# Learning to Coordinate from Offline Datasets with Uncoordinated Behavior Policies

Jinming Ma
School of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui, China
jinmingm@mail.ustc.edu.cn

Feng Wu
School of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui, China
wufeng02@ustc.edu.cn

## ABSTRACT

In offline multi-agent reinforcement learning (RL), multiple agents must learn to coordinate from previously collected datasets. Like the single-agent case, we must handle the distribution shift issue from the datasets. Most importantly, we also need to deal with possible miscoordination in the datasets, collected by some uncoordinated behavior policies. To address this, we propose a novel offline multi-agent RL method using counterfactual sample-average approximation with subteam masking. Specifically, we compute the best-response policy for each agent using sample-average approximation. For the miscoordination issue, we use counterfactual mechanism and subteam masking to reason about the agents' contributions to the team. Based on this, each agent learns to coordinate from the uncoordinated datasets. Empirically, we evaluate our method in two benchmark domains: a continuous multi-agent MuJoCo control domain, and a challenging cooperation environment Starcraft II domain. Our experimental results confirm that our approach can achieve significantly better performance than several state-of-the-art methods. The source code is available at: https://github.com/JinmingM/CAST-BCQ.

## KEYWORDS

Multi-agent Reinforcement Learning; Offline Reinforcement Learning; Deep Reinforcement Learning

**ACM Reference Format:**
Jinming Ma and Feng Wu. 2023. Learning to Coordinate from Offline Datasets with Uncoordinated Behavior Policies. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

## 1 INTRODUCTION

Offline reinforcement learning (RL) has shown great potential in applying RL to real-world tasks because it can learn from a pre-collected dataset. To date, most offline RL algorithms focus on the single-agent setting [1, 9, 15, 16, 27]. However, many sequential decision-making problems in real-world scenarios involve multiple agents, such as multi-robot control [2, 4, 31], traffic signal control [20, 30], and power grids [3]. Offline RL is appealing for these domains especially when interaction with the environment is costly or risky. Currently, the work for offline multi-agent RL (MARL) is sparse because the action space grows exponentially with the number of agents. This will make the key challenge of offline RL — the distribution shift issue — more difficult to tackle, which leads to extrapolation error [9] in value estimation. Recently, several efforts [12, 34] have been made to extend offline RL to multi-agent scenarios. They mainly focus on reducing the extrapolation error introduced by the large action space.

It is generally believed that the extrapolation error is incurred by the divergence between the distributions of the learned policy and the dataset collected by behavior policy. Under the offline setting, the error cannot be corrected by interacting with the environment. Therefore, the quality of the dataset is critical for the performance of the learned policy. In practice, the dataset is collected by running some behavior policies in the environment. In multi-agent settings, it is highly likely that the behavior policies are *uncoordinated* when collecting datasets. This may be because the original systems act independently. For example, in traffic signal control, each intersection is controlled independently by traditional methods. This may be also because collecting coordinated datasets is expensive. For example, in multiplayer online battle arena games, datasets collected from professional teams playing the games are costly. A large number of datasets are generated by uncoordinated amateur players. Therefore, the behavior policies are usually uncoordinated. As a result, the datasets generated by uncoordinated behavior policies are often of poor quality. Generally, multi-agent coordination is a hard problem that we want to solve by the offline MARL algorithms. As shown later in the experiments, existing approaches [12, 34] failed to address this specific challenge for offline MARL, incurred by uncoordinated behavior policies.

Against this background, we propose a novel offline multi-agent RL called *Counterfactual Sample-Average Approximation with Subteam Masking*, which can effectively learn coordinated policies with a given dataset collected by uncoordinated behavior policies. Here, we model the multi-agent coordination problem as a Dec-POMDP. Then, we optimize the best-response policy for each agent conditioned on the other agents' policies. Note that the Dec-POMDP can be viewed as a single-agent POMDP from the perspective of each agent if the other agents' policy is fixed. Since the other agents' policies are unknown, we maintain a distribution over them. From this distribution, we sample a set of the other agents' policies. Then, we leverage *sample-average approximation* [14] and BCQ [9] to learn the agent's policy. We address the potential miscoordination issue, inherited from uncoordinated behavior policies, by reasoning the agents' contribution to the whole team. Specifically, we rely on the *counterfactual* technique for reasoning the individual contribution of each agent and the *subteam masking* method for the inter-dependency among the agents. By doing so, each agent

Feng Wu is the corresponding author.

*Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.* 

learns to coordinate with the others given datasets generated by uncoordinated behavior policies.

In the experiments, we evaluate our method on two common multi-agent tasks, where multi-agent MuJoCo [22] is a continuous control domain with tight coordination and StarCraft II [24] is a competitive and cooperative with many agents. Our experimental results show that our method achieved effective coordination and significantly outperformed several state-of-the-art offline RL baselines in all the tested problem instances. Moreover, our ablations confirm the effectiveness of the proposed counterfactual SAA and subteam masking methods for handling uncoordinated datasets.

## 2 BACKGROUND

### 2.1 Offline Reinforcement Learning

In RL, the agent learns a policy by interacting with the environment. The underlying model can be formulated as a *Markov decision process* (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, with the state space $\mathcal{S}$ , the action space $\mathcal{A}$ and the transition function $\mathcal{P}(s'|s,a) : S \times A \times S \to [0,1]$. At each time step, the agent performs action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$, and receives a reward $r = \mathcal{R}(s,a)$. The goal of solving the MDP is to find an optimal policy for the agent that maximizes the expectation of the sum of rewards discounted by $\gamma \in (0,1]$. In the context of offline RL, the agent needs to learn a policy from a fixed dataset collected by some behavior policy $\beta$, which contains a set of transitions $\langle s, a, s', r \rangle$.

The main challenge for offline RL is the *extrapolation error*, which is mainly caused by generalization error in the approximate value function of *out-of-distribution* (OOD) actions [9, 15]. To minimize the extrapolation error, typical offline RL methods constrain the learned policy away from OOD actions and regularize the learned policy to be close to the behavior policy. For instance, BCQ [9] optimizes policy $\pi$ by sampling $n$ actions from the behavior policy $\beta$ and then chooses the one with the highest $Q$-value as:

$$\begin{aligned} &\pi(s) = \arg\max_{a^j} Q(s, a^j + \xi(s, a^j, \Phi)), \\ &\text{s.t. } \{a^j \sim G_w(s)\}_{j=1}^{n}. \end{aligned} \tag{1}$$

where $\xi(s, a^j, \Phi)$ is the perturbation model to adjust $a^j$ in the range $[-\Phi, \Phi]$ to increase the diversity of seen actions and $G_w(s)$ is the generative model trained to model the behavior policy. Another common idea is to train the $Q$-function pessimistic to OOD actions. For example, CQL [16] penalizes the $Q$-function at states in the dataset for actions not observed in the dataset $D$.

