91.5	3830.0±88.8	4082.9±45.7
	medium	2532.3±81.9	2347.8±171.8	2664.7±54.2
	m-replay	2729.9±241.5	1258.5±1015.4	2855.0±242.2
	m-expert	3665.2±74.0	3601.0±155.6	3889.7±81.6

Table 18: Impact of f -divergence on returns for ComaDICE and baselines in MaMujoco.

B.6.2 WINRATES

Instances		$f_{\chi^2}(x)$	$f_{KL}(x)$	$f_{\text{Soft-}\chi^2}(x)$
2c_vs_64zg	poor	0.0±0.0	0.0±0.0	0.6±1.3
	medium	13.1±4.6	10.6±3.8	8.8±7.0
	good	55.6±3.1	54.4±1.5	55.0±1.5
5m_vs_6m	poor	3.8±3.1	3.8±3.6	4.4±4.2
	medium	6.2±2.8	5.0±3.8	7.5±2.5
	good	8.8±3.6	6.9±3.1	8.1±3.2
6h_vs_8z	poor	0.0±0.0	0.0±0.0	1.9±3.8
	medium	5.0±2.5	5.0±3.8	3.1±2.0
	good	9.4±4.4	9.4±2.0	11.2±5.4
corridor	poor	1.2±1.5	1.2±1.5	0.6±1.3
	medium	31.2±6.2	28.1±5.9	27.3±3.4
	good	49.4±5.4	48.1±1.5	48.8±2.5

Table 19: Impact of f -divergence on winrates for ComaDICE and baselines in SMACv1.

Instances		$f_{\chi^2}(x)$	$f_{KL}(x)$	$f_{\text{Soft-}\chi^2}(x)$
Protoss	5_vs_5	52.5±4.1	46.2±7.2	46.2±6.1
	10_vs_10	48.1±7.6	55.0±9.8	50.6±8.7
	10_vs_11	22.5±8.7	20.6±6.1	20.0±4.2
	20_vs_20	38.1±2.3	41.2±7.8	47.5±7.8
	20_vs_23	16.9±4.2	15.0±3.6	13.8±5.8
Terran	5_vs_5	41.2±7.2	38.8±10.6	30.6±8.2
	10_vs_10	30.6±4.1	36.2±10.8	32.5±5.8
	10_vs_11	15.6±11.5	15.0±7.5	19.4±5.4
	20_vs_20	33.8±6.4	28.7±11.8	29.4±3.8
	20_vs_23	5.6±4.1	8.1±4.2	9.4±5.2
Zerg	5_vs_5	29.4±9.0	33.1±13.3	31.2±7.7
	10_vs_10	31.2±7.7	26.2±5.1	33.8±11.8
	10_vs_11	11.2±1.5	16.2±7.2	19.4±3.6
	20_vs_20	7.5±3.2	11.2±7.0	9.4±6.2
	20_vs_23	10.6±3.2	10.0±2.3	11.2±4.2

Table 20: Impact of f -divergence on winrates for ComaDICE and baselines in SMACv2.

B.7 ABLATION STUDY: DIFFERENT TYPES OF MIXER NETWORK

In this section, we explore the impact of using different types of mixer networks within the ComaDICE algorithm. We introduce two settings for the mixer network within the ComaDICE algorithm: 1-layer and 2-layer settings. The mixer network plays a crucial role in aggregating local value functions into a global value function, which is essential for effective policy optimization in multi-agent reinforcement learning (MARL) settings. By examining various mixer network architectures, we aim to understand how these configurations affect the performance and stability of the ComaDICE algorithm. The comparisons are presented in Tables 21–25, reporting both average returns and win rates. The results clearly show that the 1-layer configuration outperforms the 2-layer configuration, delivering more stable training outcomes across nearly all tasks. This finding contrasts with many prior online MARL studies (Rashid et al., 2020; Son et al., 2019; Wang et al., 2020), which could be attributed to overfitting issues in the offline learning setting.

Since mixing networks are effective in capturing the interdependencies between local values and policies—reflecting credit assignment across local agents—the observed instability with the 2-layer mixing network suggests that this configuration may be too complex to effectively model the relationships between local agent policies in offline settings, leading to overfitting. While the

performance of the 2-layer mixing network might improve with more offline data, increasing the dataset size could overload storage capacity, making training computationally infeasible.

