Learning and Exploiting Shaped Reward Models for Large Scale Multiagent RL

Arambam James Singh, Akshat Kumar, Hoong Chuin Lau
School of Computing and Information Systems
Singapore Management University
{arambamjs.2016,akshatkumar,hclau}@smu.edu.sg

Abstract

Many real world systems involve interaction among large number of agents to achieve a common goal, for example, air traffic control. Several model-free RL algorithms have been proposed for such settings. A key limitation is that the empirical reward signal in model-free case is not very effective in addressing the *multiagent credit assignment* problem, which determines an agent's contribution to the team's success. This results in lower solution quality and high sample complexity. To address this, we contribute (a) an approach to learn a differentiable *reward model* for both continuous and discrete action setting by exploiting the collective nature of interactions among agents, a feature commonly present in large scale multiagent applications; (b) a *shaped* reward model analytically derived from the learned reward model to address the key challenge of credit assignment; (c) a model-based multiagent RL approach that integrates shaped rewards into well known RL algorithms such as policy gradient, soft-actor critic. Compared to previous methods, our learned reward models are more accurate, and our approaches achieve better solution quality on synthetic and real world instances of air traffic control, and cooperative navigation with large agent population.

1 Introduction

In many real world applications, large group of agents interact with each other to achieve a common goal. For example, aircraft coordination in busy air space is required to ensure certain minimum separation among aircrafts (Brittain and Wei 2019), in maritime traffic management (Singh et al. 2019; Singh, Kumar, and Lau 2020) where the goal is to reduce congestion for navigation safety, and vehicle fleet optimization problems (Varakantham, Adulyasak, and Jaillet 2014). Such problems can be modeled using cooperative multiagent reinforcement learning (MARL). There are many recent successes of single agent RL methods (Mnih et al. 2015; Silver et al. 2017). However, direct extensions of single agent approaches to multiagent setting is challenging due to reasons such as multiple agents learning and acting simultaneously, and scalability with increasing number of agents resulting in exponential joint state and action spaces. As a result, several recent MARL approaches such as multiagent actor-critic (Lowe et al. 2017), counterfactual policy gradient (Foerster et al. 2018), multiagent Q-learning (Rashid et al. 2018), and actor-attention critic (Iqbal and Sha 2019) are limited to relatively small number of agents.

Specific to cooperative multiagent system with shared rewards is the critical problem of *multiagent credit assignment* (Agogino and Tumer 2004; Bagnell and Ng 2006). Given the complex nature of interactions among agents, the individual contribution of each agent (and its specific actions) to the team reward is unclear. This makes multiagent learning much more challenging and sample inefficient, specially for policy gradient techniques (Williams 1992; Peshkin et al. 2000) as the reward signal used to train agent policies has high noise due to other agents' actions.

Shaped rewards: *Shaped rewards* have been proposed to address the problem of multiagent credit assignment. Difference rewards (DRs), computed as the difference between the system reward and a *counterfactual* reward when the particular agent's impact is removed from the system, quantify the contribution of an agent to the system reward (Agogino and Tumer 2008; Colby et al. 2016). However, computing DRs is challenging as it requires access to the actual reward model (not available in a model free setting), or perform additional simulations, which is computationally challenging. To address this, previous approaches have used function approximation to learn the reward model (and DRs) from empirical returns (Proper and Tumer 2013; Colby et al. 2016). Their key limitations include being specialized to a particular domain and may not be extensible to general multiagent settings (Proper and Tumer 2012, 2013), and based only on agents' local observations (Colby et al. 2016), which makes the learned reward model inaccurate for large agent population and when agents are tightly-coupled.

Collective multiagent systems: Many multiagent applications have additional structure that can be exploited for accurate learning of reward models. In several large scale multiagent systems, an agent's behavior is mainly influenced by the *aggregate* information about neighboring agents rather than their identities (Sonu, Chen, and Doshi 2015; Robbel et al. 2016; Subramanian et al. 2020). For example, in taxi fleet optimization, the movement behavior of a taxi agent is primarily influenced by the total demand and the count of other taxis present in city zones (Varakantham, Adulyasak, and Jaillet 2014; Nguyen, Kumar, and Lau 2017a). In air and maritime traffic control, most of the agents can be considered as homogeneous (or belonging to a small number of types) (Brittain and Wei 2019; Singh, Kumar, and Lau 2020). Previous reward model learning approaches do not specifically exploit such symmetries, which our work aims to do.

Specialized MARL approaches have been developed for such collective settings such as mean field RL (Yang et al. 2018; Subramanian et al. 2020). Mean-field RL (Subramanian et al. 2020) scales well for large agent setting, but does not explicitly addresses credit assignment or learns reward models. (Hüttenrauch, Šošić, and Neumann 2018) is another approach for large agent settings where they learn agent observation embeddings, whereas our focus is on the complementary aspect of learning DRs for credit assignment. (Sun, Shen, and How 2020) propose an attention mechanism to learn a communication graph, which is a different settings than ours. Collective decentralized POMDPs (Dec-POMDPs) have also been proposed for decision making in homogeneous agent population (Nguyen, Kumar, and Lau 2017b, 2018), and applied in applications such as vehicle fleet optimization and multiagent patrolling. Their solution methods do not specifically learn reward models, but perform credit assignment at the level of action-value function (or Q-values), and the action-value function itself is learned from empirical returns. A key bottleneck in such methods is that credit assignment at the level of entire future expected reward-to-go is often more difficult than learning the one step (shaped) reward model. Such methods are not applicable to continuous action setting which we also target. Empirically, our approach worked better than such previous approaches for mean field RL and collective Dec-POMDPs.

Our key contributions include:

- An approach to learn a differentiable *reward model* for both continuous and discrete action setting by exploiting the collective nature of interactions among agents. Unlike previous reward learning methods, our approach is general, utilizes aggregate information over all the agents, and thus has better accuracy.
- A principled method to analytically compute shaped rewards from the reward model, without requiring any domain expertise or extra simulations. Resulting approach is scalable for large number of agents, and makes effective multiagent credit assignment possible.
- A model-based RL approach that uses learned shaped rewards instead of empirical returns, and generalizes well known RL algorithms for discrete and continuous actions such as policy gradient and soft-actor critic (Haarnoja et al. 2018) to the collective multiagent setting.

We also highlight that reward models can be learned simultaneously with the policy learning. We evaluate our approach on two domains—air traffic control (Brittain and Wei 2019), and cooperative navigation in a multiagent particle environment (Iqbal and Sha 2019). For the air traffic domain, in addition to synthetic instances, we obtain historical air traffic data over the European airspace containing ∼100 million data points showing the trajectory and other navigation properties (e.g., altitude, speed) over 30 days. We use the realistic BlueSky Air Traffic Control Simulator as our RL simulator (Hoekstra and Ellerbroek 2016). We compare against approaches that are specifically designed to exploit homogeneous agent population (Yang et al. 2018; Subramanian et al. 2020; Nguyen, Kumar, and Lau 2018), and specialized approaches for air traffic control (Brittain and Wei 2019), and cooperative navigation (Iqbal and Sha 2019). We show that our approach outperforms such previous methods.