To date, most of the previous research on offline RL have been successfully conducted in single-agent scenarios [17]. Since many real-world applications require multiple agents to work together, it is natural to extend offline RL to multi-agent scenarios.

### 2.2 Offline Multi-Agent RL

We can extend the MDP to the multi-agent setting as a *decentralized partially observable Markov decision process* (Dec-POMDP) with $N$ agents, defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{R}, \gamma \rangle$, where $s \in \mathcal{S}$ denotes the true state of the environment. Each agent $i$ chooses an action $a_i \in \mathcal{A}_i$ at each time step, forming a joint action vector $\mathbf{a} = [a_i]_{i=1}^{N}$. Different from the MDP, the next state follows the transition function $\mathcal{P}(s', \mathbf{o}|s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \to [0,1]$. Meanwhile, each agent $i$ receives an local (partial) observation $o_i \in \mathcal{O}_i$ and a reward $r$ based on the shared reward function $\mathcal{R} : S \times \mathcal{A}^N \to \mathbb{R}$. The goal is to find a set of optimal policies $\boldsymbol{\pi} = \{\pi_1, ..., \pi_N\}$, where aim to maximize the total discounted return $\sum_{t=0}^{T} \gamma^t r^t$ with the discount factor $\gamma$.

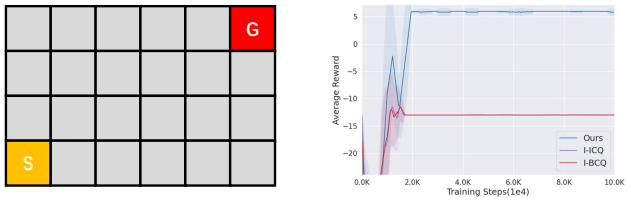
In offline multi-agent setting, agents learn their policy from a pre-collected dataset generated by some joint behavior policy. A simple idea is to treat the whole problem as a joint MDP and apply single-agent offline RL directly to learn a joint policy. Unfortunately, the joint policy cannot be executed in a decentralized manner. Furthermore, the OOD issue becomes more severe because the action space grows exponentially with the number of agents [23]. To tackle this, ICQ-MA [34] proposes to estimate the target value following the SARSA-like approach and decompose the value function by QMIX [23]. Specifically, it constrains the value estimate based only on seen state-action pairs in the dataset. Therefore, the performance largely depends on the dataset quality.

Another straightforward approach for decentralized execution is to apply single-agent offline RL to each agent and independently learn a set of policies, one for each agent. However, this will not work for multi-agent problems requiring coordination among the agents. Along this track, MABCQ [12] proposes to re-weight the offline transition dynamics by increasing the transition with high value and normalizing the biased transition dynamics. However, this method considers the underlying problem as a multi-agent MDP (MMDP) and each agent's policy is based on the state of the environment, which is not suited for the Dec-POMDP. As observed in our experiments, existing methods fail to learn good policies, especially for datasets generated by uncoordinated behavior policies. Next, we will illustrate the challenge of learning to coordinate from uncoordinated datasets in the following motivation example.

### 2.3 Motivation and Challenge

To illustrate the challenges, we design a simple and cooperative problem. As shown in Figure 1 (a), a robot needs to complete the navigation task in the $4 \times 6$ grid world, from the start position (i.e., the yellow box) in the lower left corner to the target position (i.e., the red box) in the upper right corner. Meanwhile, it must avoid being out of the grid's boundaries (i.e., falling off a cliff).

The robot is controlled by two agents: *agent 1 controls the movement in the x-direction, and agent 2 controls the movement in the y-direction.* Specifically, their actions include `forward`, `backward` and `stay`. For instance, at position (x, y), both agents 1 and 2 perform the `forward` action, the robot will move to the position (x+1,




(a) (b)

**Figure 1: Multi-agent navigation in the 4×6 grid world. (a) The environment. (b) The learning curve.**

y+1). At each time step, agents receive their position and then perform their respective actions. The agents must cooperate with each other and receive the shared reward: +10 points for reaching the target location, −10 points for falling off the cliff, and −1 point for step cost. When they reach the goal, fall off a cliff or run out of time with a maximum of 30 steps, the environment will be reset.

We tested state-of-the-art offline RL algorithms, i.e., BCQ [9] and ICQ[34]. To apply them to the multi-agent scenarios, we *independently* train each agent $i$ with BCQ or ICQ, called I-BCQ or I-ICQ respectively. For data collection, we used random behavior policies and stored 100 trajectories as the dataset.

As shown in Figure 1 (b), both I-BCQ and I-ICQ failed to learn an effective policy even in such a sample domain, because they cannot accurately estimate the values of their actions conditioned on the other agents' policies. For example, at position (4, 4), agent 1 should choose the `forward` action. However, if agent 2 also performs the `forward` action in the dataset, the robot will fall off the cliff and gets a penalty. As a result, agent 1 may fail to estimate the correct value of the `forward` action from uncoordinated experiences.

To summarize, one of the key challenges in offline MARL is that: *how to accurately estimate the Q-value conditioned on the other agents' policies, with a dataset collected by uncoordinated behavior policies*. Motivated by this, we propose a new approach to solve this critical challenge, as described next.

## 3 MAIN METHOD

We consider the following learning problem for agent $i$:

$$\begin{aligned} &\pi_i(o_i) = \arg\max_{a_i} Q_i(o_i, a_i; \hat{\pi}_{-i}), \\ &\text{s.t. } \{a_i \sim G_{w^i}(o_i)\}. \end{aligned} \tag{2}$$

where $\pi_i$ is agent $i$'s policy, $\hat{\pi}_{-i}$ is the other agents' policy and $G_{w^i}(o_i)$ is the generative model that generates the action distribution according to the local observation. Note that the other agents become parts of the environment if their policy $\hat{\pi}_{-i}$ is fixed. In this case, the Dec-POMDP can be viewed as a single-agent POMDP from the perspective of agent $i$.

