COMADICE: OFFLINE COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING WITH STATIONARY DISTRIBUTION SHIFT REGULARIZATION

The Viet Bui
School of Computing and Information Systems
Singapore Management University, Singapore
`theviet.bui.2023@phdcs.smu.edu.sg`

Hong Thanh Nguyen
University of Oregon Eugene, Oregon
United States
`thanhhng@cs.orgeon.edu`

Tien Mai
School of Computing and Information Systems
Singapore Management University, Singapore
`atmai@smu.edu.sg`

ABSTRACT

Offline reinforcement learning (RL) has garnered significant attention for its ability to learn effective policies from pre-collected datasets without the need for further environmental interactions. While promising results have been demonstrated in single-agent settings, offline multi-agent reinforcement learning (MARL) presents additional challenges due to the large joint state-action space and the complexity of multi-agent behaviors. A key issue in offline RL is the *distributional shift*, which arises when the target policy being optimized deviates from the behavior policy that generated the data. This problem is exacerbated in MARL due to the interdependence between agents' local policies and the expansive joint state-action space. Prior approaches have primarily addressed this challenge by incorporating regularization in the space of either Q-functions or policies. In this work, we introduce a regularizer in the space of stationary distributions to better handle distributional shift. Our algorithm, ComaDICE, offers a principled framework for offline cooperative MARL by incorporating stationary distribution regularization for the global learning policy, complemented by a carefully structured multi-agent value decomposition strategy to facilitate multi-agent training. Through extensive experiments on the multi-agent *MuJoCo* and *StarCraft II* benchmarks, we demonstrate that ComaDICE achieves superior performance compared to state-of-the-art offline MARL methods across nearly all tasks.

1 INTRODUCTION

Over the years, deep RL has achieved remarkable success in various decision-making tasks (Levine et al., 2016; Silver et al., 2017; Kalashnikov et al., 2018; Haydari & Yılmaz, 2020). However, a significant limitation of deep RL is its need for millions of interactions with the environment to gather experiences for policy improvement. This process can be both costly and risky, especially in real-world applications like robotics and healthcare. To address this challenge, offline RL has emerged, enabling policy learning based solely on pre-collected demonstrations (Levine et al., 2020). Despite this advancement, offline RL faces a critical issue: the distribution shift between the offline dataset and the learned policy (Kumar et al., 2019). This distribution shift complicates value estimation for unseen states and actions during policy evaluation, resulting in extrapolation errors where out-of-distribution (OOD) state-action pairs are assigned unrealistic values (Fujimoto et al., 2018).

To tackle OOD actions, many existing works impose action-level constraints, either implicitly by regulating the learned value functions or explicitly through distance or divergence penalties (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Peng et al., 2019; Fujimoto & Gu, 2021; Xu et al., 2021). Only a few recent studies have addressed both OOD actions and states using state-action-level behavior constraints (Li et al., 2022; Zhang et al., 2022; Lee et al., 2021; 2022; Mao et al., 2024). In particular, there is an important line of work on DIstribution Correction Estimation (DICE) (Nachum & Dai, 2020) that constrains the distance in terms of the joint state-action occupancy measure between the learning policy and the offline policy. These DICE-based methods have demonstrated impressive performance results on the D4RL benchmarks (Lee et al., 2021; 2022; Mao et al., 2024).

It is important to note that that all the aforementioned offline RL approaches primarily focus on the single-agent setting. While multi-agent setting is prevalent in many real-world sequential decision-making tasks, offline MARL remains a relatively under-explored area. The multi-agent setting poses significantly greater challenges due to the large joint state-action space, which expands exponentially with the number of agents, as well as the inter-dependencies among the local policies of different agents. As a result, the offline data distribution can become quite sparse in these high-dimensional joint action spaces, leading to an increased number of OOD state-action pairs and exacerbating extrapolation errors. A few recent studies have sought to address the negative effects of sparse data distribution in offline MARL by adapting the well-known centralized training decentralized execution (CTDE) paradigm from online MARL (Oliehoek et al., 2008; Kraemer & Banerjee, 2016), enabling data-related regularization at the individual agent level. Notably, some of these works (Pan et al., 2022; Shao et al., 2024; Wang et al., 2022b) extend popular offline single-agent RL algorithms, such as CQL (Kumar et al., 2020) and SQL/EQL (Xu et al., 2023), within the CTDE framework.

In our work, we focus on addressing the aforementioned challenges in offline cooperative MARL. In particular, we follow the DICE approach to address both OOD states and actions, motivated by remarkable performance of recent DICE-based methods in offline single-agent RL. Similar to previous works in offline MARL, we adopt the CTDE framework to handle exponential joint state-action spaces in the multi-agent setting. We remark that extending the DICE approach under this CTDE framework is not straightforward given the complex objective of DICE that involves the f-divergence in stationary distribution between the learning joint policy and the behavior policy. Therefore, the value decomposition in CTDE needs to be carefully designed to ensure the consistency in optimality between the global and local policies. In particular, we provide the following main contributions:

- We propose ComaDICE, a new offline MARL algorithm that integrates DICE with a carefully designed value decomposition strategy. In ComaDICE, under the CTDE framework, we decompose both the global value function ν^{tot} and the global advantage functions A_{ν}^{tot}, rather than using Q-functions as in previous MARL works. This unique factorization approach allows us to theoretically demonstrate that the global learning objective in DICE is convex in local values, provided that the mixing network used in the value decomposition employs non-negative weights and convex activation functions. This significant finding ensures that our decomposition strategy promotes an efficient and stable training process.
- Building on our decomposition strategy, we demonstrate that finding an optimal global policy can be divided into multiple sub-problems, each aims to identify a local optimal policy for an individual agent. We provide a theoretical proof that the global optimal policy is, in fact, equivalent to the product of the local policies derived from these sub-problems.
- Finally, we conduct extensive experiments to evaluate the performance of our algorithm, ComaDICE, in complex MARL environments, including: multi-agent StarCraft II (i.e., SMACv1 (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2022)) and multi-agent Mujoco (de Witt et al., 2020) benchmarks. Our empirical results show that our ComaDICE outperforms several strong baselines in all these benchmarks.

2 RELATED WORK

Offline Reinforcement Learning (offline RL). Offline RL focuses on learning policies from pre-collected datasets without any further interactions with the environment (Levine et al., 2020; Prudencio et al., 2023). A significant challenge in offline RL is the issue of distribution shift, where unseen actions and states may arise during training and execution, leading to inaccurate policy evaluations and suboptimal outcomes. Consequently, there is a substantial body of literature addressing this challenge through various approaches (Prudencio et al., 2023). In particular, some studies impose explicit or implicit policy constraints to ensure that the learned policy remains close to the behavioral policy (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Kostrikov et al., 2021; Peng et al., 2019; Nair et al., 2020; Fujimoto & Gu, 2021; Xu et al., 2021; Cheng et al., 2024; Li et al., 2023). Others incorporate regularization terms into the learning objectives to mitigate the value overestimation on OOD actions (Kumar et al., 2020; Kostrikov et al., 2021; Xu et al., 2022c; Niu et al., 2022; Xu et al., 2023; Wang et al., 2022b). Uncertainty-based offline RL methods seek to balance conservative approaches with naive off-policy RL techniques, relying on estimates of model, value, or policy uncertainty (Agarwal et al., 2020; An et al., 2021; Bai et al., 2022). Offline model-based algorithms focus on conservatively estimating the transition dynamics and reward functions based on the pre-collected datasets (Kidambi et al., 2020; Yu et al., 2020; Matsushima et al., 2020; Yu et al., 2021). Some other methods impose action-level regularization through imitation learning techniques (Xu et al., 2022b; Chen et al., 2020; Zhang et al., 2023; Zheng et al., 2024; Brandfonbrener et al., 2021; Xu et al., 2022a). Finally, while a majority of previous works target OOD actions only, there are a few recent works attempt to address both OOD states and actions (Li et al., 2022; Zhang et al., 2022; Lee et al., 2021; 2022; Sikchi et al., 2023; Mao et al., 2024). Our work on offline MARL follow the DICE-based approach, as motivated by compelling performance of DICE-based algorithms in single-agent settings (Lee et al., 2021; 2022; Sikchi et al., 2023; Mao et al., 2024).

Offline Multi-agent Reinforcement Learning (offline MARL). While there is a substantial body of literature on offline single-agent RL, research on offline MARL remains limited. Offline MARL faces challenges from both distribution shift—characteristic of offline settings—and the exponentially large joint action space typical of multi-agent environments. Recent studies have begun to merge advanced methodologies from both offline RL and MARL to address these challenges (Yang et al., 2021; Pan et al., 2022; Shao et al., 2024; Wang et al., 2022b) Specifically, these works employ local policy regularization within the centralized training with decentralized execution (CTDE) framework to mitigate distribution shift. The CTDE paradigm, well-established in online MARL, facilitates more efficient and stable learning while allowing agents to operate in a decentralized manner (Oliehoek et al., 2008; Kraemer & Banerjee, 2016). For instance, Yang et al. (2021) utilize importance sampling to manage local policy learning on OOD samples. Both works by Pan et al. (2022) and Shao et al. (2024) are built upon CQL (Kumar et al., 2020), a prominent offline RL algorithm for single-agent scenarios. Matsunaga et al. (2023) developed AlberDICE, leveraging the Nash equilibrium solution concept from game theory to iteratively update the best responses of individual agents. Both AlberDICE and our method, ComaDICE, adopt the DICE framework to address the out-of-distribution (OOD) issue. However, while AlberDICE proposes learning individual Lagrange multipliers (or value functions) to obtain occupancy ratios, our ComaDICE algorithm learns a global value function by mixing local functions, adhering to the well-established CTDE principle. This design enables ComaDICE to better capture inter-agent relationships and improve credit assignment across local agents. Finally, OMIGA (Wang et al., 2022b) establishes the equivalence between global and local value regularization within a *policy constraint framework*, making it the current state-of-the-art algorithm in offline MARL. The key difference between ComaDICE and OMIGA lies in their respective approaches: OMIGA focuses on learning a global Q-function, whereas our algorithm (and other methods in the DICE family) operates in the occupancy space, aiming to learn the ratio between the occupancy of the learning policy and the behavior policy.

Beyond this main line of research, some studies formulate offline MARL as a sequence modeling problem, employing supervised learning techniques to tackle the issue (Meng et al., 2023; Tseng et al., 2022), while others adhere to decentralized approaches (Jiang & Lu, 2023).