2 Model Definition

We describe a cooperative multiagent system formally as a Dec-POMDP (Oliehoek, Amato et al. 2016), $G = \langle \boldsymbol{S}, \boldsymbol{A}, P, O, Z, M, \gamma, r\rangle$. There are M agents in the system. At each time step, an agent m can be in any of the state $s \in S$ and takes action $a \in A$. The joint state and action spaces are denoted as $\boldsymbol{S} = \times_{m=1}^{M} S$ and $\boldsymbol{A} = \times_{m=1}^{M} A$ respectively. The reward function $r(\boldsymbol{s}, \boldsymbol{a}) : \boldsymbol{S} \times \boldsymbol{A} \to \mathbb{R}$ is shared among agents (i.e., given to the agent team), and transition of environment from current state $\boldsymbol{s}$ to next state $\boldsymbol{s}'$ after taking the joint-action $\boldsymbol{a}$ is according to state transition function $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \to [0, 1]$.

We consider a partially observable setting in which each agent m draws individual observation $z \in Z$ according to observation function $O(z, \boldsymbol{s}', \boldsymbol{a}) = \Pr(z|\boldsymbol{a}, \boldsymbol{s}')$ if the last action taken was $\boldsymbol{a}$ and the resulting joint-state was $\boldsymbol{s}'$. Each agent gets action a^m through a parameterized policy $\pi^m(a^m \mid \tau^m; \theta^m)$ which is a mapping from an action-observation history $\tau^m \in (Z \times A)^*$ to action a^m. For infinite-horizon problems, history τ^m can become very long, therefore, often a compact summary of τ^m is used (e.g., remember last k observations), or finite-state controllers are used which can compactly summarize an agent's obser-vaiton history, and their structure can also be learned using gradient based methods (Peshkin et al. 2000).

We use γ to denote the reward discounting. Let $R_t = \sum_{i=0}^{\infty} \gamma^i r_{i+t}$ be the discounted return. Using this, the joint value function V and action-value function Q are given as:

$$V(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{s}_{t+1:\infty}, \boldsymbol{a}_{t+1:\infty}}[R_t \mid \boldsymbol{s}_t] \tag{1}$$

$$Q(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{s}_{t+1:\infty}, \boldsymbol{a}_{t+1:\infty}}[R_t \mid \boldsymbol{s}_t, \boldsymbol{a}_t] \tag{2}$$

The objective is to find the joint-policy $\boldsymbol{\pi}$ that maximizes expected discounted return:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{s}_{0:\infty}, \boldsymbol{a}_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \mid \boldsymbol{\pi}\right] \tag{3}$$

Learning paradigm: In MARL, the environment receives a joint action $\boldsymbol{a}_t$ and transitions to a next state $\boldsymbol{s}_{t+1}$, and each agent receives a reward r_t and an observation z_{t+1}^m. We follow previous MARL approach of centralized learning and decentralized execution (Oliehoek, Spaan, and Vlassis 2008; Foerster et al. 2018) in which training is done centrally. That is, during learning, we assume access to extra information such as the joint-state of agents, which helps in accurate computation of different value functions and stabilize the learning. However, during the execution phase, each agent independently executes the policy based only on its local observation.

Aggregate Statistics: We assume that agents belong to a small number of types, and there are many more agents than

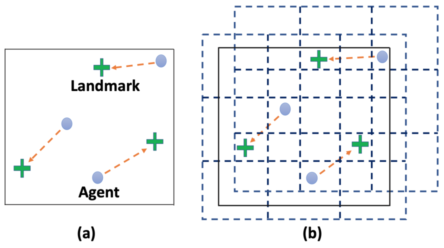


Figure 1: Discretization in Cooperative navigation. (a) shows continuous grid in which agents try to move close to landmarks. (b) shows state space discretization using two 3x3 tiles (in blue).

types. The key idea is to exploit the *aggregate* or *collective* nature of interactions among agents in several practical applications. Several previous works in multiagent decision making, such as anonymous and collective Dec-POMDPs, mean field RL (Varakantham, Adulyasak, and Jaillet 2014; Sonu, Chen, and Doshi 2015; Robbel et al. 2016; Subramanian et al. 2020; Nguyen, Kumar, and Lau 2017a, 2018; Yang et al. 2018; Subramanian and Mahajan 2019) also exploit this property for scalability. We assume that the agent type is also a component of the state. Since all agents share the same local state space S and action space A, there can be scenario where multiple agents are in same local state $s \in S$ and also taking same action $a \in A$. We introduce *count variables* which keep tracks of this aggregate information. Let $\langle \boldsymbol{s}_t, \boldsymbol{a}_t \rangle = \langle s_t^m, a_t^m \rangle_{m=1:M}$ be the joint discrete state-action pair for all agents at time t, where s_t^m and a_t^m are the random variables representing discrete state, action of agent m at time t.

- Let $\mathrm{n}_t(s) = \sum_{m=1}^{M} \mathbb{I}[s_t^m = s; \boldsymbol{s}_t]$, $\forall s \in S$, denote the state count variable that counts the number of agents in the same discrete state s at time t ($\mathbb{I}$ is the indicator function).
- Let $\mathrm{n}_t(s,a) = \sum_{m=1}^{M} \mathbb{I}[s_t^m = s, a_t^m = a; \boldsymbol{s}_t, \boldsymbol{a}_t]$, $\forall s \in S, \forall a \in A$, denote the state-action count variable that counts number of agents in same discrete state s and taking same action a at time t.

We denote $\mathbf{n}_t^S = \langle \mathrm{n}_t(s) \rangle_{\forall s}$ and $\mathbf{n}_t^{SA} = \langle \mathrm{n}_t(s,a) \rangle_{\forall s, \forall a}$ as the state and state-action count tables respectively, and $\mathbf{n}_t = \langle \mathbf{n}_t^S, \mathbf{n}_t^{SA} \rangle$ be the count table vector at time t.

Addressing continuous state space: It appears that the count-based aggregate statistic would not generalize when the underlying state space is continuous. Indeed, several MARL benchmarks do have continuous state space, such as multiagent particle environments (Lowe et al. 2017). Motivated by the use of tile coding for state aggregation in RL (Sutton and Barto 2018) and discretization of the state space in MDPs (Chow and Tsitsiklis 1991), we assume a mapping $\phi(s^m) : S \to U$ which maps the local state s^m into a discrete space $u^m \in U$. The mapping ϕ can be considered as a discretization mechanism, similar to tile-coding or grid based discretization. Although, discretization may require domain knowledge, but in many application domains, it can be fairly straightforward. For example, in air traffic control domain (Tumer and Agogino 2007) air space is discretized into sectors, in road traffic control (Wiering 2000), roads are divided into discrete cells, and in the multiagent particle environments (Lowe et al. 2017) such as cooperative navigation, we can use tile-coding to discretize the state space as shown in figure 1. Thus, the credit assignment techniques we develop are also applicable to continuous state spaces, and validated empirically on such problems.