Similar to BCQ [9], we can maintain a generative model to approximate the action distribution in the dataset. Specifically, for agent $i$, we use a conditional variational auto-encoder (VAE) [13]: $G_{w^i} = \{E_{w_1^i}, D_{w_2^i}\}$, which generates the action distribution according to the observation. The VAE can be trained by the following loss to maintain the batch-constraint:

$$\mathcal{L}(w^i) = \arg\min_{w^i} (a_i - \tilde{a}_i)^2 + D_{KL}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1)) \tag{3}$$

where $(o_i, a_i) \in B_i, \tilde{a}_i = D_{w_2^i}(o_i, z \sim \mathcal{N}(\mu, \sigma))$, and $\mu, \sigma = E_{w_1^i}(o_i, a_i)$ are the mean and variance of the random latent vector $z$.

Intuitively, if $\hat{\pi}_{-i}$ is known and stationary, we can simply learn agent $i$'s policy. For instance, if $\hat{\pi}_{-i}$ is the behavior policy of the dataset, it is equivalent to running BCQ independently for agent $i$. Additionally, if $\hat{\pi}_{-i}$ is a joint policy learned by BCQ from the dataset, it can be viewed as learning a local policy $\pi_i$ to fit the joint one. This is similar to ICQ-MA, where the results depend on the joint policy learned by the centralized critic. Ideally, if $\hat{\pi}_{-i}$ is the optimal policy of the other agents, we can learn the $\pi_i$ will be the optimal policy for agent $i$. Unfortunately, the optimal policy of the other agents is hidden. Nevertheless, it is stationary and determined by the datasets in offline settings.

Therefore, we propose to approximate it by maintaining a distribution over the other agents' policies. Indeed, the distribution considered here is inspired by the *multi-agent belief state*: $b_i \in \Delta(S \times \Pi_{-i})$, which is a distribution over the state space and the other agents' policies. This has been widely used for solving Dec-POMDPs, which first generate a set of multi-agent belief states and then optimize the policy for one agent at a time [21]. In what follows, we will give more details on applying this to offline MARL.

### 3.1 Counterfactual Sample-Average Approximation

Let $\hat{\Pi}_{-i}$ denote the distribution over the other agents' policy. Note that we want to optimize $\pi_i$ conditioned on $\hat{\pi}_{-i}$, which is a random element with the distribution $\hat{\Pi}_{-i}$. The learning problem in Eq. 2 can be solved using the *Sample-Average Approximation* (SAA) technique [14] as follow:

$$\begin{aligned} &\pi_i(o_i) = \arg\max_{a_i} \frac{1}{M} \sum_{m=1}^{M} Q_i(o_i, a_i; \hat{\pi}_{-i}^m), \\ &\text{s.t. } \{a_i \sim G_{w^i}(o_i)\} \text{ and } \{\hat{\pi}_{-i}^m \sim \hat{\Pi}_{-i}\}. \end{aligned} \tag{4}$$

In SAA, we sample $\hat{\pi}_{-i}^1, \hat{\pi}_{-i}^2, \cdots, \hat{\pi}_{-i}^M$ with the distribution $\hat{\Pi}_{-i}$, and optimize their average value. Intuitively, if $\hat{\Pi}_{-i}$ is proportional to the optimal distribution and $M \to \infty$, $\pi_i$ will eventually converge to the optimal policy of agent $i$.

To coordinate with the other agents, the maximized items in Eq. 4 must represent agent $i$'s contribution to the whole team since agent $i$ shares the same reward with all the other agents in Dec-POMDPs. To capture this, we borrow ideas from the counterfactual mechanism [5, 18] and use a counterfactual baseline. Then, the learning problem in Eq. 4 can be rewritten as follow:

$$\begin{aligned} &\max_{a_i} \frac{1}{M} \sum_{m=1}^{M} [Q_i(o_i, a_i; \hat{\pi}_{-i}^m) - Q_i(o_i, a_i^d; \hat{\pi}_{-i}^m)], \\ &\text{s.t. } \{a_i \sim G_{w^i}(o_i)\} \text{ and } \{\hat{\pi}_{-i}^m \sim \hat{\Pi}_{-i}\}. \end{aligned} \tag{5}$$

where $a_i^d$ is the default action for agent $i$. This counterfactual baseline is useful because the shaped difference reward will encourage agent $i$ to take the best-response conditioned on others' joint policy. Now, the next step is to maintain a distribution over the other agents' policy and sample from it.

### 3.2 Policy Sampling with Subteam Masking

Although the optimal policy for the other agents is unknown, we can approximate it with the joint policy since all the agents' observations are available during the centralized training process. This is very similar to the centralized critic of ICQ-MA. Specifically, we consider the Dec-POMDP as a joint MDP with joint actions and observations. We merge the datasets of all agents as the joint dataset $\mathbf{B} = [B_i]_{i=1}^N$, which contains the transitions $\langle \mathbf{o}, \boldsymbol{a}, r, \mathbf{o}' \rangle = \langle [o_i]_{i=1}^N, [a_i]_{i=1}^N, r, [o_i']_{i=1}^N \rangle$. Given this, we learn the joint value function $Q(\mathbf{o}, \boldsymbol{a}; \phi)$ by minimizing the following loss:

$$\mathcal{L}(\phi) = \mathop{\mathbb{E}}_{(\mathbf{o}, \boldsymbol{a}, \mathbf{o}') \sim \mathbf{B}} [r + \gamma \max_{\mathbf{a}' \sim \boldsymbol{\pi}(\mathbf{o}')} Q(\mathbf{o}', \boldsymbol{a}'; \phi^-) - Q(\mathbf{o}, \boldsymbol{a}; \phi)]^2 \tag{6}$$