B.7.1 RETURNS

Instances		ComaDICE (ours)	
		1-layer	2-layer
2c_vs_64zg	poor	12.1±0.5	11.5±0.9
	medium	16.3±0.7	11.2±0.8
	good	20.3±0.1	9.0±2.2
5m_vs_6m	poor	8.1±0.5	3.8±1.1
	medium	8.7±0.4	0.8±0.3
	good	8.7±0.5	7.7±0.1
6h_vs_8z	poor	11.4±0.6	10.3±0.3
	medium	12.8±0.2	9.1±0.6
	good	13.1±0.5	8.3±0.5
corridor	poor	6.4±0.5	1.5±0.7
	medium	12.9±0.6	3.9±1.7
	good	18.0±0.1	2.6±2.3

Table 21: Average returns for ComaDICE and baselines on SMACv1 with different mixer settings.

Instances		ComaDICE (ours)	
		1-layer	2-layer
Protoss	5_vs_5	14.4±1.1	10.5±1.4
	10_vs_10	14.6±1.8	11.2±1.6
	10_vs_11	13.2±0.9	9.5±0.4
	20_vs_20	14.8±1.0	9.5±0.9
	20_vs_23	13.3±0.9	7.1±2.2
Terran	5_vs_5	10.7±1.5	8.3±0.8
	10_vs_10	11.8±0.9	8.8±1.1
	10_vs_11	9.4±0.9	6.4±1.2
	20_vs_20	11.8±0.5	7.8±0.9
	20_vs_23	8.2±0.7	6.6±0.9
Zerg	5_vs_5	10.7±2.0	7.8±1.1
	10_vs_10	11.5±1.0	9.7±0.6
	10_vs_11	11.0±0.9	7.9±0.7
	20_vs_20	9.4±1.2	7.8±0.6
	20_vs_23	10.5±0.8	8.0±0.5

Table 22: Average returns for ComaDICE and baselines on SMACv2 with different mixer settings.

Instances		ComaDICE (ours)	
		1-layer	2-layer
Hopper	expert	2827.7±62.9	483.7±349.7
	medium	822.6±66.2	648.4±245.9
	m-replay	906.3±242.1	441.9±260.8
	m-expert	1362.4±522.9	402.3±288.2
Ant	expert	2056.9±5.9	1583.0±160.4
	medium	1425.0±2.9	1198.9±53.9
	m-replay	1122.9±61.0	1041.8±38.4
	m-expert	1813.9±68.4	1426.6±171.4
Half Cheetah	expert	4082.9±45.7	2159.4±658.0
	medium	2664.7±54.2	2026.7±244.3
	m-replay	2855.0±242.2	1299.2±196.1
	m-expert	3889.7±81.6	1336.3±381.9

Table 23: Average returns for ComaDICE and baselines on MaMujoco with different mixer settings.

B.7.2 WINRATES

Instances		ComaDICE (ours)	
		1-layer	2-layer
2c_vs_64zg	poor	0.6±1.3	0.0±0.0
	medium	8.8±7.0	3.8±3.6
	good	55.0±1.5	19.4±5.0
5m_vs_6m	poor	4.4±4.2	3.1±0.0
	medium	7.5±2.5	1.2±1.5
	good	8.1±3.2	3.1±0.0
6h_vs_8z	poor	1.9±3.8	0.0±0.0
	medium	3.1±2.0	0.0±0.0
	good	11.2±5.4	1.9±2.5
corridor	poor	0.6±1.3	0.0±0.0
	medium	27.3±3.4	11.2±2.5
	good	48.8±2.5	23.1±8.1

Table 24: Average winrates for ComaDICE and baselines on SMACv1 with different mixer settings.

Instances		ComaDICE (ours)	
		1-layer	2-layer
Protoss	5_vs_5	46.2±6.1	31.9±3.6
	10_vs_10	50.6±8.7	32.5±5.8
	10_vs_11	20.0±4.2	10.6±7.3
	20_vs_20	47.5±7.8	21.9±4.0
	20_vs_23	13.8±5.8	6.9±5.4
Terran	5_vs_5	30.6±8.2	25.6±4.6
	10_vs_10	32.5±5.8	28.1±3.4
	10_vs_11	19.4±5.4	12.5±4.0
	20_vs_20	29.4±3.8	11.2±3.2
	20_vs_23	9.4±5.2	3.1±2.0
Zerg	5_vs_5	31.2±7.7	20.6±4.7
	10_vs_10	33.8±11.8	21.2±7.2
	10_vs_11	19.4±3.6	13.1±4.1
	20_vs_20	9.4±6.2	5.6±1.3
	20_vs_23	11.2±4.2	3.1±3.4

Table 25: Average winrates for ComaDICE and baselines on SMACv2 with different mixer settings.

B.8 COMADICE ON THE PENALTY XOR GAME

We discuss how ComaDICE addresses the Penalty XOR Game, a benchmark task previously considered in the AlberDICE paper (Matsunaga et al., 2023; Fu et al., 2022).