3 Reward Model for Discrete Actions

In this section, we first show how to learn a function approximator for the system reward based only on reward signals from the simulator and by exploiting the aggregate nature of interaction among agents. We then develop techniques to analytically compute shaped difference rewards from such a reward approximator.

3.1 Learning System Reward Model

Before we can compute difference rewards, it is crucial to first learn an accurate approximator for the system reward. We address both the non-decomposable reward setting where there is a single global reward given to the entire agent team, and the decomposable reward setting when the global reward is decomposable among agents.

Non-decomposable rewards: In this setting, the simulator provides single numerical reward r to the agent team. We maintain a buffer $\mathcal{B}$ where each recorded sample ξ involves the tuple $\langle r_\xi, \mathbf{n}_\xi^{SA} \rangle$. The $\mathbf{n}_\xi^{SA}$ is the state-action count table given the joint state-action was $\langle \boldsymbol{s}, \boldsymbol{a} \rangle$ (defined in section 2), and reward signal r_ξ was provided by the simulator as a result of the joint action $\boldsymbol{a}$ taken in the joint state $\boldsymbol{s}$. Given the homogeneous agent population, the reward function does not depend on the identities of the agent, as also noted in previous models (Yang et al. 2018; Nguyen, Kumar, and Lau 2017a). Therefore, we learn a function approximator $r_{\mathbf{w}}(\mathbf{n}^{SA})$ that takes as input the state-action count table. The loss function can be defined as:

$$\mathcal{L}(\mathbf{w}) = \sum_{\xi \in \mathcal{B}} \left(r_\xi - r_{\mathbf{w}} \left(\mathbf{n}_\xi^{SA} \right) \right)^2 \qquad (4)$$

We can minimize the above loss using standard auto-differentiation libraries.

We also comment on the scalability of learning $r_{\mathbf{w}}$. The size of input to $r_{\mathbf{w}}$ is $|S| \times |A|$ (each input can be a normalized count value in $[0,1]$). Crucially, input vector's dimensions do not depend on the number of agents. Therefore, it is scalable to learn such a function approximator even for large agent settings. In several practical applications such as air traffic control, discrete states correspond to zones in the airspace (Brittain and Wei 2019), similarly, for maritime traffic, each discrete state corresponds to a navigable zone in the port area (Singh, Kumar, and Lau 2020). In such settings, number of zones are limited by the physical dimensions of the region in which agents move, and therefore, the input size $|S| \times |A|$ remains tractable. These are the settings where our approach would be more impactful.

Decomposable rewards: In many application domains, the global reward is decomposable. E.g., in the air traffic domain, if two aircrafts are closer than a threshold distance, then penalty is given to each aircraft (Brittain and Wei 2019). For such settings, we assume the global reward is decomposed into local reward received by each agent which depends on agent's local state and action, as well as the aggregate statistic of the agent population. Since agents are homogeneous, we can aggregate the reward over all the agents in a particular state-action (s, a) as:

$$r\left(\mathbf{n}_t^{SA}\right) = \sum_{s\in S}\sum_{a\in A} \mathrm{n}_t(s,a)\cdot \tilde{r}(s,a,\mathbf{n}_t^S) \quad (5)$$

where $\tilde{r}$ is the reward given to an agent. Corresponding to the reward structure in (5), the reward function approximator $r_{\mathbf{w}}$ can also be structured in a decomposable way:

$$r_{\mathbf{w}}\left(\mathbf{n}_t^{SA}\right) = \sum_{s\in S}\sum_{a\in A} \mathrm{n}_t(s,a)\cdot \tilde{r}_w(s,a,\mathbf{n}_t^S) \quad (6)$$

The loss function for training $\tilde{r}_w$ is given as:

$$\mathcal{L}(\mathbf{w}) = \sum_{\xi\in\mathcal{B}}\Big[\sum_{s\in S}\sum_{a\in A}\mathrm{n}_\xi(s,a)\Big(\tilde{r}_\xi(s,a,\mathbf{n}_\xi^S) - \tilde{r}_w(s,a,\mathbf{n}_\xi^S)\Big)\Big]^2 \quad (7)$$

Empirically, when we used the loss function in (7), the performance was not good, and it required several samples to train $\tilde{r}_w$. We then investigated another surrogate loss function that is an upper bound on (7) using the Cauchy-Schwarz inequality (details omitted).

$$\tilde{\mathcal{L}}(\mathbf{w}) = M\sum_{\xi\in\mathcal{B}}\sum_{s\in S, a\in A}\mathrm{n}_\xi(s,a)\Big(\tilde{r}_\xi(s,a,\mathbf{n}_\xi^S) - \tilde{r}_w(s,a,\mathbf{n}_\xi^S)\Big)^2 \quad (8)$$

This loss function provided much better results. Intuitively, the reason is that in (7), some $\tilde{r}_w$ can overestimate or underestimate the corresponding target $\tilde{r}_\xi$. When we do summation over all the states and actions, the loss may still appear small as different overestimates and underestimates may cancel out. However, this issue does not happen in the loss (8) as the loss term $(\tilde{r}_\xi(\cdot) - \tilde{r}_w(\cdot))^2$ is always positive, and is magnified by the count $\mathrm{n}_\xi(s,a)$.

3.2 Computing Difference Rewards

We now show how to compute difference rewards (DRs) from $r_{\mathbf{w}}$ in a model-free setting without requiring any extra simulations or domain expertise. Unlike previous approaches (Proper and Tumer 2013), our methods is not tied to a domain, and also explicitly utilizes aggregate information over all the agents, in contrast to methods that use only local information available to an agent (Colby et al. 2016). As a result, our method results in much better solution quality when combined with a policy optimization technique than such previous approaches.

Let s_t^m and a_t^m be the state and action of an agent m at time t. From the definition of difference rewards(Agogino and Tumer 2008; Colby et al. 2016),

$$D^m\left(s_t^m, a_t^m\right) = r(\boldsymbol{s}_t, \boldsymbol{a}_t) - r(\boldsymbol{s}_t^{-m}\cup d_s, \boldsymbol{a}_t^{-m}\cup d_a) \quad (9)$$

where, $\boldsymbol{s}_t^{-m} = \boldsymbol{s}_t \setminus s_t^m$ and $\boldsymbol{a}_t^{-m} = \boldsymbol{a}_t \setminus a_t^m$ are joint state and action without agent m. Let d_s and d_a denote default state and action for agent m. Intuitively, the difference provides the contribution (or credit) of the agent to the total reward. DRs are also aligned with the overall objective of a cooperative multiagent system. Therefore, if agents learn to optimize their DRs, it will also optimize the long term global reward as the second term in (9) always uses fixed state-action for an agent i. DRs provide a conceptual framework for credit assignment, and do not provide a concrete algorithm to compute them. In large multiagent systems, it is often intractable to exactly compute such shaped rewards. We therefore present several new techniques that can accurately estimate DRs from the learned reward model in large mutiagent systems.