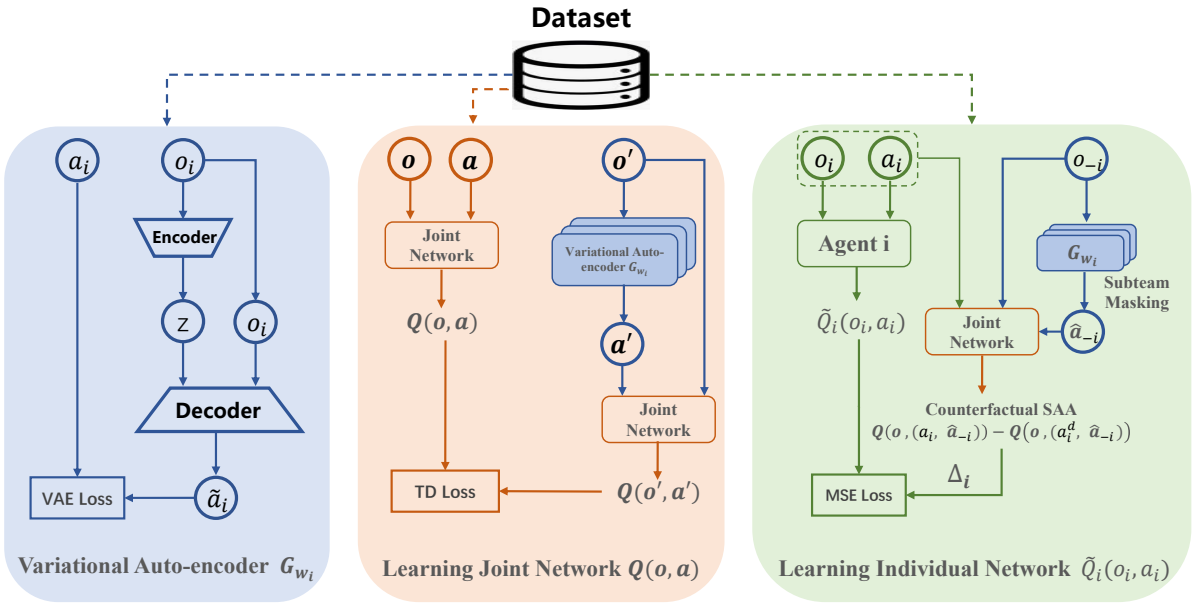



**Figure 2: The basic framework of our method. From left to right, we show the training processes of the VAE model, joint value network and individual value network. The losses of VAE, TD and MSE are given by Equations 3, 6 and 9 respectively.**

where $\boldsymbol{\pi}$ is the joint policy, $\phi$ are the parameters of the joint Q-function, and $\phi^-$ are the previous ones.

To avoid selecting OOD actions, we need to constrain the joint policy $\boldsymbol{\pi}$ in the dataset $\mathbf{B}$. Specifically, we use a population with $N$ individuals VAEs: $[G_{w^i}]_{i=1}^N$, one for each agent. Then, a set of candidate joint actions are generated by sampling from the VAEs and selecting the joint actions with the highest joint values according to the estimation of the joint value function.

Now, we have learned a joint policy $\boldsymbol{\pi}$. If the other agents follow the joint policy $\pi_{-i}$ except agent $i$, we have:

$$Q_i(o_i, a_i; \pi_{-i}) \approx Q(\mathbf{o}, \boldsymbol{a}; \phi) \tag{7}$$

where $o_i \in \mathbf{o}, a_i \in \boldsymbol{a}$ and $a_{-i} = \pi_{-i}(o_{-i})$. Note that following the joint policy is only possible in centralized training. Here, we use it to approximate the other agents' policy and learn a best-response to fit it. In addition, the best-response can be measured by the $Q(\mathbf{o}, \boldsymbol{a}; \phi)$ in centralized training.

As aforementioned, we aim to address the issue introduced by datasets collected by uncoordinated behavior policies. Inspired by the Shapley value [25], we split the other agents except $i$ into a set of subteams $ST(i)$. For example, given a team of 3 agents $\{1, 2, 3\}$, $ST(1) = \{\emptyset, \{2\}, \{3\}, \{2, 3\}\}$. Now, we can sample a subteam: $st \sim ST(i)$ and mask the actions of the agents in the subteam $st$ with the default actions. The agents that are not in the subteam $st$ follow the joint policy $\boldsymbol{\pi}$. Note that all the other agents follow the joint policy if $st = \emptyset$. By doing so, we maintain a distribution over the other agents' policies. The sampling of $\hat{\pi}_{-i} \sim \hat{\Pi}_{-i}$ can be done by: first sampling a subteam $st \sim ST(i)$ and masking the actions of the agents in the subteam $st$ with the default actions.

For datasets generated by uncoordinated behavior policies, there are possible miscoordinations in the joint policy. In the cases of miscoordination, it is critical to reason the agents' contributions and this cannot be done without considering the complex interdependency among the agents. By masking the actions of the agents in a subteam, we get a feedback about their contributions to the whole team. Then, we leverage the *counterfactual sample-average approximation* with the *subteam masking* to compute the agent $i$'s contributions to the whole team with the joint value function:

$$\Delta_i = \frac{1}{M} \sum_{m=1}^{M} [Q(\mathbf{o}, (a_i, \hat{a}_{-i}^m)) - Q(\mathbf{o}, (a_i^d, \hat{a}_{-i}^m))] \tag{8}$$

where $M$ represents the times of sampling and $\hat{a}_{-i}^m$ denote the other agents' actions sampled in $m$-th time. Given the individual contributions of agent $i$, we train the value network $\tilde{Q}_i$ to approximate it by minimizing the loss:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{(o_i,a_i)\sim B_i} \left[ (\tilde{Q}_i(o_i, a_i; \theta_i) - \Delta_i)^2 \right] \tag{9}$$

where $\theta_i$ denotes the parameters of the $\tilde{Q}_i$.

In addition, if some transitions are missing in the joint dataset $\mathbf{B}$, the estimation of the agents' contributions is inaccurate. To handle this, we borrow ideas from the clipped double Q-learning [8]. Specifically, we modify the joint values as a combination of the two joint Q-functions $\{Q(\cdot; \phi^1), Q(\cdot; \phi^2)\}$, with different initialization and a higher weight on the minimum one:

$$Q(\mathbf{o}, \boldsymbol{a}) = \lambda \min_{k=1,2} Q(\mathbf{o}, \boldsymbol{a}; \phi^k) + (1 - \lambda) \max_{k=1,2} Q(\mathbf{o}, \boldsymbol{a}; \phi^k) \tag{10}$$

where $\lambda \in [0, 1]$ is the coefficient to control over how heavily uncertainty is penalized. Intuitively, if the necessary transitions are in $\mathbf{B}$, the two joint-value functions should be well trained and equal to each other. If some transitions are missing, the minimum joint-value function is favorable with higher weight to reduce overestimations.