3 PRELIMINARIES

Our work focuses on cooperative multi-agent RL, which can be modeled as a multi-agent Partially Observable Markov Decision Process (POMDP), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \mathcal{Z}, \mathcal{O}, n, \mathcal{N}, \gamma \rangle$. Here, n is number of agents, $\mathcal{N} = \{1, \ldots, n\}$ is the set of agents, $\mathbf{s} \in S$ represents the true state of the multi-agent environment, and $\mathcal{A} = \prod_{i \in \mathcal{N}} \mathcal{A}_i$ is the set of joint actions, where $\mathcal{A}_i$ is the set of individual actions available to agent $i \in \mathcal{N}$. At each time step, each agent $i \in \{1, 2, \ldots, n\}$ selects an action $a_i \in \mathcal{A}_i$, forming a joint action $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathcal{A}$. The transition dynamics $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ describe the probability of transitioning to the next state $\mathbf{s}'$ when agents take an action $\mathbf{a}$ from the current state $\mathbf{s}$. The discount factor $\gamma \in [0, 1)$ represents the weight given to future rewards. In a partially observable environment, each agent receives a local observation $s_i \in \mathcal{O}_i$ based on the observation function $\mathcal{Z}_i(\mathbf{s}) : \mathcal{S} \rightarrow \mathcal{O}_i$, and we denote the joint observation as $\mathbf{o} = (o_1, o_2, \ldots, o_n)$. In cooperative MARL, all agents share a global reward function $r(\mathbf{s}, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The goal of all agents is to learn a joint policy $\boldsymbol{\pi}_{\text{tot}} = \{\pi_1, \ldots, \pi_n\}$ that collectively maximize the expected discounted returns $\mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\boldsymbol{\pi}_{\text{tot}}}[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$. In the offline MARL setting, a pre-collected dataset $\mathcal{D}$ is obtained by sampling from a behavior policy $\mu_{\text{tot}} = \{\mu_1, \ldots, \mu_n\}$, and the policy learning is conducted soly based on $\mathcal{D}$, with no interactions with the environment. We also define the occupancy measure (or stationary distribution) as follows:

$$\rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}, \mathbf{a}) = (1 - \gamma) \sum\nolimits_{t=0}^{\infty} P(\mathbf{s}_t = \mathbf{s},\ \mathbf{a}_t = \mathbf{a})$$

which represents distribution visiting the pair (observation, action) $(\mathbf{s}_t, \mathbf{a}_1)$ when following the joint policy $\boldsymbol{\pi}_{tot}$, where $\mathbf{s}_0 \sim P_0,\ \mathbf{a}_t \sim \boldsymbol{\pi}_{tot}(\cdot|\mathbf{s}_t)$ and $\mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$.

4 COMADICE: OFFLINE COOPERATIVE MULTI-AGENT RL WITH STATIONARY DISTRIBUTION CORRECTION ESTIMATION

We consider an offline cooperative MARL problem where the goal is to optimize the expected discounted joint reward. In this work, we focus on the DICE objective function Nachum & Dai (2020); Lee et al. (2021), which incorporates a stationary distribution regularizer to capture the divergence between the occupancy measures of the learning policy, $\boldsymbol{\pi}_{tot}$, and the behavior policy, $\boldsymbol{\mu}_{tot}$, formulated as follows:

$$\max\nolimits_{\boldsymbol{\pi}_{tot}} \quad \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\pi}_{tot}}} \left[r(\mathbf{s}, \mathbf{a})\right] - \alpha D^f \left(\rho^{\boldsymbol{\pi}_{tot}} \parallel \rho^{\boldsymbol{\mu}_{tot}}\right) \tag{1}$$

where $D^f \left(\rho^{\boldsymbol{\pi}_{tot}} \parallel \rho^{\boldsymbol{\mu}_{tot}}\right) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\pi}_{tot}}} \left[f\left(\frac{\rho^{\boldsymbol{\pi}_{tot}}}{\rho^{\boldsymbol{\mu}_{tot}}}\right)\right]$ is the f-divergence between the stationary distribution $\rho^{\boldsymbol{\pi}_{tot}}$ of the learning policy and $\rho^{\boldsymbol{\mu}_{tot}}$ of the behavior policy. In this work, we consider $f(\cdot)$ to be strictly convex and differentiable. The parameter α controls the trade-off between maximizing the reward and penalizing deviation from the offline dataset's distribution (i.e., penalizing distributional shift). When $\alpha = 0$, the problem becomes the standard offline MARL, where the objective is to find a joint policy that maximizes the expected joint reward. On the other hand, when $\alpha \gg 1$, the problem shifts towards imitation learning, aiming to closely mimic the behavioral policy.

This DICE-based approach offers the advantage of better capturing the system dynamics inherent in the offline data. Such stationary distributions, $\rho^{\boldsymbol{\pi}_{tot}}$ and $\rho^{\boldsymbol{\mu}_{tot}}$, however, are not directly available. We will discuss how to estimate them in the next subsection.

4.1 CONSTRAINED OPTIMIZATION IN THE STATIONARY DISTRIBUTION SPACE

We first formulate the learning problem in Eq. 1 as a constrained optimization on the space of $\rho^{\boldsymbol{\pi}_{tot}}$:

$$\max\nolimits_{\rho^{\boldsymbol{\pi}_{tot}}} \quad \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\pi}_{tot}}} \left[r(\mathbf{s}, \mathbf{a})\right] - \alpha D^f \left(\rho^{\boldsymbol{\pi}_{tot}} \parallel \rho^{\boldsymbol{\mu}_{tot}}\right) \tag{2}$$

$$s.t. \quad \sum\nolimits_{\mathbf{a}'} \rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}, \mathbf{a}') = (1 - \gamma)p_0(\mathbf{s}) + \gamma \sum\nolimits_{\mathbf{a}',\mathbf{s}'} \rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}', \mathbf{a}')P(\mathbf{s}|\mathbf{a}', \mathbf{s}'),\ \forall \mathbf{s} \in \mathcal{S}. \tag{3}$$

When f is convex, (2-3) becomes a convex optimization problem, as it involves maximizing a concave objective function subject to linear constraints. We now consider the Lagrange dual of (2-3):

$$\begin{aligned} \mathcal{L}(\nu^{tot}, \rho^{\boldsymbol{\pi}_{tot}}) &= \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\pi}_{tot}}} \left[r(\mathbf{s}, \mathbf{a})\right] - \alpha \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\mu}_{tot}}} \left[f\left(\frac{\rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}, \mathbf{a})}{\rho^{\boldsymbol{\mu}_{tot}}(\mathbf{s}, \mathbf{a})}\right)\right] \\ &\quad - \sum\nolimits_{\mathbf{s}} \nu^{tot}(\mathbf{s}) \Big(\sum\nolimits_{\mathbf{a}'} \rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}, \mathbf{a}') - (1 - \gamma)p_0(\mathbf{s}) - \gamma \sum\nolimits_{\mathbf{a}',\mathbf{s}'} \rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s}', \mathbf{a}')P(\mathbf{s}|\mathbf{a}', \mathbf{s}')\Big), \end{aligned} \tag{4}$$

where $\nu^{tot}(\mathbf{s})$ is a Lagrange multiplier. Since (2-3) is a convex optimization problem, it is equivalent to the following minimax problem over the spaces of ν^{tot} and $\rho^{\boldsymbol{\pi}_{tot}}$: $\min_{\nu^{tot}} \max_{\rho^{\boldsymbol{\pi}_{tot}}} \{\mathcal{L}(\nu^{tot}, \rho^{\boldsymbol{\pi}_{tot}})\}$. Furthermore, we observe that $\mathcal{L}(\nu^{tot}, \rho^{\boldsymbol{\pi}_{tot}})$ is linear in ν^{tot} and concave in $\rho^{\boldsymbol{\pi}_{tot}}$, so the minimax problem has a saddle point, implying: $\min_{\nu^{tot}} \max_{\rho^{\boldsymbol{\pi}_{tot}}} \{\mathcal{L}(\nu^{tot}, \rho^{\boldsymbol{\pi}_{tot}})\} = \max_{\rho^{\boldsymbol{\pi}_{tot}}} \min_{\nu^{tot}} \{\mathcal{L}(\nu^{tot}, \rho^{\boldsymbol{\pi}_{tot}})\}$. In a manner analogous to the single-agent case (Lee et al., 2021), by defining $w_{\nu}^{tot}(\mathbf{s}, \mathbf{a}) = \frac{\rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s},\mathbf{a})}{\rho^{\boldsymbol{\mu}_{tot}}(\mathbf{s},\mathbf{a})}$, the Lagrange dual function can be simplified into the more compact form (with detailed derivations are in the appendix):

$$\mathcal{L}(\nu^{tot}, w^{tot}) = (1 - \gamma)\mathbb{E}_{\mathbf{s}\sim p_0}[\nu^{tot}(\mathbf{s})] + \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\mu}_{tot}}} \left[-\alpha f\left(w_{\nu}^{tot}(\mathbf{s}, \mathbf{a})\right) + w_{\nu}^{tot}(\mathbf{s}, \mathbf{a})A_{\nu}^{tot}(\mathbf{s}, \mathbf{a})\right],$$

where A_ν^{tot} is an "advantage function" defined based on ν^{tot} as:

$$A_\nu^{tot}(\mathbf{s},\mathbf{a}) = q^{tot}(\mathbf{s},\mathbf{a}) - \nu^{tot}(\mathbf{s}), \tag{5}$$

with $q^{tot}(\mathbf{s},\mathbf{a}) = r(\mathcal{Z}(\mathbf{s}),\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})}[\nu^{tot}(\mathbf{s}')]$. It is important to note that $\nu^{tot}(\mathbf{s})$ and $q^{tot}(\mathbf{s},\mathbf{a})$ can be interpreted as a value function and a Q function, respectively, arising from the decomposition of the stationary distribution regularizer. We can now write the learning problem as follows:

$$\min_{\nu^{tot}} \max_{w^{tot}\geq 0} \ \{\mathcal{L}(\nu^{tot}, w^{tot})\}. \tag{6}$$

It can be observed that $\mathcal{L}(\nu^{tot}, w^{tot})$ is linear in ν^{tot} and concave in w^{tot}, which ensures well-behaved properties in both the ν^{tot}- and w^{tot}-spaces. Following the derivations in Lee et al. (2021) , a key feature of the above minimax problem is that the inner maximization problem has a closed-form solution, which greatly simplifies the minimax problem, making it no longer adversarial. We formalize this result as follows:

Proposition 4.1. *The minimax problem in Eq. 6 is equivalent to* $\min_{\nu^{tot}} \ \{\widetilde{\mathcal{L}}(\nu^{tot})\}$, *where*

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

Here, 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 $\widetilde{\mathcal{L}}(\nu^{tot})$ w.r.t. θ is given as follows:

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

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

Proposition 4.1 above is a direct extension of the formulations in Lee et al. (2021) developed for the single-agent setting, differing only in the inclusion of the closed-form expression for the first-order derivative of the objective function, $\widetilde{\mathcal{L}}(\nu^{tot})$.