For our homogeneous agent setting, using the state-action count table, $\mathbf{n}_t^{SA}$, derived from the joint state-action pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$, we can rewrite (9) as:

$$D^m(s_t^m, a_t^m) = r\left(\mathbf{n}_t^{SA}\right) - r\left(\mathbf{n}_t^{SA-(s_t^m,a_t^m)+(d_s,d_a)}\right) \quad (10)$$

where, $\mathbf{n}_t^{SA-(s_t^m,a_t^m)+(d_s,d_a)}$ is like a counterfactual state-action count table obtained by replacing the current state and action (s_t^m, a_t^m) of agent m with a default state and action (d_s, d_a). There can be many agents in the same state taking same action, and thus sharing the same DR value. Therefore, instead of computing DR for each agent individually, we can define DR for each state-action pair. This way of defining DRs is going to scale much better with the increasing number of agents. Let $(s \in S, a \in A)$ be any arbitrary state-action pair. Let $\mathcal{I}^{sa}$ and $\mathcal{I}^{d_s d_a}$ be an identity matrix with the same dimension as $\mathbf{n}_t^{SA}$ with all zero entries except value 1 at the index corresponding to (s, a) and (d_s, d_a) respectively. The difference rewards for state-action pair (s, a) is also given as:

$$D_t(s,a) = r\left(\mathbf{n}_t^{SA}\right) - r\left(\mathbf{n}_t^{SA} - \mathcal{I}^{sa} + \mathcal{I}^{d_s d_a}\right) \quad (11)$$

Recall that we do not have access to the actual reward function $r(\mathbf{n}_t^{SA})$ in a model free setting. Therefore, we replace $r(\mathbf{n}_t^{SA})$ with the learned reward function $r_{\mathbf{w}}$ from section 3.1. For stable training of $r_{\mathbf{w}}$, we also normalize the count variables with the total agent population, $\tilde{\mathbf{n}}_t^{SA} = \mathbf{n}_t^{SA}/M$. With reward function approximator, (11) becomes:

$$D_t(s,a) \approx r_{\mathbf{w}}\left(\frac{\mathbf{n}_t^{SA}}{M}\right) - r_{\mathbf{w}}\left(\frac{\mathbf{n}_t^{SA} - \mathcal{I}^{sa} + \mathcal{I}^{d_s d_a}}{M}\right) \quad (12)$$

Direct evaluation of above expression is computationally expensive as the argument matrix in the second term is different for different state-action pairs, and needs to be computed at each time step as $\mathbf{n}_t^{SA}$ keeps changing. Furthermore, as action-value function used to train the policy is learned from such DRs in our approach, the scale of policy gradients can be adversely affected by $r_{\mathbf{w}}$, specially during early training when $r_{\mathbf{w}}$ is inaccurate. This can lead to convergence to a poor quality solution. To address these issues, we develop a gradient based method that can analytically approximate DRs faster in a vectorized form using optimized autodiff libraries, and also uses only the gradients of $r_{\mathbf{w}}$. Empirically, we observed that our new method resulted in better quality than computing DRs using (12). As our focus is on settings with large agent population, we assume

$M \to \infty$. Under this assumption, the expression for approximate difference rewards (12) becomes:

$$D_t(s,a) \approx \lim_{M\to\infty}\left[r_{\mathbf{w}}\left(\frac{\mathbf{n}_t^{SA}}{M}\right) - r_{\mathbf{w}}\left(\frac{\mathbf{n}_t^{SA} - \mathcal{I}^{sa} + \mathcal{I}^{d_s d_a}}{M}\right)\right]$$

$$= \lim_{\Delta=1/M\to 0}\left[r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right) - r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA} - \Delta\cdot\left(\mathcal{I}^{sa} - \mathcal{I}^{d_s d_a}\right)\right)\right]$$

$$= -1\cdot\lim_{\Delta\to 0}\left[r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA} - \Delta\cdot\left(\mathcal{I}^{sa} - \mathcal{I}^{d_s d_a}\right)\right) - r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right)\right]$$

$$= -1\cdot(-\Delta)\left(\frac{\partial r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right)}{\partial\tilde{\mathbf{n}}_t^{SA}(s,a)} - \frac{\partial r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right)}{\partial\tilde{\mathbf{n}}_t^{SA}(d_s,d_a)}\right) \tag{13}$$

$$D_t(s,a) \approx \frac{1}{M}\cdot\left(\frac{\partial r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right)}{\partial\tilde{\mathbf{n}}_t^{SA}(s,a)} - \frac{\partial r_{\mathbf{w}}\left(\tilde{\mathbf{n}}_t^{SA}\right)}{\partial\tilde{\mathbf{n}}_t^{SA}(d_s,d_a)}\right) \tag{14}$$

where (13) was derived from the previous expression by using total differential. Total differential states that if $\boldsymbol{x}$ is a small vector, then $f(a+\boldsymbol{x}) - f(a) \approx \sum_{i=1}^{n}\frac{\partial f}{\partial x_i}x_i$. In our case, $\Delta\left(\mathcal{I}^{sa} - \mathcal{I}^{d_s d_a}\right)$ is analogous to $\boldsymbol{x}$ as Δ is a small number, and entries in different $\mathcal{I}$ vectors are either 0 or 1.

Once difference rewards are computed, it is possible to integrate them in a policy gradient algorithm (details in the supplement [1]). The resulting approach computes returns R_t using the learned difference rewards (rather than empirical rewards from the simulator). This results in a model-based RL where difference rewards provide a counterfactual value highlighting an agent's contribution to the team's reward. Furthermore, learning of the reward model and policy can be done simultaneously. As the learning proceeds, the reward model becomes more accurate, which in turn leads the convergence to a good solution quality.

4 Reward Model for Continuous Actions

We next show how a system reward model with continuous actions can be learned using aggregate variables. Then, using the learned reward model, we derive an expression for difference rewards, and show how DRs can be adapted to the soft actor-critic (SAC) algorithm (Haarnoja et al. 2018) for our setting. In several problems, the underlying continuous action space is often discretized. E.g., in the air traffic domain (Brittain and Wei 2019), the speed of the aircraft is discretized. Empirically we show that different levels of discretization can result in significant differences in the solution quality. Training a policy on the true underlying continuous action space helps alleviate such domain engineering issues.