**Algorithm 1** Offline Multi-Agent Reinforcement Learning

**Input:** Datasets $[B_i]_{i=0}^N$, training step $T$, the agents number $N$, number of sampled actions n.
**Initialization:** The joint value networks $Q(\cdot;\phi^1)$, $Q(\cdot;\phi^2)$ with random parameters, and the target $\phi'^1, \phi'^2 = \phi^1, \phi^2$; The individual network $\tilde{Q}_i(\cdot;\theta^i)$ of agent $i$, the individual VAEs $G_{w^i} = \{E_{w_1^i}, D_{w_2^i}\}$ with random parameters for $i \in [1, N]$.

1: Merge the joint dataset $\mathbf{B} = [B_i]_{i=1}^N$.
2: **for** $t = 0$ **to** $T$ **do**
3: Sample mini-batch transitions $\langle \mathbf{o}, \boldsymbol{a}, r, \mathbf{o}' \rangle$ from $\mathbf{B}$
4: Train VAEs according to Eq. 3
5: Sample $n$ joint actions:
$$\boldsymbol{a} = [a_i^j]_{i=1}^N, \left\{a_i^j \sim G_{w^i}(o_i)\right\}_{j=1}^n.$$
6: Set target value $y = r + \gamma \max_{\boldsymbol{a}} Q(\mathbf{o}', \boldsymbol{a}; \phi^-)$
7: Update the joint value network according to Eq. 6
8: **for** each agent $i = 1$ **to** $N$ **do**
9: Sample $M$ subteams from $ST(i)$
10: Masking the agents' actions in the subteams
11: Update $\theta_i$ according to Eq. 9
12: **end for**
13: Update the target networks: $\phi'^1 = \phi^1$, $\phi'^2 = \phi^2$
14: **end for**

## 3.3 Implementation Details

The main procedures are outlined in Algorithm 1, and the framework of our method is shown as Figure 2. Specifically, we first consider the Dec-POMDP as a joint MDP. Given this, we use the VAE $G_{w^i}$ to model the action distribution of agent $i$, which is constraint in the dataset. The VAE is capable of sampling the action given the observation $o_i$. We can learn the VAE by the VAE loss (Lines 3-4), which contains the reconstruction loss and KL-divergence term according to the distribution of the latent vectors. Then, we learn the joint Q-function $Q(\mathbf{o}, \boldsymbol{a})$ (Lines 5-7). In the target value, we generate a set of candidate joint actions by sampling from the population VAEs and selecting the joint actions with the highest joint values according to the estimation of the joint value function. With the target value, we train the joint value network $Q(\mathbf{o}, \boldsymbol{a})$ with the TD loss function as shown in Eq. 6.

Next, we learn the individual value network $\tilde{Q}_i$ with the counterfactual sample-average approximation with subteam masking. In lines (9-10), we sample $M$ subteams and mask the actions of the agents in the subteams. By doing so, we maintain a set of candidate policies for the other agents, which can be used in the counterfactual SAA. Finally, we optimize each agent $i$'s policy given the sampled policies of the other agents (Line 11). Note that we set the default actions as 0 in the subteam masking. Same as the Shapley Value, we estimated the agent $i'$ contributions by masking the actions of the agents in a subteam with the default actions. The default action for agent $i$ should be equivalent to a team without the presence of agent $i$. In practice, we set the default actions as a vector 0 in the Multi-Agent MuJoCo, which means the default actions do not change the velocity and position of the motors controlled by the agent. We set the default actions as *noop* in the StarCraft II, which means that the agent takes no action.

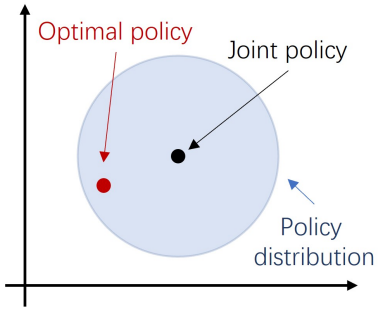



**Figure 3: Illustration of Policy Distribution**

## 3.4 Discussion

To better understand our work, we illustrate the ideal of policy distribution in Figure 3. Given a dataset, we can learn a joint policy by treating the Dec-POMDP as a big POMDP. However, the decentralized optimal policy of the Dec-POMDP is not necessarily very close to the joint policy. This mainly depends on how tight the coordination is in the domain. For example, if there is no coordination required and the agents can act independently, the decentralized optimal policy will be equal to the joint policy. If the dataset is collected by some uncoordinated behavior policies, it is highly likely that the joint policy is far away from the optimal policy because the dataset may contain very few coordination experiences. In this case, learning some decentralized policies to fit the joint policy will not perform well. In this paper, we maintain a distribution over the other agents' policies. This is done by sampling a subteam and masking their actions to create a population of policies around the joint policy. Hopefully, the optimal policy is within or near this population so it can be sampled. Based on this, each agent's policy can be optimized accordingly. This can also improve the generalization of each agent's policy.

# 4 EXPERIMENTS

We empirically evaluate our method on two multi-agent domains: 1) multi-agent MuJoCo [1] that is a continuous control problem with tight coordination, and 2) StarCraft II micromanagement [2] that is a challenging discrete control benchmark with complex tasks. Note that they are challenging multi-agent problems widely used in the multi-agent RL literature. In particular, multi-agent MuJoCo represents multi-agent robot control where interaction with the physical world is expensive and risky. Both of them require learning from offline datasets collected by some behavior policies. Finally, we conduct ablation studies on the multi-agent MuJoCo to better understand the benefit of our key techniques.