4.2 VALUE FACTORIZATION

Directly optimizing $\min_{\nu^{tot}} \ \{\mathcal{L}(\nu^{tot}, w_\nu^{tot*})\}$ in multi-agent settings is generally impractical due to the large state and action spaces. Therefore, we follow the idea of value decomposition in the well-known CTDE framework in cooperative MARL to address this computational challenge. However, it is not straightforward to extend the DICE approach within this CTDE framework due to the complex objective of DICE, which involves the f-divergence between the learned joint policy and the behavior policy in stationary distributions. Thus, it is crucial to carefully design the value decomposition in CTDE to ensure optimality consistency between the global and local policies.

Specifically, we adopt a factorization approach that decomposes the value function $\nu^{tot}(\mathbf{s})$ (or global Lagrange multipliers) into local values using mixing network architectures. Let $\boldsymbol{\nu}(\mathbf{s}) = \{\nu_1(s_1),\ldots,\nu_n(s_n)\}$ represent a collection of local "value functions" and let $\mathbf{A}_{\boldsymbol{\nu}}(\mathbf{s},\mathbf{a}) = \{A_i(s_i,a_i),\ i = 1,...,n\}$ represent a collection of local advantage functions. The local advantage functions are computed as $A_i(s_i,a_i) = q_i(s_i,a_i) - \nu_i(s_i)$ for all $i \in \mathcal{N}$, where $\mathbf{q}(\mathbf{s},\mathbf{a}) = \{q_i(s_i,a_i),\ i = 1,...,n\}$ is a vector of local Q functions. To facilitate centralized learning, we create a mixing network, $\mathcal{M}_\theta$, where θ are the learnable weights, that aggregates the local values to form the global value and advantage functions as follows:

$$\nu^{tot}(\mathbf{s},\mathbf{a}) = \mathcal{M}_\theta[\boldsymbol{\nu}(\mathbf{s})], \quad A_\nu^{tot}(\mathbf{s},\mathbf{a}) = \mathcal{M}_\theta[\mathbf{q}(\mathbf{s},\mathbf{a}) - \boldsymbol{\nu}(\mathbf{s})],$$

where each network takes the vectors $\boldsymbol{\nu}(\mathbf{s})$ or $\mathbf{A}_{\boldsymbol{\nu}}(\mathbf{s},\mathbf{a})$ as inputs and outputs ν^{tot} and A_ν^{tot}, respectively. Under this architecture, the learning objective becomes:

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

with the observation that $\mathbf{A}_\nu(\mathbf{s},\mathbf{a})$ can be expressed as a linear function of $\boldsymbol{\nu}$. There are different ways to construct the mixing network $\mathcal{M}_\theta$; previous work often employs a single linear combination (1-layer network) or a two-layer network with convex activations such as ReLU, ELU, or Maxout. In the following, we show a general result stating that the learning objective function is convex in $\boldsymbol{\nu}$, provided that the mixing network is constructed with nonnegative weights and convex activations.

Theorem 4.2. *If the mixing network $\mathcal{M}_\theta[\cdot]$ is constructed with non-negative weights and convex activations, then $\widetilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ is convex in $\boldsymbol{\nu}$.*

Mixing networks with non-negative weights and concave activations (e.g., ELU or ReLU) have been extensively used in MARL, forming the foundation of several notable state-of-the-art algorithms such as QMIX (Rashid et al., 2020), QTRAN (Son et al., 2019), and MFIQ (Bui et al., 2024). In particular, it has been demonstrated that mixing networks with either negative weights or non-concave activations result in significantly degraded performance (Bui et al., 2024). Theorem 4.2 shows that $\widetilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ is convex in $\boldsymbol{\nu}$ when using *any multi-layer feed-forward mixing networks with non-negative weights and convex activation functions*. This finding is highly general and non-trivial, given the nonlinearity and complexity of both the function (in terms of $\boldsymbol{\nu}$) and the mixing networks. Previous work has often focused on single-layer (Wang et al., 2022b) or two-layer mixing structures (Rashid et al., 2020; Bui et al., 2024), emphasizing that such two-layer networks can approximate any monotonic function arbitrarily closely as network width approaches infinity (Dugas et al., 2009). In our experiments, we test two configurations for the mixing network: a linear combination (or 1-layer) and a 2-layer feed-forward network. While 2-layer mixing structures have shown strong performance in online MARL (Rashid et al., 2020; Son et al., 2019; Wang et al., 2020), we observe in our offline settings that the linear combination approach provides more stable results.

4.3 POLICY EXTRACTION

Let $\boldsymbol{\nu}^*$ be an optimal solution to the training problem with mixing networks, i.e.,

$$\min_{\boldsymbol{\nu},\theta} \widetilde{\mathcal{L}}(\boldsymbol{\nu}, \theta). \tag{7}$$

We now need to extract a local and joint policy from this solution. Based on Prop. 4.1, given $\boldsymbol{\nu}^*$, we can compute this occupancy ratio as follows: : $w^{tot*}(\mathbf{s}, \mathbf{a}) = \max\left\{0, f'^{-1}\left(\frac{\mathcal{M}_\theta[\mathbf{A}_{\boldsymbol{\nu}^*}(\mathbf{s},\mathbf{a})]}{\alpha}\right)\right\}$. The global policy can then be obtained as follows: $\boldsymbol{\pi}^*_{tot}(\mathbf{a}|\mathbf{s}) = \frac{w^{tot*}(\mathbf{s},\mathbf{a})\cdot\rho^{\boldsymbol{\mu}_{tot}}(\mathbf{s},\mathbf{a})}{\sum_{\mathbf{a}'\in\mathcal{A}} w^{tot*}(\mathbf{s},\mathbf{a}')\cdot\rho^{\boldsymbol{\mu}_{tot}}(\mathbf{s},\mathbf{a}')}$. This computation, however, is not practical since $\rho^{\boldsymbol{\mu}_{tot}}$ is generally not available and might not be accurately estimated in the offline setting. A more practical way to estimate the global policy, $\boldsymbol{\pi}^*_{tot}$, as the result of solving the following weighted behavioral cloning (BC):

$$\max_{\boldsymbol{\pi}_{tot}\in\Pi_{tot}} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\pi}^*_{tot}}}[\log\boldsymbol{\pi}_{tot}(\mathbf{a}|\mathbf{s})] = \max_{\boldsymbol{\pi}_{tot}\in\Pi_{tot}} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\rho^{\boldsymbol{\mu}_{tot}}}[w^{tot*}(\mathbf{s}, \mathbf{a})\log\boldsymbol{\pi}_{tot}(\mathbf{a}|\mathbf{s})], \tag{8}$$

where Π_{tot} represents the feasible set of global policies. Here we assume that Π_{tot} contains decomposable global policies, i.e., $\Pi_{tot} = \{\boldsymbol{\pi}_{tot} \mid \exists\pi_i,\ \forall i \in \mathcal{N} \text{ such that } \boldsymbol{\pi}_{tot}(\mathbf{a}|\mathbf{s}) = \prod_{i\in\mathcal{N}}\pi_i(a_i|s_i)\}$. In other words, Π_{tot} consists of global policies that can be expressed as a product of local policies. This decomposability is highly useful for decentralized learning and has been widely adopted in MARL (Wang et al., 2022b; Bui et al., 2024; Zhang et al., 2021).

While the above weighted BC appears practical, as $(\mathbf{s}, \mathbf{a})$ can be sampled from the offline dataset generated by $\rho^{\boldsymbol{\pi}_{tot}}$, and since $w^{tot*}(\mathbf{s}, \mathbf{a})$ is available from solving 7, it does not directly yield local policies, which are essential for decentralized execution. To address this, we propose solving the following weighted BC for each local agent $i \in \mathcal{N}$:

$$\max\nolimits_{\pi_i} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[w^{tot*}(\mathbf{s}, \mathbf{a})\log\pi_i(a_i|s_i)\right]. \tag{9}$$

This local WBC approach has several attractive properties. First, $w^{tot*}(\mathbf{s}, \mathbf{a})$ appears explicitly in the local policy optimization and is computed from global observations and actions. This enables local policies to be optimized with global information, ensuring consistency with the credit assignment in the multi-agent system. Furthermore, as shown in Proposition 4.3 below, the optimization of local policies through local WBC is highly consistent with the global weighted BC in 8.

Proposition 4.3. *Let π^*_i be the optimal solution to the local weighted BC 9. Then $\boldsymbol{\pi}^*_{tot}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i\in\mathcal{N}}\pi^*_i(a_i|s_i)$ is also optimal for the global weighted BC in 8.*

Here we note that consistency between global and local policies is a critical aspect of centralized training with CTDE. Previous MARL approaches typically achieve this by factoring Q or V functions into local functions and training local policies based on these local functions (Rashid et al., 2020; Wang et al., 2020; Bui et al., 2024). However, in our case, there are key differences that prevent us from employing such local values to derive local policies. Specifically, we factorize the Lagrange multipliers ν^{tot} to train the stationary distribution ratio w^{tot}. Although local w values can be extracted from local ν_i, these local w values do not represent a local stationary distribution ratio and therefore cannot be used to recover local policies.