4.1 Learning Reward Model

Let $\pi(a^m|s^m;\theta)$ denote the stochastic policy of agent m parameterized by θ. The policy π can be a Gaussian with mean and variance as output of a neural network parameterized by θ. A popular approach to address continuous actions is using the reparameterization trick, e.g. in the SAC algorithm, action a^m in state s_t^m is computed as (Haarnoja et al. 2018):

$$a^m = f_\theta(\epsilon^m; s^m) \tag{15}$$

where $f_\theta(\bullet)$ is a deterministic function, ϵ^m is an input noise sampled from some fixed distribution. Let $\boldsymbol{\epsilon} = \langle\epsilon^m,\ m = 1\!:\!M\rangle$ be input noise vector, $\boldsymbol{a} = \langle a^m = f_\theta(\epsilon^m; s^m), \forall m = 1:M\rangle$ be the joint action. As the joint action deterministically depends on $\boldsymbol{\epsilon}$, parameters θ, and state $\boldsymbol{s}$, we use an alternate notation $r_\theta(\boldsymbol{s},\boldsymbol{\epsilon})$ for the joint reward $r(\boldsymbol{s},\boldsymbol{a})$. This interpretation would be useful later.

Partitioning the noise space: Extending count table based techniques presented in the previous section is non-trivial for continuous actions as no two agents will take the same action under a stochastic policy. We also do not wish to discretize the action space to avoid domain engineering overhead. The solution to using count based methods for continuous actions lies in partitioning the space of noise ϵ. The noise ϵ comes from a fixed known distribution (such as a Gaussian), and it does not require domain knowledge to partition the noise space. Although for theoretical reasons, the partitioning must be as fine as possible, we show empirically that solution quality was fairly robust to different levels of noise space discretization.

We divide the noise space into K partitions $\mathcal{P} = \{1,\ldots,K\}$. Partition index $k \in \mathcal{P}$ denotes the range $[\epsilon^{k-1},\epsilon^k]$. For simplicity, we assume the width of each partition is the same $\Delta = \epsilon^k - \epsilon^{k-1}$. We denote $\epsilon_\star^k$ as the midpoint of the range $[\epsilon^{k-1},\epsilon^k]$.

Linear approximation of the reward: Let us assume that ϵ^m, the input noise of agent m, falls in partition k (or $\epsilon^m \in [\epsilon^{k-1},\epsilon^k]$). For a given joint noise vector $\boldsymbol{\epsilon}$, let $\boldsymbol{k} \in \mathcal{P}^M$ denote the joint partition index vector for all the agents. We use $\boldsymbol{k}(m)$ to indicate the specific partition index for agent m under $\boldsymbol{k}$. We use $\boldsymbol{\epsilon}_\star^{\boldsymbol{k}} = \langle\epsilon_\star^{\boldsymbol{k}(m)},\ m = 1:M\rangle$ to denote the midpoint of partitions indexed by each agent's noise value.

Using the above notations, we can express $\epsilon^m = \epsilon_\star^{\boldsymbol{k}(m)} + \delta_m^{\boldsymbol{k}(m)}$ where $\delta_m^{\boldsymbol{k}(m)}$ is the small adjustment made to the midpoint $\epsilon_\star^{\boldsymbol{k}(m)}$ to match ϵ^m. Using first order Taylor approximation of the system reward at $\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}$, we have:

$$\begin{aligned} r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}) &\approx r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}) + (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\star^{\boldsymbol{k}})\cdot\nabla_{\boldsymbol{\epsilon}} r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}) \\ &= r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}) + \boldsymbol{\delta}\cdot\nabla_{\boldsymbol{\epsilon}} r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}) \end{aligned} \tag{16}$$

If the width Δ of noise partitions is small, then $\boldsymbol{\delta}$ (or the vector of adjustments $\delta_m^{\boldsymbol{k}(m)}$) would also be small. Under this setting, we can ignore the second term, and approximate the reward as:

$$r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}) \approx r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}}) \tag{17}$$

The state-noise vector $(\boldsymbol{s},\boldsymbol{\epsilon})$ does not result in meaningful count tables as $\boldsymbol{\epsilon}$ is continuous. However, given the relationship in (17), we can design a parameterized reward approximator based on the symmetries (e.g., different types of count information) present in $(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}})$. In $r_\theta(\boldsymbol{s},\boldsymbol{\epsilon}_\star^{\boldsymbol{k}})$, we use the midpoint value of noise partitions instead of the actual noise. Therefore, many agents may have noise values which fall in the same partition, which gives rise to the count information as noted next. We define count tables $(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}})$ where the state count table $\mathbf{n}_t^S = \langle\mathrm{n}_t(s)\rangle_{\forall s}$ counts the number of agents in same state at time t given the joint state $\boldsymbol{s}_t$ (same as for the discrete action case). Let $\mathbf{n}_t^{S\mathcal{P}} = \langle\mathrm{n}_t(s,k), \forall s \in S, \forall k \in K\rangle$ be the state-noise count table resulting from the joint state-noise vector $(\boldsymbol{s}_t,\boldsymbol{\epsilon}_t)$:

[1]https://jamesarambam.github.io/files/icaps21-sup.pdf

$$\mathrm{n}_t(s,k) = \sum_{m=1}^{M} \mathbb{I}\left[s_t^m = s, \mathrm{idx}(\epsilon_t^m) = k; \boldsymbol{s}_t, \boldsymbol{\epsilon}_t\right] \quad (18)$$

where $\mathrm{idx}(\epsilon_t^m)$ maps the noise for agent m, ϵ_t^m, to its corresponding noise partition index.

Structure of reward approximator: Based on count tables $(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}})$, we use a parameterized function $r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)$ which approximates the system reward. Intuitively, the reward signal r_ξ given the joint state-noise $(\boldsymbol{s}_t, \boldsymbol{\epsilon}_t)$ is approximated using $r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)$, where count tables result from $(\boldsymbol{s}_t, \boldsymbol{\epsilon}_t)$. The parameters $\mathbf{w}$ of the reward approximator are trained by minimizing the loss: $\mathcal{L}(\mathbf{w}) = \sum_{\xi\in\mathcal{B}}\left(r_\xi - r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)\right)^2$, as in the discrete action case.