## 4.1 Multi-Agent MuJoCo

To test tight coordination and challenging tasks, we adopt the multi-agent MuJoCo environments [22], which extend the high-dimensional MuJoCo locomotion tasks to multi-agent settings. Specifically, several agents independently control one or more joints of a robot simulated by MuJoCo. Hence, each agent's observation space

[1] https://github.com/schroederdewitt/multiagent$_$mujoco
[2] https://github.com/oxwhirl/smac

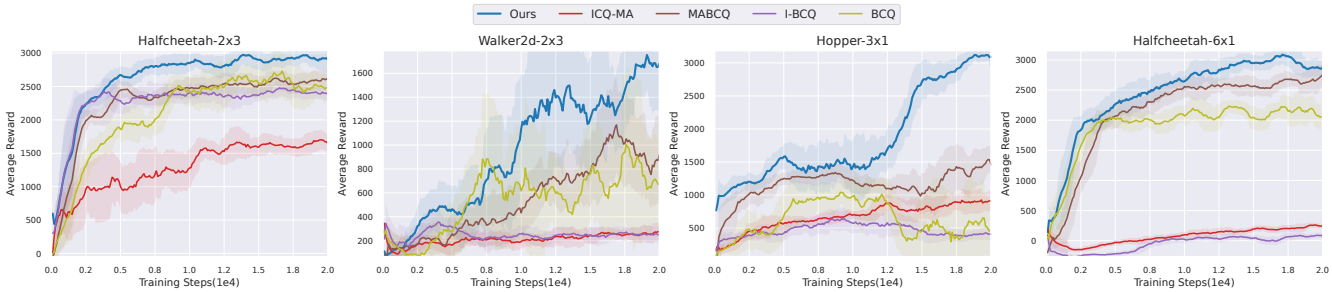



**Figure 4: Performance of our method and the baselines in multi-agent MuJoCo domains: HalfCheetah, Walker, Hopper. The brackets indicate (number of agents × controled joints per agent). The learning curves are plotted based the mean and standard deviation of three runs with difference random seeds.**

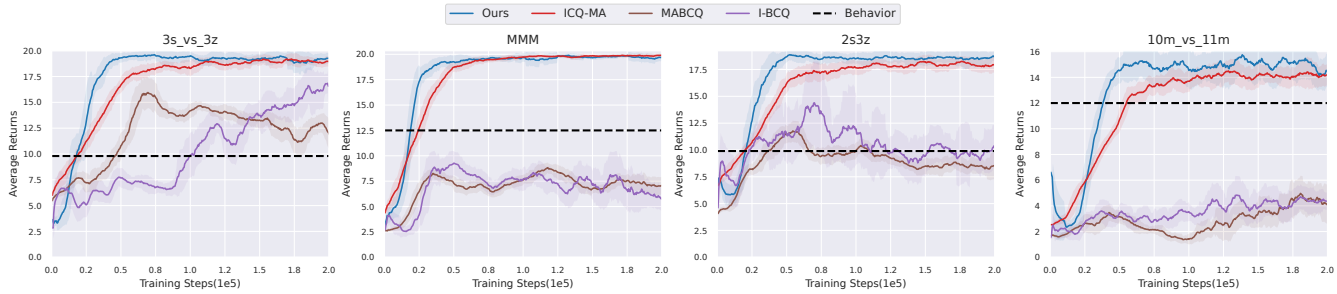



**Figure 5: Performance in StarCraft II with the offline datasets. The learning curves are also plotted based the mean and standard deviation of three runs with different random seeds.**

and action space in multi-agent MuJoCo are specified by the local observations or actions of the motors controllable by that agent. All multi-agent MuJoCo environments are configured according to the default configuration of multi-agent MuJoCo, where each agent can also observe both the velocity and position of its own body parts. Apart from that, the tasks themselves are identical to the original MuJoCo tasks. Therefore, the rewards are given in the same way as the original MuJoCo tasks and shared for all agents in multi-agent MuJoCo. At each time step, they get their own observations on those joints controlled by them and receive a reward based on the robot's performance on the tasks.

**Datasets.** For each domain, we first ran SAC [10] to obtain two behavior policies: 1) the *medium-policy* is recorded in the middle of the training process, and 2) the *expert-policy* is the final policy. Then, we followed the common procedure of offline RL and collected each dataset $B_i$ by running the environment with agent $i$ and the others choosing either the medium-policy or expert-policy. For example, agent 1 follows the medium-policy or expert-policy, while agent 2 follows the medium-policy or expert-policy. Each combination produced equally one-fourth of the trajectories. Totally, each dataset $B_i$ for agent $i$ contains 1000 trajectories with the length of 1000, i.e. $10^6$ of the overall transitions. Note that different from the dataset in D4RL [6] generated by a trained SAC agent, we used uncoordinated datasets generated by the mixed-levels policy of multiple agents.

**Baselines.** We compare our approach with the following methods: 1) **BCQ** [9]: applying centralized BCQ directly to multi-agent and learning joint policies; 2) **I-BCQ**: independent BCQ where each agent $i$ is trained independently on the dataset $B_i$ with BCQ; 3) **MABCQ** [12]: multi-agent BCQ where re-weights the offline transition dynamics by increasing the transition with high-value and normalizing the biased transition dynamics; 4) **ICQ-MA** [34]: the leading offline multi-agent RL method that decomposes the joint policy using QMIX. Note that both the joint BCQ and MABCQ require the joint observations of all agents for execution.

**Results.** As shown in Figure 4, our method substantially outperformed all the compared baselines in all the tested tasks, in terms of convergence speed, learning stability, and policy quality. BCQ does not get good performance because the centralized methods have difficulty handling the uncoordinated. Single-agent offline RL methods (i.e., BCQ) failed to learn an effective policy because they are not able to deal with the multi-agent issues. Surprisingly, ICQ-MA also performed poorly in our tests. Because ICQ-MA is based on only trusting seen state-action pairs in the dataset and heavily relies on the quality of the datasets, which shows poor performance in the uncoordinated datasets. In particular, the Walker and Hopper domains require very tightly coupled coordination, otherwise, the task will terminate immediately. Here, even MABCQ does not achieve good results due to the mismatch of agents' behavior policies in the dataset. Those baselines do not perform well in the uncoordinated datasets, due to they do not know who should be blamed for the failure of the tasks. Against this issue, our method applies counterfactual sample-average approximation to learn each

agent's local value function and marginalize the influence of the other agents, which can effectively learn the coordinate policies with a given dataset collected by uncoordinated behavior policies. The evaluation performance of all the methods is summarized in Table 2. As shown, our method significantly outperforms baseline methods in the uncoordinated datasets, and most of the coordination datasets split from D4RL. In the ablation studies, we will analyze the effect of different quality of the datasets (uncoordinated vs coordination) on all the methods in more detail.