5 PRACTICAL ALGORITHM

Let $\mathcal{D}$ represent the offline dataset, consisting of sequences of local observations and actions gathered from a global behavior policy $\boldsymbol{\pi}_{tot}$. To train the value function $\boldsymbol{\nu}$, we construct a value network $\nu_i(s_i; \psi_\nu)$ for each local agent i, along with a network for each local Q-function $q_i(s_i, a_i; \psi_q)$, where ψ_ν and ψ_q are learnable parameters for the local value and Q-functions. We note that the introduction and learning of the Q-functions are intended to facilitate the decomposition of the advantage function, A_ν^{tot}. In our multi-agent setting, the absence of local rewards makes it difficult to directly compute local advantage functions. To overcome this challenge, we learn local Q-functions, which are then used to derive the local advantage functions. Additionally, as explained below, a MSE is optimized to ensure that the global Q-function and state-value function align properly with the global rewards.

Now, each local advantage function is then calculated as follows: The global value function and advantage function are subsequently aggregated using two mixing networks with a shared set of learnable parameters θ:

$$\nu^{tot}(\mathbf{s}) = \mathcal{M}_\theta^{\mathbf{s}}[\boldsymbol{\nu}(\mathbf{s}; \psi_\nu)], \quad A_\nu^{tot}(\mathbf{s}, \mathbf{a}) = \mathcal{M}_\theta^{\mathbf{s}}[\mathbf{q}(\mathbf{s}, \mathbf{a}; \psi_q) - \boldsymbol{\nu}(\mathbf{s}; \psi_\nu)],$$

where $\mathcal{M}_\theta^{\mathbf{s}}[\cdot]$ represents a linear combination of its inputs with non-negative weights, such that $\mathcal{M}_\theta^{\mathbf{s}}[\boldsymbol{\nu}(\mathbf{s}; \psi_\nu)] = \boldsymbol{\nu}(\mathbf{s}; \psi_\nu)^\top W_\theta^{\mathbf{s}} + b_\theta^{\mathbf{s}}$, where $W_\theta^{\mathbf{s}}$ and $b_\theta^{\mathbf{s}}$ are weights of the mixing network.[1] It is important to note that $W_\theta^{\mathbf{s}}$ and $b_\theta^{\mathbf{s}}$ are generated by hyper-networks that take the global state $\mathbf{s}$ and the learnable parameters θ as inputs. In this context, we employ the same mixing network $\mathcal{M}_\theta^{\mathbf{s}}$ to combine the local values and advantages. However, our framework is flexible enough to allow the use of two different mixing networks for ν^{tot} and A_ν^{tot}.

In our setting, the relationship between the global Q-function, value, and advantage functions is described in Eq. 5. Specifically, we have: $A_\nu^{tot}(\mathbf{s}, \mathbf{a}) = r(\mathcal{Z}(\mathbf{s}), \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}[\nu^{tot}(\mathbf{s}')] - \nu^{tot}(\mathbf{s})$. To capture this relationship, we train the Q-function by optimizing the following MSE loss:

$$\min_{\mathbf{q}} \sum\nolimits_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim \mathcal{D}} \left(A_\nu^{tot}(\mathbf{s}, \mathbf{a}) - r(\mathcal{Z}(\mathbf{s}), \mathbf{a}) + \gamma \nu^{tot}(\mathbf{s}') - \nu^{tot}(\mathbf{s})\right)^2.$$

This is equivalent to:

$$\begin{aligned}\min_{\psi_q} \mathcal{L}_q(\psi_q) = \sum\nolimits_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim \mathcal{D}} \Big(&\mathcal{M}_\theta^{\mathbf{s}}[\mathbf{q}(\mathbf{s}, \mathbf{a}; \psi_q) - \boldsymbol{\nu}(\mathbf{s}; \psi_\nu)] \\ &- r(\mathcal{Z}(\mathbf{s}), \mathbf{a}) + \gamma \mathcal{M}_\theta^{\mathbf{s}'}[\boldsymbol{\nu}(\mathbf{s}'; \psi_\nu)] - \mathcal{M}_\theta^{\mathbf{s}}[\boldsymbol{\nu}(\mathbf{s}; \psi_\nu)]\Big)^2. \end{aligned} \tag{10}$$

For the primary loss function used to train the value function, we leverage transitions from the offline dataset to approximate the objective $\widetilde{\mathcal{L}}$, resulting in the following loss function for offline training:

$$\widetilde{\mathcal{L}}(\psi_\nu, \theta) = (1-\gamma)\mathbb{E}_{\mathbf{s}_0 \sim \mathcal{D}}[\mathcal{M}_\theta^{\mathbf{s}_0}[\boldsymbol{\nu}(\mathbf{s}_0; \psi_\nu)]] + \mathbb{E}_{(\mathbf{s},\mathbf{a}) \sim \mathcal{D}}\left[\alpha f^*\!\left(\frac{\mathcal{M}_\theta^{\mathbf{s}}[\mathbf{q}(\mathbf{s}, \mathbf{a}; \psi_q) - \boldsymbol{\nu}(\mathbf{s}; \psi_\nu)]}{\alpha}\right)\right]. \tag{11}$$

As mentioned, after obtaining $(\boldsymbol{\nu}^*, \theta^*)$ by solving $\min_{\psi_\nu, \theta} \widetilde{\mathcal{L}}(\psi_\nu, \theta)$, we compute the occupancy ratio: $w_\nu^{tot*}(\mathbf{s}, \mathbf{a}) = \max\left\{0, f'^{-1}\left(\frac{\mathcal{M}_{\theta^*}^{\mathbf{s}}[\boldsymbol{\nu}^*(\mathbf{s})] - \mathcal{M}_{\theta^*}^{\mathbf{s}}[\mathbf{q}(\mathbf{s},\mathbf{a};\psi_q)]}{\alpha}\right)\right\}$. To train the local policy $\pi_i(a_i|s_i)$, we represent it using a policy network $\pi_i(a_i|s_i; \eta_i)$, where η_i are the learnable parameters. The training process involves optimizing the following weighted behavioral cloning (BC) objective:

$$\max_{\eta_i} \;\; \mathcal{L}_\pi(\eta_i) = \sum\nolimits_{(\mathbf{s},\mathbf{a}) \sim \mathcal{D}} w_\nu^{tot*}(\mathbf{s}, \mathbf{a}) \log(\pi_i(a_i|s_i; \eta_i)). \tag{12}$$

Our ComaDICE algorithm consists of two primary steps. The first step involves estimating the occupancy ratio w^{tot*} from the offline dataset. The second step focuses on training the local policy

---

[1] In our experiments, we use a single-layer mixing network due to its superior performance compared to a two-layer structure, though our approach is general and can handle any multi-layer feed-forward mixing network.

by solving the weighted BC problem using w^{tot*}. In the first step, we simultaneously update the Q-functions ψ_q, the mixing network parameters θ, and the value function ψ_ν, aiming to minimize the mean squared error (MSE) in Eq. 10 while optimizing the main loss function in Eq. 11.

It is important to note that, in practical POMDP scenarios, the global state $\mathbf{s}$ is not directly accessible during training and is instead represented by the joint observations $\mathbf{o}$ from the agents. For notational convenience, we use the global state $\mathbf{s}$ in our formulation; however, in practice, it corresponds to the joint observation $\mathcal{Z}(\mathbf{s})$. Specifically, terms like $\rho^{\boldsymbol{\mu}^{tot}}(\mathbf{s}, \mathbf{a})$ and $\nu^{tot}(\mathbf{s})$ actually refer to $\rho^{\boldsymbol{\mu}^{tot}}(\mathbf{o}, \mathbf{a})$ and $\nu^{tot}(\mathbf{o})$, where $\mathbf{o} = \mathcal{Z}(\mathbf{s})$.

6 EXPERIMENTS

6.1 ENVIRONMENTS

We utilize three standard MARL environments: SMACv1 (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2022), and Multi-Agent MuJoCo (MaMujoco) (de Witt et al., 2020), each offering unique challenges and configurations for evaluating cooperative MARL algorithms.

SMACv1. SMACv1 is based on Blizzard's StarCraft II. It uses the StarCraft II API and DeepMind's PySC2 to enable agent interactions with the game. SMACv1 focuses on decentralized micromanagement scenarios where each unit is controlled by an RL agent. Tasks like *2c_vs_64zg* and *5m_vs_6m* are labeled hard, while *6h_vs_8z* and *corridor* are super hard. The offline dataset, provided by Meng et al. (2023), was generated using MAPPO-trained agents (Yu et al., 2022).

SMACv2. In comparison to SMACv1, SMACv2 introduces increased randomness and diversity by randomizing start positions, unit types, and modifying sight and attack ranges. This version includes tasks such as *protoss*, *terran*, and *zerg*, with instances ranging from *5_vs_5* to *20_vs_23*, increasing in difficulty. Our offline dataset for SMACv2 was generated by running MAPPO for 10 million training steps and collecting 1,000 trajectories, ensuring medium quality but comprehensive coverage of the learning process. To the best of our knowledge, we are the first to explore SMACv2 in offline MARL, whereas most prior work has used this environment in online settings.

MaMujoco. MaMujoco serves as a benchmark for continuous cooperative multi-agent robotic control. Derived from the single-agent MuJoCo control suite in OpenAI Gym (Brockman et al., 2016), it presents scenarios where multiple agents within a single robot must collaborate to achieve tasks. The tasks include *Hopper-v2*, *Ant-v2*, and *HalfCheetah-v2*, with instances labeled as *expert*, *medium*, *medium-replay*, and *medium-expert*. The offline dataset was created by (Wang et al., 2022b) using the HAPPO method (Wang et al., 2022a).

6.2 BASELINES

We consider the following baselines, which represent either standard or state-of-the-art (SOTA) methods for offline MARL: (i) **BC** (Behavioral Cloning); (ii) **BCQ** (Batch-Constrained Q-learning) (Fujimoto et al., 2019) – an offline RL algorithm that constrains the policy to actions similar to those in the dataset to reduce distributional shift, adapted for offline MARL settings; (iii) **CQL** (Conservative Q-Learning) (Kumar et al., 2020) – a method that stabilizes offline Q-learning by penalizing out-of-distribution actions, ensuring conservative value estimates; (iv) **ICQ** (Implicit Constraint Q-learning) (Yang et al., 2021) – an approach using importance sampling to manage out-of-distribution actions in multi-agent settings; (v) **OMAR** (Offline MARL with Actor Rectification) (Pan et al., 2022) – a method combining CQL with optimization techniques to ensure the global validity of local regularizations, promoting cooperative behavior; (vi) **OMIGA** (Offline MARL with Implicit Global-to-Local Value Regularization) (Wang et al., 2022b) – a SOTA method that transforms global regularizations into implicit local ones, optimizing local policies with global insights; (vii) **OptDICE** - a naive extension of the OptDICE algorithm Lee et al. (2021) to multi-agent settings where the global value function are directly learned without value factorization; and (viii) **AlberDICE** Matsunaga et al. (2023) - an offline MARL algorithm which also leverages the DICE framework to address the OOD.