4.2 Computing Difference Rewards

Recall that, action $a_t^m = f_\theta(\epsilon_t^m; s_t^m)$ is a deterministic function of noise ϵ_t^m, θ and state s_t^m. We can rewrite the definition of DR (9) for continuous action settings as:

$$D^m\left(s_t^m, \epsilon_t^m\right) = r_\theta(\boldsymbol{s}_t, \boldsymbol{\epsilon}_t) - r_\theta(\boldsymbol{s}_t^{-m} \cup d_s, \boldsymbol{\epsilon}_t^{-m} \cup d_\epsilon)$$

where d_ϵ and d_s are default noise and default state respectively. In our setting, analytical form of the reward model is not available. Therefore, we use the reward model $r_{\mathbf{w}}$ to rewrite the DR expression $D^m(s_t^m, \epsilon_t^m)$ as:

$$r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right) - r_{\mathbf{w}}\left(\mathbf{n}_t^{S - s_t^m + d_s}, \mathbf{n}_t^{S\mathcal{P} - (s_t^m, k) + (d_s, d_{k^\star})}\right) \quad (19)$$

where k and $d_{k^\star}$ are the partition indices of noise ϵ_t^m and default noise d_ϵ respectively. As agents with the same state and noise partition will have same DR value, we can compute it for each state and noise partition pair ($i \in S, k \in \mathcal{P}$):

$$\begin{aligned} D_t(i,k) \approx\; & r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right) - r_{\mathbf{w}}\big((\mathbf{n}_t^S - \mathcal{I}^i + \mathcal{I}^{d_s}), \\ & (\mathbf{n}_t^{S\mathcal{P}} - \mathcal{I}^{ik} + \mathcal{I}^{d_s d_{k^\star}})\big) \end{aligned} \quad (20)$$

$\mathcal{I}^i$, $\mathcal{I}^{d_s}$, $\mathcal{I}^{ik}$ and $\mathcal{I}^{d_s d_{k^\star}}$ have similar definition to the discrete action case as defined in paragraph above (11).

As our setting is for large agent population, as the agent population M becomes large, we can show similar to the proof of (14) (shown in supplement):

$$\begin{aligned} D_t(i,k) \approx & \frac{1}{M}\bigg(\frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(i)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(d_s)} + \\ & \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(i,k)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(d_s, d_{k^\star})}\bigg) \end{aligned} \quad (21)$$

4.3 SAC with DRs for Multiagent Setting

We adapt the SAC approach, which trains a policy for optimizing continuous actions using the re-parameterization trick (15) and entropy bonus for better exploration. The SAC uses empirical rewards to train action-value function approximator. This is where a key difference in our adaptation lies—we use the DR value D_t computed above to learn the action-value function, which is then used for policy training. Training with DR signals gives better solution quality as it is a much cleaner signal than the empirical reward, which has high noise due to actions of multiple agents. This also shows the strength of shaped rewards we have computed as they can be easily integrated in a variety of RL algorithms.

Let $Q_\eta\left(o^m(\boldsymbol{s}_t), a_t^m\right)$ and $V_\psi\left(o^m(\boldsymbol{s}_t)\right)$ be parameterized action-value and state-value functions respectively, where o^m denote agent m's observation (observation encodes the count-based information agent receives), $\boldsymbol{s}_t$ is the joint state. Since, the policy and value function approximators are shared among agents (we can also have one policy, value function per agent type), we aggregate the experiences collected by each agent, and update different function approximators as shown below.

State-value function update: The parameters ψ of state-value function are updated by minimizing:

$$\begin{aligned} \mathcal{L}_V(\psi) = \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D}}\Big[\frac{1}{2}\big(V_\psi\big(o^m(\boldsymbol{s}_t)\big) - \\ \mathbb{E}_{a_t^m\sim\pi_\theta}\left[Q_\eta\left(o^m(\boldsymbol{s}_t), a_t^m\right) - \log\pi_\theta(a_t^m|o^m(\boldsymbol{s}_t))\right]\big)^2\Big] \end{aligned}$$

where $\mathcal{D}$ is replay buffer where samples collected during simulation rollouts are stored; π_θ is the shared policy.

Action-value function update: The parameters η of action-value function is updated by minimizing:

$$\begin{aligned} \mathcal{L}_Q(\eta) = \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{s}_t, a_t^m\sim\mathcal{D}}\Big[\frac{1}{2}\big(Q_\eta\big(o^m(\boldsymbol{s}_t), a_t^m\big) - \\ \hat{Q}\big(o^m(\boldsymbol{s}_t), a_t^m\big)\big)^2\Big] \end{aligned} \quad (22)$$

Updating action-value is where our model-based approach differs from the standard SAC. The target for Q function prediction, $\hat{Q}$, is computed as:

$$\hat{Q}(o^m(\boldsymbol{s}_t), a_t^m) = D_t(o^m(\boldsymbol{s}_t), k^m) + \gamma\mathbb{E}_{s_{t+1}}\left[V_{\bar{\psi}}(o^m(\boldsymbol{s}_{t+1}))\right]$$

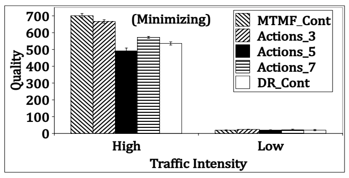
Crucially, here we use the difference reward value instead of the empirical reward. This signal provides better credit assignment than learning directly from empirical rewards. We use k^m to denote the partition index of the noise parameter ϵ_t^m corresponding to the action a_t^m.

Policy function update: The policy gradient update is the same as in the original SAC except that we also aggregate gradients over all the agents, similar to action-value and value function updates noted earlier.

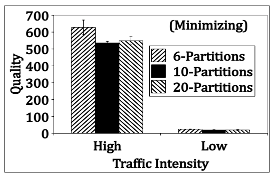
5 Experiments

We evaluate our proposed approach on both synthetic and real world instances of air-traffic control problem, and in a continuous state cooperative navigation problem.

Air traffic control (ATC): We follow similar settings used in (Brittain and Wei 2019) where only aircraft speed is controlled, not the altitude. We use following baselines. Two domain specific approaches—AT-BASELINE (Brittain and



(a) Comparison for discrete and continuous actions DR



(b) Comparison with different noise partitions

Figure 2: Discrete and continuous DR comparisons

Wei 2019) and AT-DR (Proper and Tumer 2013). LOCAL-DR (Colby et al. 2016) is another DR based baseline. We also compare against MCAC (Nguyen, Kumar, and Lau 2018) and MTMF (multitype mean field RL) (Subramanian et al. 2020), which are designed specifically for homogenous agent population. AT-DR is a DR based approach developed specifically for ATC. In our adaptation of AT-DR we used our reward approximation model because we did not have access to their customized simulator and for DR computation we use the expression in (11). Detailed experimental setting are in supplement. We omit results of training with global rewards as the learning curve was flat, implying the signal was too noisy.

Synthetic instance experiments setup: We follow the same experimental setup and cost function used in (Brittain and Wei 2019). We use BlueSky Air Traffic Control Simulator (Hoekstra and Ellerbroek 2016) to enable realistic simulation of aircraft movements.