Overview of the Penalty XOR Game. The **Penalty XOR Game** is a commonly used benchmark in multi-agent cooperative reinforcement learning, designed to evaluate the agents’ ability to learn coordinated policies. In this game, two agents interact with a shared environment defined by a global state consisting of two binary features. Each agent selects a binary action, and the reward is determined by the relationship between their actions and the global state (as illustrated in Figure 10). This game highlights key challenges in multi-agent learning, such as credit assignment and coordination, as agents must infer the XOR-like reward logic from their experiences while aligning their actions to optimize joint behavior. This benchmark is particularly valuable for testing algorithms’ capabilities in capturing inter-agent dependencies and handling sparse, state-dependent rewards.

	<i>A</i>	<i>B</i>
<i>A</i>	0	1
<i>B</i>	1	−2

Figure 10: The Penalty XOR Game environment.

Experimental Setup. Following the setup in AlberDICE, we construct four datasets with increasing complexity: 1. **(a)** {AB} 2. **(b)** {AB, BA} 3. **(c)** {AA, AB, BA} 4. **(d)** {AA, AB, BA, BB}

Results. The optimal policy values returned by ComaDICE after a few epochs of training are presented in Table 26. Our results show that ComaDICE successfully learns the optimal policy across all four datasets. Compared to the results reported in the AlberDICE paper (Matsunaga et al., 2023), ComaDICE achieves similar policy values while outperforming other baselines considered in that study.

(a)	A	B	(b)	A	B	(c)	A	B	(d)	A	B
A	0.00	1.00									
B	0.00	0.00									

Table 26: Policy values after convergence returned by ComaDICE.

We now delve into the toy example to explain how ComaDICE achieves optimal policy values by balancing the maximization of global reward and the minimization of divergence between the occupancy of the learning policy and the behavior policy.

Consider the dataset {AB}, where the observation yields a high reward (i.e., 1). When optimizing the global policy with this dataset, ComaDICE seeks a policy that maximizes the reward across the dataset while aligning with the behavioral policy represented by {AB}. Consequently, it returns a global optimal policy (in the form of an occupancy ratio) that assigns the highest possible probabilities to the joint action {AB}. Subsequently, the weighted behavior cloning (BC) step learns decentralized policies that also assign the highest possible probabilities to the joint action {AB}, producing the desired optimal policy observed in our experiments.

For the dataset {AB, BA}, ComaDICE returns a global policy ensuring that the first player always chooses *A* and the second always chooses *B*. To understand why this occurs, note that ComaDICE’s learning objective consists of two terms: one aims to maximize the global reward, and the other minimizes the divergence between the learned policy and the dataset. When the dataset includes {AB, BA}, the occupancy-matching term favors a policy that assigns (uniformly) positive probabilities to both joint actions {AB} and {BA}. However, since ComaDICE learns decentralized policies, assigning significantly positive probabilities to both joint actions {AB} and {BA} implies that both players would take both actions *A* and *B* with significant probabilities, leading to a lower

expected global reward. In other words, exactly matching the dataset distribution would result in suboptimal reward. To optimize the overall objective, ComaDICE assigns the highest probability to one of the joint actions, $\{AB\}$ or $\{BA\}$. In our experiments, it assigned the highest probability to $\{AB\}$, achieving a better balance between reward maximization and divergence minimization. This explains why ComaDICE converges to this optimal policy.

The other datasets can be explained similarly. For example, with the dataset $\{AA, AB, BA\}$, the second term of the objective favors a policy that assigns equal probabilities (1/3) to these three joint actions. However, this would imply that both players take both actions A and B with non-zero and significant probabilities, resulting in lower accumulated rewards. To balance reward maximization and dataset alignment, ComaDICE returns an optimal policy ensuring that the first player always chooses A and the second always chooses B .

In comparison with OptDICE, both our experiments and those reported in the AlberDICE paper demonstrate that OptDICE fails to return optimal policy values even when provided with an optimal dataset, e.g., when the dataset is $\{AB, BA\}$. This is despite the fact that both OptDICE and ComaDICE aim to balance maximizing the joint reward and matching the data distribution. Here, we provide an intuitive explanation for why this occurs.

First, we note that while ComaDICE learns the global objective function over decentralized and factorized policies, OptDICE learns only the global policy by directly solving the original objective function. In this context, when the dataset is $\{AB, BA\}$, OptDICE learns a global policy that assigns uniform probabilities to both joint actions $\{AB\}$ and $\{BA\}$. However, when extracting local policies, OptDICE will return local policies that make both the first and second players choose actions A and B with probabilities of 0.25, as shown in Table 27, which is indeed suboptimal.

(b)	A	B
A	0.25	0.25
B	0.25	0.25

Table 27: Policy values returned by OptDICE with dataset (b).