We used experimental results contributed by the authors of OMIGA (Wang et al., 2022b) as our baselines. They provided both the results and source code for all the baseline methods. This source

Instances		**BC**	**BCQ**	**CQL**	**ICQ**	**OMAR**	**OMIGA**	**OptDICE**	**AlberDICE**	**ComaDICE** (ours)
2c_vs_64zg	poor	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**0.6 ± 1.3**
	medium	1.9 ± 1.5	2.5 ± 3.6	2.5 ± 3.6	1.9 ± 1.5	1.2 ± 1.5	6.2 ± 5.6	1.0 ± 1.5	1.6 ± 1.6	**8.8 ± 7.0**
	good	31.2 ± 9.9	35.6 ± 8.8	44.4 ± 13.0	28.7 ± 4.6	28.7 ± 9.1	40.6 ± 9.5	37.5 ± 3.1	42.2 ± 6.4	**55.0 ± 1.5**
5m_vs_6m	poor	2.5 ± 1.3	1.2 ± 1.5	1.2 ± 1.5	1.2 ± 1.5	0.6 ± 1.2	**6.9 ± 1.2**	0.0 ± 0.0	0.0 ± 0.0	4.4 ± 4.2
	medium	1.9 ± 1.5	1.2 ± 1.5	2.5 ± 1.2	1.2 ± 1.5	0.6 ± 1.2	2.5 ± 3.1	0.0 ± 0.0	3.1 ± 0.0	**7.5 ± 2.5**
	good	2.5 ± 2.3	1.9 ± 2.5	1.9 ± 1.5	3.8 ± 2.3	3.8 ± 1.2	6.9 ± 1.2	7.3 ± 3.9	3.9 ± 1.4	**8.1 ± 3.2**
6h_vs_8z	poor	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 1.5	**1.9 ± 3.8**
	medium	1.9 ± 1.5	1.9 ± 1.5	1.9 ± 1.5	2.5 ± 1.2	1.9 ± 1.5	1.2 ± 1.5	0.0 ± 0.0	2.3 ± 2.6	**3.1 ± 2.0**
	good	8.8 ± 1.2	8.8 ± 3.6	7.5 ± 1.5	9.4 ± 2.0	0.6 ± 1.3	5.6 ± 3.6	0.0 ± 0.0	0.0 ± 0.0	**11.2 ± 5.4**
corridor	poor	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**0.6 ± 1.3**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**0.6 ± 1.3**
	medium	15.0 ± 2.3	23.1 ± 1.5	14.4 ± 1.5	22.5 ± 3.1	11.9 ± 2.3	23.8 ± 5.1	19.8 ± 2.9	9.4 ± 6.8	**27.3 ± 3.4**
	good	30.6 ± 4.1	42.5 ± 6.4	5.6 ± 1.2	42.5 ± 6.4	3.1 ± 0.0	41.9 ± 6.4	39.6 ± 5.3	43.1 ± 6.4	**48.8 ± 2.5**

Table 1: Comparison of average winrates for ComaDICE and baselines on SMACv1 tasks.

Instances		**BC**	**BCQ**	**CQL**	**ICQ**	**OMAR**	**OMIGA**	**OptDICE**	**AlberDICE**	**ComaDICE** (ours)
Protoss	5_vs_5	36.9±8.7	16.2±2.3	10.0±4.1	36.9±9.1	21.2±4.1	33.1±5.4	10.8±1.2	12.6±0.9	**46.2±6.1**
	10_vs_10	36.2±10.6	9.4±5.6	26.2±7.6	28.1±6.6	13.8±7.0	40.0±10.7	9.5±0.8	11.8±0.9	**50.6±8.7**
	10_vs_11	19.4±4.6	10.0±4.1	10.6±5.4	12.5±4.4	12.5±3.4	16.2±6.1	10.0±0.5	9.8±0.3	**20.0±4.2**
	20_vs_20	37.5±4.4	6.2±2.0	11.9±4.1	32.5±8.1	23.8±2.5	36.2±5.1	10.0±2.0	10.1±0.6	**47.5±7.8**
	20_vs_23	**13.8±1.5**	1.2±1.5	0.0±0.0	12.5±5.6	11.2±7.8	12.5±8.1	8.1±1.4	8.8±0.8	**13.8±5.8**
Terran	5_vs_5	30.0±4.2	12.5±6.2	9.4±7.9	23.1±5.8	14.4±4.7	28.1±4.4	6.4±1.1	8.1±1.4	**30.6±8.2**
	10_vs_10	29.4±5.8	6.9±6.1	9.4±5.6	16.9±5.8	15.0±4.6	29.4±3.2	6.0±1.6	8.2±1.0	**32.5±5.8**
	10_vs_11	16.2±3.6	3.8±4.6	7.5±6.4	5.0±4.2	9.4±5.6	12.5±5.2	4.8±1.2	6.2±0.9	**19.4±5.4**
	20_vs_20	26.2±10.4	5.0±3.2	10.6±4.2	15.6±3.4	7.5±7.3	21.9±4.4	6.3±1.8	5.9±1.2	**29.4±3.8**
	20_vs_23	4.4±4.2	0.0±0.0	0.0±0.0	7.5±6.1	5.0±4.2	4.4±2.5	4.4±0.7	3.9±0.8	**9.4±5.2**
Zerg	5_vs_5	26.9±10.0	14.4±4.2	14.4±5.8	18.8±7.1	13.8±6.1	21.9±5.9	8.2±1.8	9.5±0.8	**31.2±7.7**
	10_vs_10	25.0±2.8	5.6±4.6	5.6±4.6	15.6±7.4	19.4±2.3	23.8±6.4	7.8±1.0	8.5±0.3	**33.8±11.8**
	10_vs_11	13.8±4.7	9.4±5.2	6.2±4.4	10.6±6.7	10.6±3.8	13.8±6.7	7.2±0.7	9.1±0.5	**19.4±3.6**
	20_vs_20	8.1±1.5	2.5±1.2	1.2±1.5	10.0±7.8	**12.5±4.4**	10.0±2.3	7.3±0.7	8.3±0.5	9.4±6.2
	20_vs_23	7.5±3.2	0.6±1.3	1.2±1.5	7.5±3.2	3.8±2.3	4.4±4.2	7.1±1.2	8.8±0.5	**11.2±4.2**

Table 2: Comparison of win rates for ComaDICE and baselines across SMACv2 tasks.

code was also employed to run these baselines for the SMACv2 environment. All hyperparameters were kept at their default settings, and each experiment was conducted with *five different random seeds* to ensure robustness and reproducibility of the results.

6.3 MAIN COMPARISON

We now present a comprehensive evaluation of our proposed algorithm, ComaDICE, against several baseline methods in offline MARL. The baselines selected for comparison include both standard and SOTA approaches, providing a robust benchmark to assess the effectiveness of ComaDICE.

Our evaluation focuses on two primary metrics: returns and winrates. Returns are the average rewards accumulated by the agents across multiple trials, providing a measure of policy effectiveness. Winrates, applicable in competitive environments such as SMACv1 and SMACv2, indicate the success rate of agents against opponents, reflecting the algorithm's robustness in adversarial settings.

The experimental results, summarized in Tables 1-3, demonstrate that ComaDICE consistently achieves superior performance compared to baseline methods across a range of scenarios. Notably, ComaDICE excels in complex tasks, highlighting its ability to effectively manage distributional shifts in challenging environments.

6.4 ABLATION STUDY - IMPACT OF THE REGULARIZATION PARAMETER ALPHA

We investigate how varying the regularization parameter alpha (α) affects the performance of our ComaDICE algorithm. The parameter α is crucial for balancing the trade-off between maximizing rewards and penalizing deviations from the offline dataset's distribution. We conducted experiments with α values ranging from $\{0.01, 0.1, 1, 10, 100\}$, evaluating performance using average winrates across all the SMACv2 tasks and average returns across all the MaMujoco tasks. These results, illustrated in Figure 1, highlight the sensitivity of ComaDICE to different α values. In particular, we

Instances		BCQ	CQL	ICQ	OMIGA	OptDICE	AlberDICE	ComaDICE (ours)
Hopper	expert	77.9 ± 58.0	159.1 ± 313.8	754.7 ± 806.3	859.6 ± 709.5	655.9 ± 120.1	844.6 ± 556.5	**2827.7 ± 62.9**
	medium	44.6 ± 20.6	401.3 ± 199.9	501.8 ± 14.0	**1189.3 ± 544.3**	204.1 ± 41.9	216.9 ± 35.3	822.6 ± 66.2
	m-replay	26.5 ± 24.0	31.4 ± 15.2	195.4 ± 103.6	774.2 ± 494.3	257.8 ± 55.3	419.2 ± 243.5	**906.3 ± 242.1**
	m-expert	54.3 ± 23.7	64.8 ± 123.3	355.4 ± 373.9	709.0 ± 595.7	400.9 ± 132.5	515.1 ± 303.4	**1362.4 ± 522.9**
Ant	expert	1317.7 ± 286.3	1042.4 ± 2021.6	2050.0 ± 11.9	2055.5 ± 1.6	1717.2 ± 27.0	1896.8 ± 33.7	**2056.9 ± 5.9**
	medium	1059.6 ± 91.2	533.9 ± 1766.4	1412.4 ± 10.9	1418.4 ± 5.4	1199.0 ± 26.8	1304.3 ± 2.6	**1425.0 ± 2.9**
	m-replay	950.8 ± 48.8	234.6 ± 1618.3	1016.7 ± 53.5	1105.1 ± 88.9	869.4 ± 62.6	1042.8 ± 80.8	**1122.9 ± 61.0**
	m-expert	1020.9 ± 242.7	800.2 ± 1621.5	1590.2 ± 85.6	1720.3 ± 110.6	1293.2 ± 183.1	1780.0 ± 23.6	**1813.9 ± 68.4**
Half Cheetah	expert	2992.7 ± 629.7	1189.5 ± 1034.5	2955.9 ± 459.2	3383.6 ± 552.7	2601.6 ± 461.9	3356.4 ± 546.9	**4082.9 ± 45.7**
	medium	2590.5 ± 1110.4	1011.3 ± 1016.9	2549.3 ± 96.3	**3608.1 ± 237.4**	305.3 ± 946.8	522.4 ± 315.5	2664.7 ± 54.2
	m-replay	-333.6 ± 152.1	1998.7 ± 693.9	1922.4 ± 612.9	2504.7 ± 83.5	-912.9 ± 1363.9	440.0 ± 528.0	**2855.0 ± 242.2**
	m-expert	3543.7 ± 780.9	1194.2 ± 1081.0	2834.0 ± 420.3	2948.5 ± 518.9	-2485.8 ± 2338.4	2288.2 ± 759.5	**3889.7 ± 81.6**

Table 3: Average returns for ComaDICE and baselines on MaMuJoCo benchmarks.