Robustness to noise partitions for continuous action: We first checked sensitivity of our continuous action DR (DR-Cont) on different noise partitions $\mathcal{P}$. Fig 2b shows the results. We evaluate on a high and low traffic intensity settings (rate=0.1, 0.5 respectively) with 30 sectors map and 50 agents. Traffic rate is explained later. We observe that our approach is not very sensitive to number of partitions; for both 10 and 20 partitions, it achieved similar quality; 6 partitions slightly worse. For low traffic (rate=0.5), the trend was similar. We chose 10 partition setting for other experiments.

Comparison between discrete and continuous DR**:** Figure (2a) shows comparisons of our discrete action DR approach DIFF-RW with different action discretizations, our continuous action based DR (DR-Cont), and a continuous version of MTMF . Our goal is to show that different levels of discretizations can result in significant quality differences, and our continuous action approach avoids such engineering issues. We use DIFF-RW with 3, 5 and 7 actions. We use a large map with 30 sectors and 50 agents. We evaluate it for high and low traffic intensity settings. In high traffic setting, the quality varies with different levels of discretization, this show sensitivity of discrete DR with number of actions. The DR-Cont is able to achieve close to best quality (Actions-5), and performs better than MTMF also. For next discrete action experiments, we use 5 discretizations levels.

Accuracy of learned reward model: We also test the accuracy of our learned reward model by measuring the training loss. We use a map with 30 sectors, arrival rate = 0.1 and 50 agents. In figure (3d) we show the loss curve for reward model approximation. We include both of our approaches continuous (DR-Cont) and discrete action (DIFF-RW) versions, and previous reward approximation approach LOCAL-DR . We observe both of our approaches are able to minimize the loss effectively; LOCAL-DR is inaccurate as local information is not sufficient to predict complex rewards. For AT-DR , we used the same reward approximation model as ours, thus it has same value as our approach.

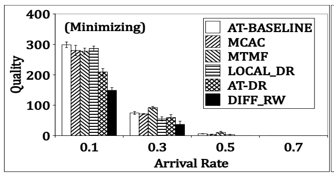
Figure (3a) shows result for experiments with 50 aircraft agents, 20 sector map, and with varying arrival rate. We use the same arrival rate setting in (Brittain and Wei 2019). At each time step and for each trajectory, the next arrival time of an aircraft is uniformly sampled from a set of intervals $arr_{intv} = \{20, 25, 30\}$. For example, a random sample of 20 for a trajectory means the next aircraft will arrive in $t+20$ time step. A value on the x-axis of figure (3a) denotes the fraction of each arrival intervals in arr_{intv}. For example, for arrival rate = 0.3, the arrival interval set become $arr_{intv} = \{0.3 \cdot 20, 0.3 \cdot 25, 0.3 \cdot 30\}$. We also round it to the nearest integer values as $arr_{intv} = \{6, 7, 9\}$. For arrival rate = 0.1, we have $arr_{intv} = \{2, 3\}$. The traffic becomes more intense with lower arrival rate, and would lead to higher congestion in an uncoordinated setting.

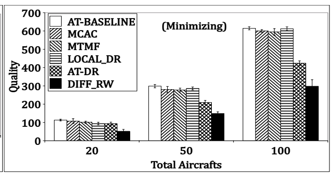
In figure (3a), we observe the expected trend of total quality improving with increasing arrival rate. We also observe the performance gap of DIFF-RW with other approaches decrease with increasing arrival rate. At arrival rate = 0.1 setting, due to high frequency of aircraft arrivals, most of the baseline approaches suffer from high congestion. This setting require tighter coordination among aircrafts, which is better achieved by DIFF-RW. Other DR baselines, LOCAL-DR suffers because in difficult instances DR value computation is noisy due to noisy reward model approximation. AT-DR is able to perform well than other baselines but still suboptimal to our approach. We also show the result for arrival rate = 0.7 setting to verify all approaches behave similarly at slightly easier problem instances.

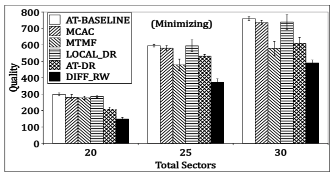
Figure (3b) shows result for setting with 20 sector map, fixed arrival rate = 0.1, and with varying aircraft population. In this setting, we observe the empirical evidence of DIFF-RW performing better with higher agent population. At lower population setting, almost all approach perform equally well. But for large population setting, other baselines suffer due to lack of efficient credit assignment.

We also tested with increasing number of sectors. Figure (3c) shows results with fixed arrival rate = 0.1 and aircraft population as 50. We observe that AT-BASELINE suffers most among other approaches. This is because in difficult instances, parameter sharing based method lacks coordination among agent without explicit credit assignment, even though it is scalable. Similar to previous results, DIFF-RW performed best for different map sizes.

We also performed experiments to compare the solution quality of DIFF-RW with two different loss functions described in section 3.1. The system reward in air traffic domain is decomposed among agents (Brittain and Wei 2019). Old loss and new loss in figure (4c) denote DIFF-RW approach with the loss functions in (7) and (8) respectively (20


(a) Varying arrival rate


(b) Varying agent population


(c) Varying number of sectors

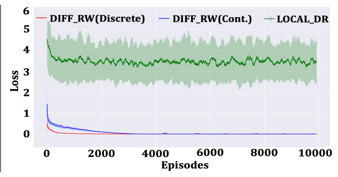
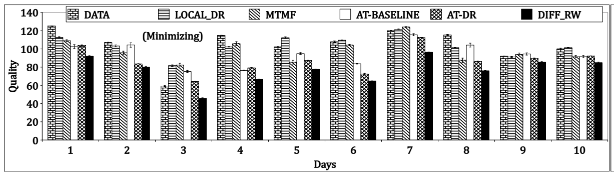
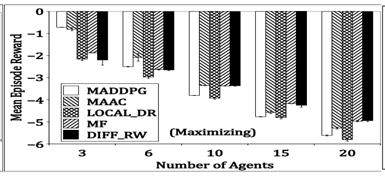

(d) Reward model approx. accuracy

Figure 3: Results on synthetic instance experiments


(a) Results of real world instances


(b) Cooperative Navigation

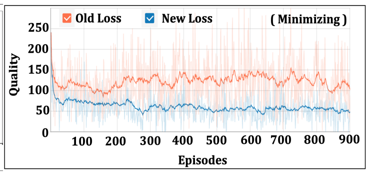

(c) Quality comparison of different loss

Figure 4: Different comparison results

Agents, 15 sectors, 0.1 arrival rate). The y-axis denotes the performance metric. We observe that DIFF-RW with new loss is able to achieve better quality than with old loss. We observed similar results on several other settings. Therefore, for all previous experiments we used our new loss function in (8).

Real data experiments: We evaluate our approach on air space surrounding one of the busiest airport Heathrow, London. The data for 30 days is obtained from a commercial company over the European airspace. We use 20 days data for training and 10 days for testing. More details on experimental setup is provided in supplement. We train and test our approach only for the peak hour period formed between 6th-10th hour. The plot for traffic intensity of aircrafts is provided in supplement.