## 4.2 StarCraft II

The StarCraft II micromanagement tasks [24], which involve combat between two armies of units, were proposed to study decentralized multi-agent control. In the tasks, a group of learning agents control the allied team to beat the enemy team, controlled by built-in hand-crafted heuristics. Specifically, at each time step, each agent receives local observations within its field of view. Then, they perform their respective actions that are integrated into the joint action, which is called macro-actions in the game. Meanwhile, the agents receive an identical joint reward from the environment ( i.e., 10 points for killing each opponent, and 200 points for killing all opponents). In addition, the cumulative return in an episode is normalized to $[0, 20]$.

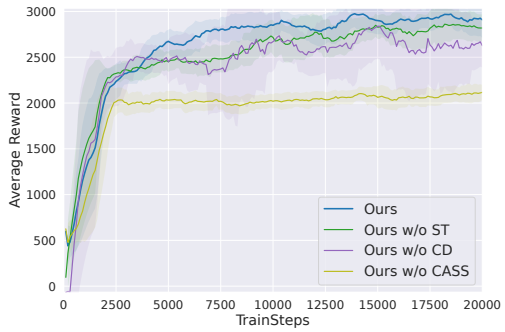
**Datasets.** We conducted our experiments in four maps: 3 Stalkers vs 3 Zealots (3d vs 3z), 1 Medivac & 2 Marauders & 7 Marines (MMM), 2 Stalkers & 3 Zealots (2s3z) and 10 Marines vs 11 Marines (10m vs 11m). In more detail, we use the datasets in [34], which contains 3000 non-expert or multi-source trajectories.

**Baselines.** We compared our approach against the following methods: 1) **I-BCQ** [9]: independent BCQ methods; 2) **MABCQ** [12]: multi-agent BCQ where re-weights the offline transition dynamics by increasing the transition with high-value and normalizing the biased transition dynamics.; 3) **ICQ-MA** [34]: the leading offline multi-agent RL method that decomposes the joint policy using QMIX. Here, we did not test joint BCQ because this domain is not designed for centralized control.

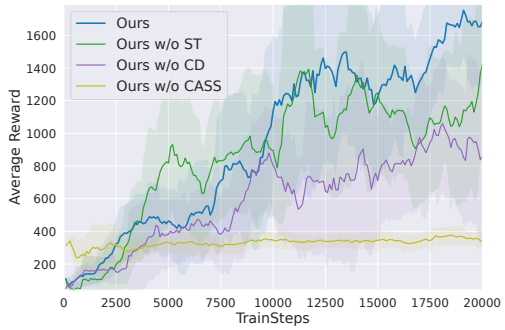
**Results.** As shown in Figure 5, our method outperformed all the compared methods and achieves good results in all the maps. As expected, single-agent offline RL methods (I-BCQ) fail to learn a policy that successfully defeats the enemy. The multi-agent offline RL approaches (MABCQ) also performed poorly due to the lack of cooperation mechanism which is important for the StarCraft domain. ICQ-MA and ours achieved satisfactory performance thanks to their respective cooperation mechanisms. Among them, our method achieves better performance. The test win rates of all the methods in the four maps are summarized in Table 1. In evaluation, our method outperforms all baselines and achieves state-of-the-art performance in most of the maps.

**Table 1: Comparison of test win rates (%) in StarCraft II micromanagement tasks. The test win rates is the percentage of these test episodes in which the method defeats all enemy units within the limit time. (best values are in bold)**

| Map | I-BCQ | MABCQ | ICQ-MA | Ours |
|---|---|---|---|---|
| 3s vs 3z | 58 | 6 | **83** | 81 |
| MMM | 4 | 0 | 89 | **93** |
| 2s3z | 8 | 1 | 61 | **71** |
| 10m vs 11m | 0 | 0 | 21 | **33** |




(a) HalfCheetah (2×3)




(b) Walker2d (2×3)

**Figure 6: Ablation experiments on the key components of our method in the multi-agent MuJoCo domains.**

## 4.3 Ablation Studies

We investigate the effect of each component in our method and the quality of the datasets with ablation experiments.

*4.3.1 Counterfactual SAA with Subteam Masking.* We conducted ablation studies of our methods to test the effectiveness of counterfactual SAA (CSAA). As shown in Figure 6, the method without CSAA (**Ours w/o CSAA**) performs significantly worse, which confirms the usefulness of our method. In addition, our method without subteam masking (**Ours w/o ST**) also works well in the multi-agent MuJoCo domains. This is not surprising because the two agents in these domains contribute more or less equally to the overall tasks. The advantage of the proposed method becomes apparent in the Walker domain, where each leg has different contributions to the walking pace. Since our method takes more comprehensive

**Table 2: Comparison average score of different quality of the datasets. The multi-agent MuJoCo datasets represent uncoordinated datasets and are generated by mixed levels of agents, such as agent 1 following the medium-level policy and agent 2 following the expert-level policy. The multi-agent variation of the D4RL datasets represent coordinated datasets. (best values are in bold).**

| Dataset | Behavior Policy | Environment | BCQ | I-BCQ | MABCQ | ICQ-MA | Ours |
|---|---|---|---|---|---|---|---|
| Multi-Agent MuJoCo | Medium-Expert | HalfCheetah-2x3 | 2465.1 | 2401.4 | 2613.6 | 1648.6 | **2937.1** |
| | | Hopper-3x1 | 408.2 | 402.8 | 1364.3 | 923.2 | **3077.9** |
| | | Walker2d-2x3 | 590.5 | 245.6 | 986.5 | 253.1 | **1589.9** |
| D4RL (Multi-Agent Variation) | Medium-Medium | HalfCheetah-2x3 | 4296.1 | 4466.5 | 4341.0 | 4313.7 | **4496.8** |
| | | Hopper-3x1 | 1026.5 | 922.5 | 927.7 | 984.3 | **2309.6** |
| | | Walker2d-2x3 | **1148.5** | 383.6 | 539.3 | 242.0 | 1080.8 |
| D4RL (Multi-Agent Variation) | Expert-Expert | HalfCheetah-2x3 | 10848.9 | 5669.3 | 10561.6 | 8480.0 | **11580.1** |
| | | Hopper-3x1 | **3624.4** | 1403.6 | 2544.2 | 2870.0 | 2428.4 |
| | | Walker2d-2x3 | 3298.8 | 834.4 | 2921.5 | 910.8 | **3902.5** |

consideration of the individual contributions of each agent in the entire team, it achieves better performance and stability.