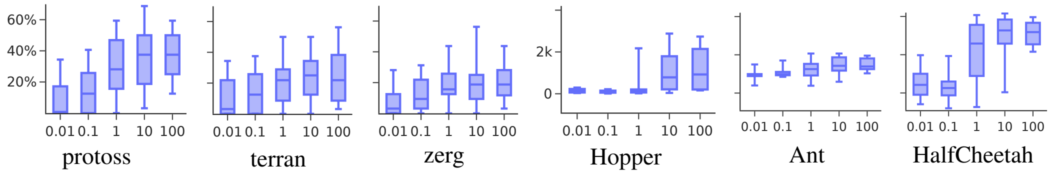


Figure 1: Impact of regularization parameter α on performance in different environments.

observe that ComaDICE achieves optimal performance when α is around 10, suggesting that the stationary distribution regularizer plays a essential role in the success of our algorithm.

In our appendix, we provide additional ablation studies to analyze the performance of our algorithm using different forms of f-divergence functions, as well as comparisons between 1-layer and 2-layer mixing network structures. The appendix also includes proofs of the theoretical claims made in the main paper, details of our experimental settings, and other experimental information.

7 CONCLUSION, FUTURE WORK AND BROADER IMPACTS

Conclusion. In this paper, we propose ComaDICE, a principled framework for offline MARL. Our algorithm incorporates a stationary distribution shift regularizer into the standard MARL objective to address the conventional distribution shift issue in offline RL. To facilitate training within a CTDE framework, we decompose both the global value and advantage functions using a mixing network. We demonstrate that, under our mixing architecture, the main objective function is concave in the value function, which is crucial for ensuring stable and efficient training. The results of this training are then utilized to derive local policies through a weighted BC approach, ensuring consistency between global and local policy optimization. Extensive experiments on SOTA benchmark tasks, including SMACv2, show that ComaDICE outperforms other baseline methods.

Limitations and Future Work: There are some limitations that are not addressed within the scope of this paper. For instance, we focus solely on cooperative learning, leaving open the question of how the approach would perform in cooperative-competitive settings. Additionally, in our training objective, the DICE term is designed to reduce the divergence between the learning policy and the behavior policy. As a result, the performance of the algorithm is heavily dependent on the quality of the behavior policy. Furthermore, our algorithm, like other baselines, still requires a large amount of data to achieve desirable learning outcomes. Improving sample efficiency would be another valuable area for future research.

Broader Impacts: Developing an offline MARL algorithm with a stationary distribution shift regularizer can enhance performance in costly real-time tasks like robotics, autonomous driving, and healthcare. It also enables safer exploration and broader adoption in high-stakes settings. However, reliance on the behavior policy means flawed or biased data could degrade performance, reinforcing biases or suboptimal behaviors. Additionally, the algorithm, like any AI systems, risks unintended misuse in surveillance or military applications, where multi-agent systems could manipulate environments without proper oversight.

ACKNOWLEDGMENT

This work is supported by the Lee Kong Chian Fellowship awarded to Tien Mai.

ETHICAL STATEMENT

Our work introduces ComaDICE, a framework for offline MARL, aimed at improving training stability and policy optimization in complex multi-agent environments. While this research has significant potential for positive applications, particularly in domains such as autonomous systems, resource management, and multi-agent simulations, it is crucial to address the ethical implications and risks associated with this technology.

The deployment of reinforcement learning systems in real-world, multi-agent settings raises concerns about unintended behaviors, especially in safety-critical domains. If the policies learned by ComaDICE are applied without proper testing and validation, they may lead to undesirable or harmful outcomes, especially in areas such as autonomous driving, healthcare, or robotics. Additionally, bias in the training data or simulation environments could result in suboptimal policies that unfairly impact certain agents or populations, potentially leading to ethical concerns regarding fairness and transparency.

To mitigate these risks, we emphasize the need for extensive testing and validation of policies generated using ComaDICE, particularly in real-world environments where the consequences of errors could be severe. It is also essential to ensure that the datasets and simulations used in training are representative, unbiased, and carefully curated. We encourage practitioners to use human oversight and collaborate with domain experts to ensure that ComaDICE is applied responsibly, particularly in high-stakes settings.

REPRODUCIBILITY STATEMENT

In order to facilitate reproducibility, we have submitted the source code for ComaDICE, along with the datasets utilized to produce the experimental results presented in this paper (all these will be made publicly available if the paper gets accepted). Additionally, in the appendix, we provide details of our algorithm, including key implementation steps and details needed to replicate the results. The hyper-parameter settings for all experiments are also included to ensure that others can reproduce the findings under the same experimental conditions. We invite the research community to explore and apply the ComaDICE framework in various environments to further validate and expand upon the results reported in this work.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL `http://arxiv.org/abs/1606.01540`.

The Viet Bui, Tien Mai, and Thanh Hong Nguyen. Inverse factorized q-learning for cooperative multi-agent imitation learning. *Advances in Neural Information Processing Systems*, 38, 2024.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

Peng Cheng, Xianyuan Zhan, Wenjia Zhang, Youfang Lin, Han Wang, Li Jiang, et al. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*, 19, 2020.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, 10(6), 2009.

Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 6863–6877. PMLR, 2022.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

Jiechuan Jiang and Zongqing Lu. Offline decentralized multi-agent reinforcement learning. In *ECAI*, pp. 1148–1155, 2023.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pp. 651–673. PMLR, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. *arXiv preprint arXiv:2204.08957*, 2022.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, and Ya-Qin Zhang. Proto: Iterative policy regularized offline-to-online reinforcement learning. *arXiv preprint arXiv:2305.15669*, 2023.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Dealing with the unknown: Pessimistic offline reinforcement learning. In *Conference on Robot Learning*, pp. 1455–1464. PMLR, 2022.

Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. *arXiv preprint arXiv:2402.00348*, 2024.

Daiki E Matsunaga, Jongmin Lee, Jaeseok Yoon, Stefanos Leonardos, Pieter Abbeel, and Kee-Eung Kim. Alberdice: addressing out-of-distribution joint actions in offline multi-agent rl via alternating stationary distribution correction estimation. *Advances in Neural Information Processing Systems*, 36:72648–72678, 2023.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2):233–248, 2023.

Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality. *arXiv preprint arXiv:2001.01866*, 2020.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Haoyi Niu, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, Xianyuan Zhan, et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36599–36612, 2022.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pp. 17221–17237. PMLR, 2022.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Harshit Sikchi, Amy Zhang, and Scott Niekum. Imitation from arbitrary experience: A dual unification of reinforcement and imitation learning methods. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Wei-Cheng Tseng, Tsun-Hsuan Johnson Wang, Yen-Chen Lin, and Phillip Isola. Offline multi-agent reinforcement learning with knowledge distillation. *Advances in Neural Information Processing Systems*, 35:226–237, 2022.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

Jun Wang, Yaodong Yang, and Zongqing Wang. Trust region policy optimization in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2022a.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 36, 2022b.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Haoran Xu, Xianyuan Zhan, Jianxiong Li, and Honglei Yin. Offline reinforcement learning with soft behavior regularization. *arXiv preprint arXiv:2110.07395*, 2021.

Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4085–4098, 2022a.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 24725–24742, 2022b.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022c.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Hongchang Zhang, Jianzhun Shao, Yuhang Jiang, Shuncheng He, Guanwen Zhang, and Xiangyang Ji. State deviation correction for offline reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 9022–9030, 2022.

Qin Zhang, Linrui Zhang, Haoran Xu, Li Shen, Bowen Wang, Yongzhe Chang, Xueqian Wang, Bo Yuan, and Dacheng Tao. Saformer: A conditional sequence modeling approach to offline safe reinforcement learning. *arXiv preprint arXiv:2301.12203*, 2023.

Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International conference on machine learning*, pp. 12491–12500. PMLR, 2021.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. *arXiv preprint arXiv:2401.10700*, 2024.

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* $\min_{\nu^{tot}} \left\{\widetilde{\mathcal{L}}(\nu^{tot})\right\}$*, where*

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

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* $\widetilde{\mathcal{L}}(\nu^{tot})$ *w.r.t.* θ *is given as*

$$\nabla_\theta\widetilde{\mathcal{L}}(\nu^{tot}) = (1-\gamma)\mathbb{E}_{\boldsymbol{s}\sim p_0}[\nabla_\theta\nu^{tot}(\boldsymbol{s})] + \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\rho^{\boldsymbol{\mu}_{tot}}}\left[\nabla_\theta A^{tot}_{\nu}(\boldsymbol{s},\boldsymbol{a})w^{tot*}_{\nu}(\boldsymbol{s},\boldsymbol{a})\right].$$

where $w^{tot*}_{\nu}(s,a) = \max\{0, {f'}^{-1}(A^{tot}_{\nu}(\boldsymbol{s},\boldsymbol{a})/\alpha)\}$*, 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 $\widetilde{\mathcal{L}}(\nu^{tot})$, 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, $\nabla_\theta\widetilde{\mathcal{L}}(\nu^{tot})$.

We write the Lagrange dual function as:

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

where $w^{tot}_{\nu}(\mathbf{s},\mathbf{a}) = \frac{\rho^{\boldsymbol{\pi}_{tot}}(\mathbf{s},\mathbf{a})}{\rho^{\boldsymbol{\mu}_{tot}}(\mathbf{s},\mathbf{a})}$. We now see that, for each $(\mathbf{s},\mathbf{a})$, each component $-\alpha f\left(w^{tot}_{\nu}(\mathbf{s},\mathbf{a})\right) + w^{tot}_{\nu}(\mathbf{s},\mathbf{a})A^{tot}_{\nu}(\mathbf{s},\mathbf{a})$ is maximized at:

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

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

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

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

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

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

To get derivatives of $\widetilde{\mathcal{L}}(\nu^{tot})$, 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:

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

which implies:

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

we complete the proof. ☐

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 $\widetilde{\mathcal{L}}(\boldsymbol{\nu}, \theta)$ is convex in $\boldsymbol{\nu}$.

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

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

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

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

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 $n = 0, \ldots$. Here we note that $F^n(\mathbf{X})$ is a vector, so when we say "$F^n(\boldsymbol{X})$ *is convex and non-decreasing in* $\boldsymbol{X}$," 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:

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

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:

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

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 $n = 0, \ldots, N$.

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 $\boldsymbol{\nu}$. □

We are now ready to prove the convexity of $\widetilde{\mathcal{L}}(\boldsymbol{\nu},\theta)$ 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:

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

where

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

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 $\widetilde{\mathcal{L}}(\boldsymbol{\nu},\theta)$ 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:

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

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

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

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

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

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

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:

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

which directly implies that $\boldsymbol{\pi}_{tot}^*$ is optimal for the global WBC problem 8.

$\square$

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 $\widetilde{\mathcal{L}}$ is approximately equivalent to optimizing factorized occupancy ratios.

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

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

where $\mathcal{M}_\theta$ 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:

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

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:

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

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

Thus, the above discussion implies that optimizing the main objective function $\widetilde{\mathcal{L}}$ 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 $\widetilde{\mathcal{L}}$ is expected to behave similarly to minimizing each local function $\widetilde{\mathcal{L}}_i$, 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 MaMujoco 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 **Co**operative **MA**RL with Stationary **DI**stribution **C**orrection **E**stimation

```
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 $\mathcal{D}$.
Output: Local optimized polices $\pi_i$.
# Training the occupancy ratio $w^{tot*}$
for a certain number of training steps do
    $\psi_q = \psi_q - \lambda_{\psi_q} \nabla_{\psi_q} \mathcal{L}(\psi_q)$    # Update Q-function towards the MSE in 10
    $\theta = \theta - \lambda_\theta \nabla_\theta \widetilde{\mathcal{L}}(\psi_\nu, \theta)$    # Update $\theta$ to minimize the loss in 11
    $\psi_\nu = \psi_\nu - \lambda_{\psi_\nu} \nabla_{\psi_\nu} \widetilde{\mathcal{L}}(\psi_\nu, \theta)$    # Update $\psi_\nu$ to minimize the loss in 11
end for
# Training local policy
for a certain number of training steps do
    $\eta_i = \eta_i + \lambda_\eta \nabla_{\eta_i} \mathcal{L}_\pi(\eta_i)$    # Update the local policy by optimizing 12
end for
Return $\pi_i(a_i|o_i; \eta_i)$, $i = 1, ..., n$
```

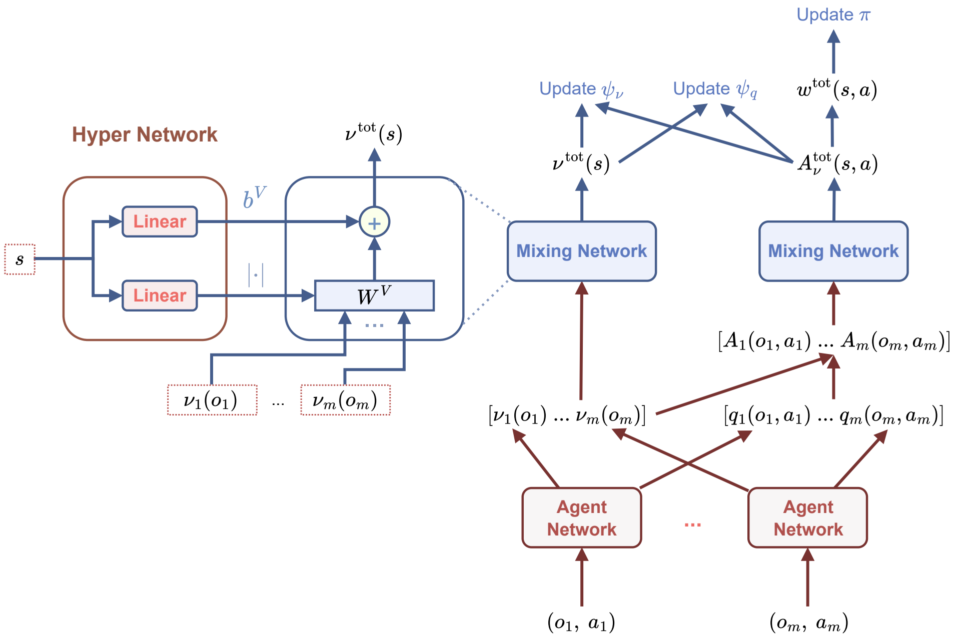


Figure 2: Our ComaDICE model architecture.

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.

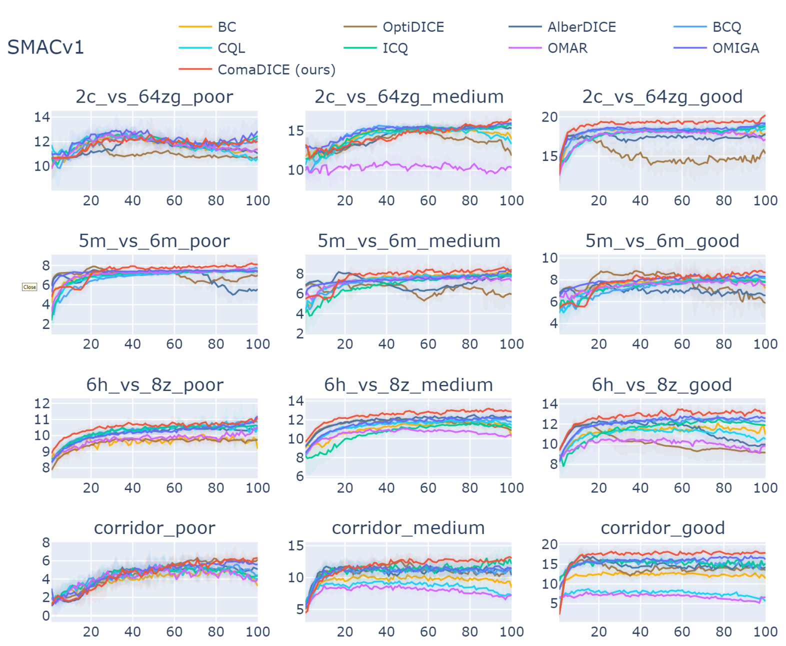


Figure 3: Evaluation of SMACv1 tasks comparing the returns achieved by ComaDICE and baselines.

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.

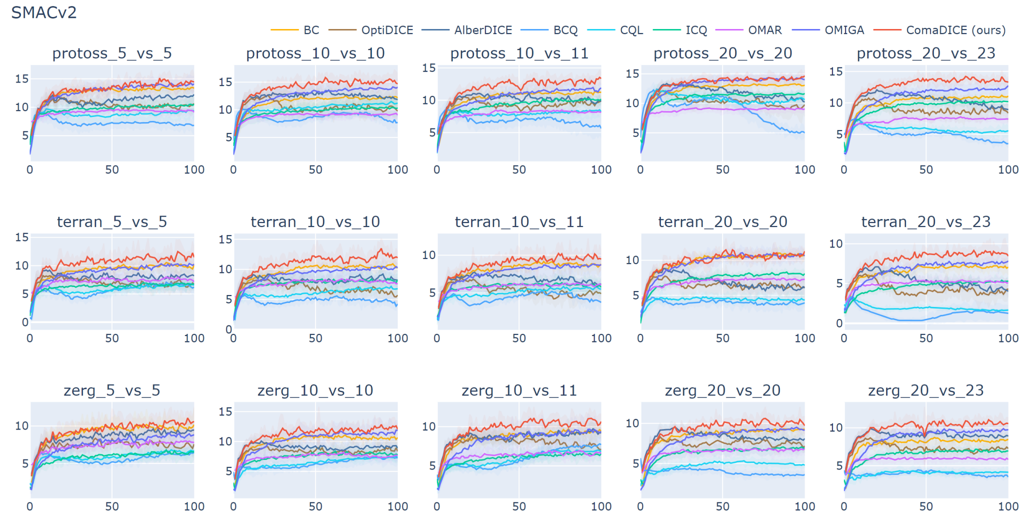


Figure 4: Evaluation of SMACv2 tasks comparing the returns achieved by ComaDICE and baselines.

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
OptDICE	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

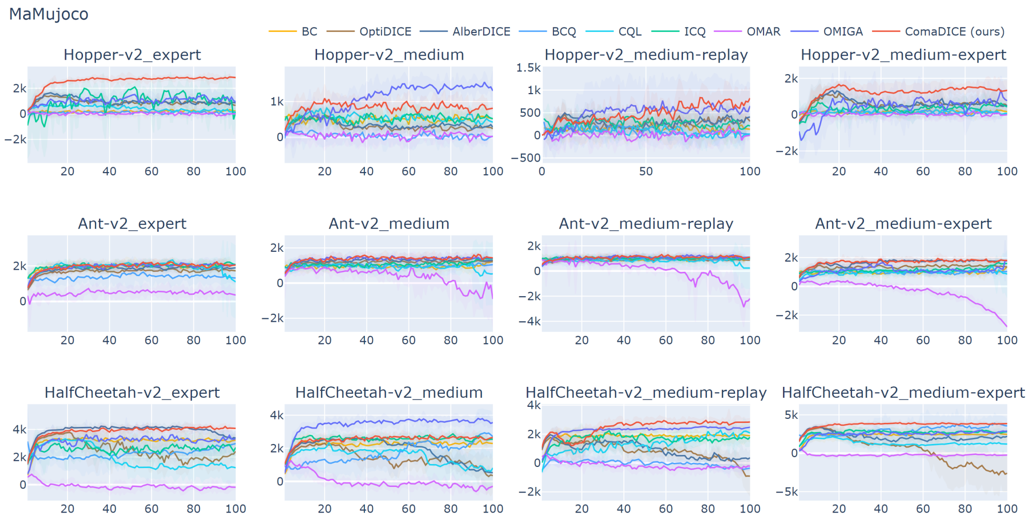


Figure 5: Evaluation of MaMujoco tasks comparing the returns achieved by ComaDICE and baselines.