*4.3.2 Clipped Double Q-learning.* We investigate the effect of the clipped double Q-learning [8] in our method. As shown in Figure 6, without it (**Ours w/o CD**), the overall performance dropped sharply, especially in the Walker2d environment. As aforementioned, this method reduces the overestimation bias and implicitly penalizes uncertainty due to missing transitions. This is reflected empirically that our method benefits substantially from this overestimation reduction technique.

*4.3.3 Quality of Datasets.* As aforementioned, the tested baselines do not perform well in the uncoordinated datasets. Here, we perform a further study on those methods with varying data quality. Specifically, we split the D4RL datasets[3] to create the multi-agent counterpart. In more detail, the action from the single-agent datasets in D4RL is split into several individual actions corresponding to the controlled joints. For example, the action space in HalfCheetah is a 6-dimensional vector, which is split into two 3-dimensional vectors for HalfCheetah-2x3. As shown in Table 2, splitting the action from D4RL datasets will generate easier datasets for offline MARL because the agents' behavior policies are coordinated and consistent. As we can see from the table, our method outperformed all the baselines for the uncoordinated datasets (Multi-Agent MuJoCo), even better than the centralized approach (BCQ). For the coordination datasets (D4RL), our method achieved the best performance compared to the decentralized baselines.

# 5 RELATED WORK

## 5.1 Offline Reinforcement Learning

Offline RL can broadly be categorized as: policy-constraint methods and conservative methods. The basic idea of policy-constraint methods is to constrain the learned policy to be close to the behavior policy. It can be implemented via an explicit action constraint [7, 9], using an implicit regularization [15, 32], or adding an uncertainty weight to the policy improvement objective [33, 36]. The basic idea of conservative methods is to learn a lower-bound or conservative Q-function for OOD actions [11, 16, 35]. For example, CQL [16] adds a regularizer to penalize the Q-function of OOD actions and encourage Q-function for state-action in the dataset to be large.

[3]https://github.com/rail-berkeley/d4rl

## 5.2 Multi-Agent Reinforcement Learning

To date, most multi-agent RL are online methods requiring agents to interact with the environment. For value-based methods, VDN [28] and QMIX [23] factor the joint Q-function into individual Q-function via a simple summation or monotonic mixing function respectively. For representational limitation, QTRAN [26] and QPLEX [29] learn an unrestricted joint Q-function to obtain a richer class of Q-functions. For the policy-based methods, MADDPG [19] proposes to use a single centralized critic and decentralized actor for each agent. COMA [5] and Shapley-COMA [18] employ centralized critic to generate a counterfactual advantage baseline to guide the learning of local policies. Generally, it is nontrivial to apply online multi-agent RL to offline settings, which is trained with only the pre-collected dataset.

# 6 CONCLUSION

In this paper, we propose an offline multi-agent RL that effectively learns coordinated policies with datasets collected by uncoordinated behavior policies. Specifically, we treat the Dec-POMDP as a single-agent POMDP from the perspective of each agent and learn the best-response policy given a distribution over the other agents' policies. In more detail, we use the counterfactual sample-average approximation to reason the agent's contribution to the whole team considering the complex interdependency among the agents, and maintain the policy distribution by masking the other agents' policies. The experimental results show that our method could learn to coordinate with the others with datasets collected from uncoordinated behavior policies, and outperforms several state-of-the-art methods in the multi-agent MuJoCo and the StarCraft II micromanagement tasks. In the future, we plan to extend our framework to more complex multi-agent systems, e.g., partially observable Markov games, where the agents are self-interested.

## ACKNOWLEDGMENTS

This work was supported in part by the Major Research Plan of the National Natural Science Foundation of China (Grant No. 92048301) and Anhui Provincial Natural Science Foundation (Grant No. 2208085MF172).

## REFERENCES

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 104–114.

[2] Christopher Amato. 2018. Decision-Making Under Uncertainty in Multi-Agent and Multi-Robot Systems: Planning and Learning.. In *IJCAI*. 5662–5666.

[3] Duncan S Callaway and Ian A Hiskens. 2010. Achieving controllability of electric loads. *Proc. IEEE* 99, 1 (2010), 184–199.

[4] Shaofei Chen, Feng Wu, Lincheng Shen, Jing Chen, and Sarvapali D. Ramchurn. 2015. Multi-Agent Patrolling under Uncertainty and Threats. *Public Library of Science (PLOS ONE)* 10(6) (2015), e0130154. https://doi.org/10.1371/journal.pone.0130154

[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[6] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[7] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021).

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

[9] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*. PMLR, 2052–2062.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[11] Qiang He, Xinwen Hou, and Yu Liu. 2022. Popo: Pessimistic offline policy optimization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4008–4012.

[12] Jiechuan Jiang and Zongqing Lu. 2021. Offline decentralized multi-agent reinforcement learning. *arXiv preprint arXiv:2108.01832* (2021).

[13] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[14] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12, 2 (2002), 479–502.

[15] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

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

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

[18] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. 2021. Shapley Counterfactual Credits for Multi-Agent Reinforcement Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 934–942.

[19] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[20] Jinming Ma and Feng Wu. 2020. Feudal Multi-Agent Deep Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*. Auckland, New Zealand, 816–824.

[21] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, Vol. 3. Citeseer, 705–711.

[22] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems* 34 (2021).

[23] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.

[24] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[25] Lloyd S Shapley. 1953. A value for n-person games, Contributions to the Theory of Games, 2, 307–317.

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

[27] Haoyuan Sun and Feng Wu. 2023. Less Is More: Refining Datasets for Offline Reinforcement Learning with Reward Machines. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*. London, United Kingdom.

[28] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[29] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. [n.d.]. QPLEX: Duplex Dueling Multi-Agent Q-Learning. ([n. d.]).

[30] Marco A Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 1151–1158.

[31] Feng Wu, Sarvapali D. Ramchurn, and Xiaoping Chen. 2016. Coordinating Human-UAV Teams in Disaster Response. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. New York, USA, 524–530.

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

[33] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. 2021. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140* (2021).

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

[35] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. Combo: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems* 34 (2021).

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