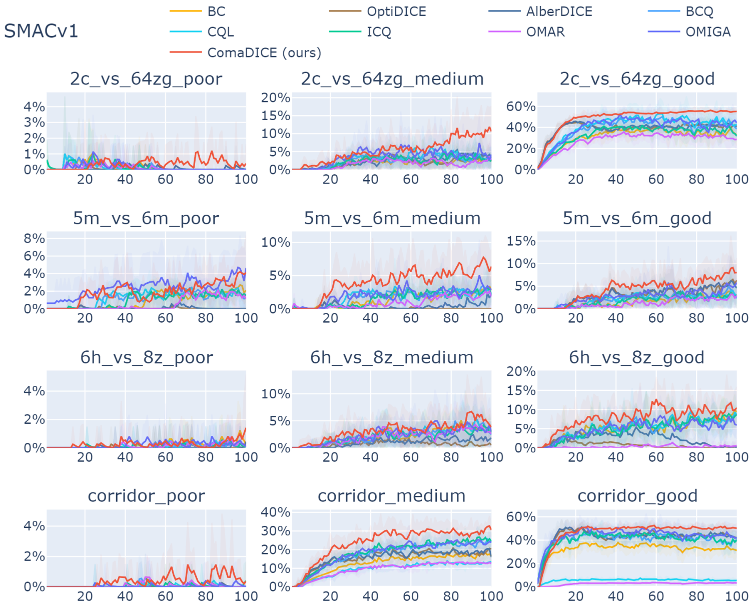


Figure 6: Evaluation of SMACv1 tasks comparing the winrates achieved by ComaDICE and baselines.

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 .

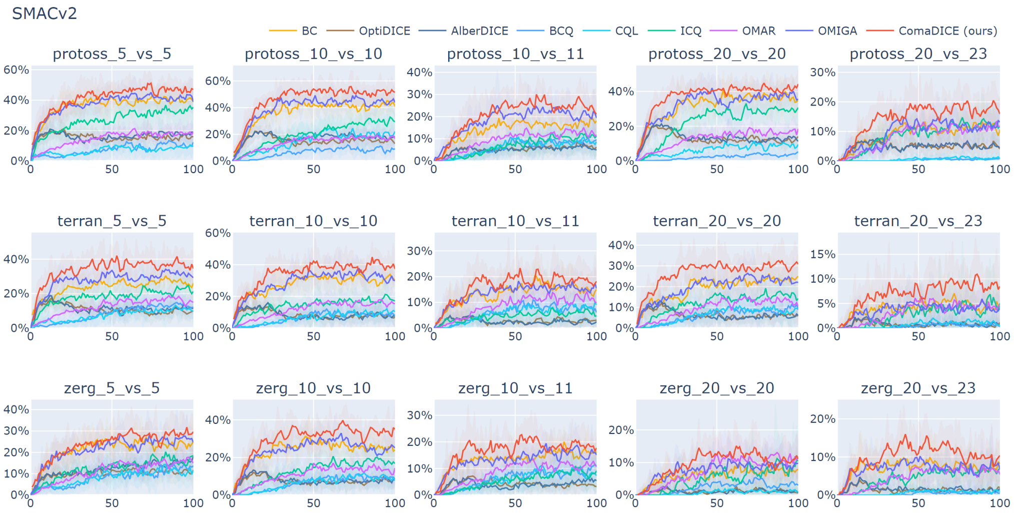


Figure 7: Evaluation of SMACv2 tasks comparing the winrates achieved by ComaDICE and baselines.

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	14.0±2.0
	10_vs_10	12.8±0.9	14.0±0.8	13.4±1.2	14.6±1.8	14.1±1.3
	10_vs_11	9.9±1.1	11.1±0.8	11.3±1.2	13.2±0.9	12.2±1.1
	20_vs_20	10.3±0.5	11.1±1.0	12.2±0.9	14.8±1.0	13.2±0.4
	20_vs_23	8.0±2.3	11.2±1.2	11.7±0.6	13.3±0.9	13.2±0.5
Terran	5_vs_5	11.1±1.8	10.1±1.2	9.0±1.0	10.7±1.5	12.6±1.9
	10_vs_10	8.5±0.8	10.3±0.7	10.4±1.1	11.8±0.9	11.8±1.7
	10_vs_11	7.5±0.7	8.6±2.1	8.5±1.6	9.4±0.9	9.6±0.9
	20_vs_20	6.2±1.1	6.4±1.7	9.1±0.7	11.8±0.5	9.3±0.6
	20_vs_23	5.5±1.1	6.5±1.6	6.5±0.8	8.2±0.7	8.2±0.4
Zerg	5_vs_5	7.9±0.6	9.3±0.9	10.5±1.4	10.7±2.0	10.4±1.2
	10_vs_10	10.9±1.5	11.4±1.5	11.8±0.7	11.5±1.0	10.9±2.2
	10_vs_11	10.1±2.5	9.1±1.2	10.0±1.2	11.0±0.9	9.8±0.8
	20_vs_20	8.0±0.5	9.2±1.3	9.2±1.0	9.4±1.2	10.5±0.9
	20_vs_23	9.1±1.1	10.0±0.7	10.4±0.6	10.5±0.8	10.1±0.7

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±67.9	107.9±65.5	545.7±820.6	2827.7±62.9	2690.7±58.6
	medium	149.6±96.8	107.5±66.9	244.7±267.5	822.6±66.2	807.5±122.2
	m-replay	165.6±104.1	109.6±38.7	155.6±61.6	906.3±242.1	186.5±16.8
	m-expert	119.1±77.1	95.6±69.5	58.8±26.1	1362.4±522.9	1358.4±595.1
Ant	expert	1016.4±196.5	1179.0±273.7	1927.7±174.1	2056.9±5.9	1950.0±3.3
	medium	907.3±32.2	1000.0±90.4	1424.3±3.1	1425.0±2.9	1354.6±2.5
	m-replay	969.1±21.9	978.4±39.6	944.6±28.9	1122.9±61.0	1072.1±41.4
	m-expert	915.8±364.1	1132.9±282.2	738.5±250.2	1813.9±68.4	1559.6±86.8
Half Cheetah	expert	1068.9±635.2	935.2±905.9	3637.0±80.9	4082.9±45.7	3843.7±149.4
	medium	575.9±724.8	445.2±403.9	2690.0±92.4	2664.7±54.2	2523.4±59.0
	m-replay	412.3±310.5	233.5±270.1	861.6±173.5	2855.0±242.2	2557.4±241.5
	m-expert	-107.5±298.1	-275.9±544.5	1136.9±1608.3	3889.7±81.6	3605.6±70.4

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

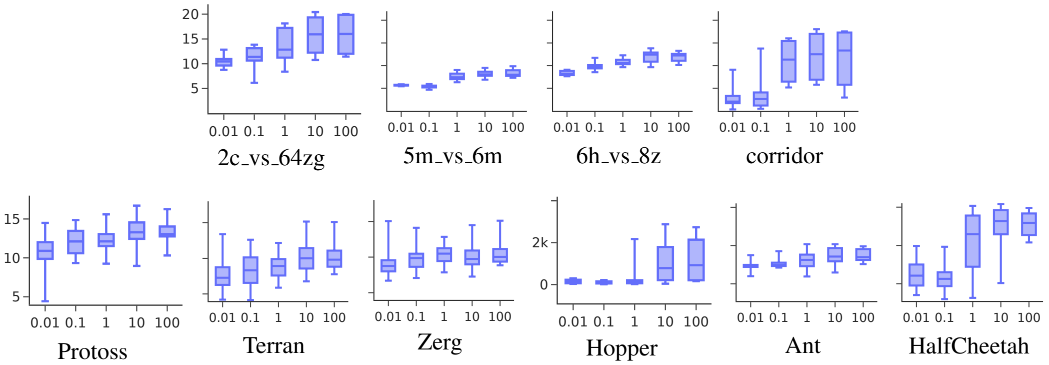


Figure 8: Impact of alpha on returns for ComaDICE and baselines.

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.

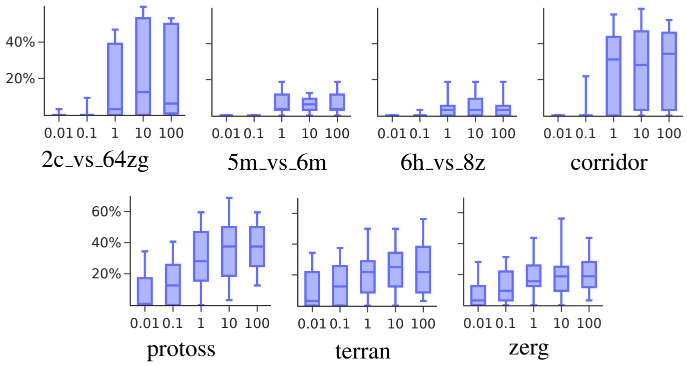
We assess their impact on both returns and winrates in environments such as SMACv1, SMACv2, and MaMujoco. Our results, detailed in Tables 16-20, reveal that the choice of f-divergence function significantly influences the algorithm's effectiveness. For instance, the Soft-χ^2 divergence consistently yields 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.



2c_vs_64zg 5m_vs_6m 6h_vs_8z corridor

protoss terran zerg

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

B.6.1 RETURNS

Instances		$f_{\chi^2}(x)$	$f_{\text{KL}}(x)$	$f_{\text{Soft-}\chi^2}(x)$
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		$f_{\chi^2}(x)$	$f_{\text{KL}}(x)$	$f_{\text{Soft-}\chi^2}(x)$
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		$f_{\chi^2}(x)$	$f_{\text{KL}}(x)$	$f_{\text{Soft-}\chi^2}(x)$
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		$f_{\chi^2}(x)$	$f_{\text{KL}}(x)$	$f_{\text{Soft-}\chi^2}(x)$
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		$f_{\chi^2}(x)$	$f_{\text{KL}}(x)$	$f_{\text{Soft-}\chi^2}(x)$
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.

$$\begin{array}{c|c|c|} & A & B \\ \hline A & 0 & 1 \\ \hline B & 1 & -2 \\ \hline \end{array}$$

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
A	0.00	1.00
B	0.00	0.00

(b)	A	B
A	0.00	1.00
B	0.00	0.00

(c)	A	B
A	0.00	1.00
B	0.00	0.00

(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).