Training: For each training day, we extract the initial information of simulation such as speed and lat-lon of aircrafts at 1st second of 6th Hour from data. We also extract the future arrival information of aircrafts such as arrival time points between the 6th-10th hour period, speed when aircrafts enters the planning region and their lat-lon information. During training, at the start of each RL episode, we randomly select a day from the pool of 20 training days. Then we use the initial and future arrival information of that day to simulate and learn the policy.

Testing: We evaluate our learned policy on 10 separate testing days. For each day, we extract the initial and future arrival information for simulation. Then the trained policy is evaluated on each of the remaining 10 days separately. Figure (4a) shows result of our approach compared against baselines (MCAC was slightly worse than MTMF , to avoid clutter, its bars are omitted). We observe in most of the days all approaches perform equal or better than DATA (which is the replay of the historical dataset). DIFF-RW is able to achieve much better solution quality than other baselines.

Cooperative navigation domain: We also evaluate our approach on cooperative multiagent navigation (Lowe et al. 2017). The state space in this environment is continuous, therefore we use tile coding based technique for discretization. For this domain we use following baselines—MADDPG (Lowe et al. 2017), MAAC (Iqbal and Sha 2019), (LOCAL-DR) (Colby et al. 2016) and mean field multiagent RL (MF) (Yang et al. 2018) (MTMF was not applicable as all agents were of same type). The MCAC did not perform well as unlike the air traffic domain where the underlying network of sectors helped design a good critic structure, there is no such structure in this domain.

Figure (4b) shows results with varying agent population; y-axis denotes mean episode reward. For small agent population $n = 3$, MADDPG and MAAC perform better than MF , LOCAL-DR and DIFF-RW . This is because our approximation of DR for small number of agents may not be accurate. However, with increasing agent population, solution quality of MADDPG , MAAC and LOCAL-DR drops, and DIFF-RW and MF improves. This trend is an empirical evidence of the accuracy of our DR method with increasing agent population. For 20 agent setting, even though DIFF-RW and MF have similar solution quality, our approach is able to converge faster than MF . At 1000 episodes, MF has quality of around -6.3 where as DIFF-RW converged to around -5.2 value (higher is better). MF eventually converge to -5.2, but at around 3000 episodes. Learning curve plot provided in supplement.

6 Conclusion

We presented a new approach to estimate difference rewards for effective multiagent credit assignment in large multiagent systems. We exploited the property that in many practical applications, agents are homogeneous (or belong to a few types). Unlike previous techniques for difference rewards, our method does not require domain expertise or extra simulations, and is highly scalable with the number of agents. On a range of synthetic and real world instances, we showed significantly improved performance of our approach against several other competing MARL algorithms.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR), Fujitsu Limited and the National Research Foundation Singapore as part of the A*STAR-Fujitsu- SMU Urban Computing and Engineering Centre of Excellence.

References

Agogino, A. K.; and Tumer, K. 2004. Unifying temporal and structural credit assignment problems. In *International Conference on Autonomous Agents and Multiagent Systems*, 980–987.

Agogino, A. K.; and Tumer, K. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Journal of Autonomous Agents and Multi-Agent Systems* 17(2): 320–338.

Bagnell, D.; and Ng, A. Y. 2006. On local rewards and scaling distributed reinforcement learning. In *Advances in Neural Information Processing Systems*, 91–98.

Brittain, M.; and Wei, P. 2019. Autonomous Separation Assurance in An High-Density En Route Sector: A Deep Multi-Agent Reinforcement Learning Approach. In *2019 IEEE Intelligent Transportation Systems Conference, ITSC*, 3256–3262. IEEE.

Chow, C. .; and Tsitsiklis, J. N. 1991. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control* 36(8): 898–914.

Colby, M. K.; Duchow-Pressley, T.; Chung, J. J.; and Tumer, K. 2016. Local Approximation of Difference Evaluation Functions. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 521–529. ACM.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2974–2982. AAAI Press.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the ICML*. PMLR.

Hoekstra, J. M.; and Ellerbroek, J. 2016. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of International Conference on Research in Air Transportation*.

Hüttenrauch, M.; Šošić, A.; and Neumann, G. 2018. Deep Reinforcement Learning for Swarm Systems. *JMLR* .

Iqbal, S.; and Sha, F. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *Proceedings of ICML*. PMLR.

Lowe, R.; WU, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, 6379–6390.

Mnih, V.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2017a. Collective Multiagent Sequential Decision Making Under Uncertainty. In *AAAI Conference on Artificial Intelligence*, 3036–3043.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2017b. Policy Gradient With Value Function Approximation For Collective Multiagent Planning. In *Advances in Neural Information Processing Systems*.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2018. Credit assignment for collective multiagent RL with global rewards. In *Advances in Neural Information Processing Systems*, 8102–8113.

Oliehoek, F. A.; Amato, C.; et al. 2016. *A concise introduction to decentralized POMDPs*, volume 1. Springer.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32: 289–353.

Peshkin, L.; Kim, K.-E.; Meuleau, N.; and Kaelbling, L. P. 2000. Learning to Cooperate via Policy Search. In *Conference in Uncertainty in Artificial Intelligence*, 489–496.

Proper, S.; and Tumer, K. 2012. Modeling difference rewards for multiagent learning. In *International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS.

Proper, S.; and Tumer, K. 2013. Multiagent Learning with a Noisy Global Reward Signal. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press.

Rashid, T.; et al. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of ICML*. PMLR.

Robbel, P.; et al. 2016. Exploiting Anonymity in Approximate Linear Programming: Scaling to Large Multiagent MDPs. In *Proceedings of AAAI Conference on Artificial Intelligence*. AAAI Press.

Silver, D.; et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550(7676): 354–359.

Singh, A. J.; Kumar, A.; and Lau, H. C. 2020. Hierarchical Multiagent Reinforcement Learning for Maritime Traffic Management. In *Proceedings of AAMAS*. IFAAMAS.

Singh, A. J.; Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2019. Multiagent Decision Making For Maritime Traffic Management. In *AAAI Conference on Artificial Intelligence*. AAAI Press.

Sonu, E.; Chen, Y.; and Doshi, P. 2015. Individual Planning in Agent Populations: Exploiting Anonymity and Frame-Action Hypergraphs. In *Proceedings of ICAPS*. AAAI Press.

Subramanian, J.; and Mahajan, A. 2019. Reinforcement learning in stationary mean-field games. In *International Conference on Autonomous Agents and Multiagent Systems*, volume 1, 251–259.

Subramanian, S. G.; Poupart, P.; Taylor, M. E.; and Hegde, N. 2020. Multi Type Mean Field Reinforcement Learning. In *Proceedings of the 19th International Conference on AAMAS*. IFAAMAS.

Sun, C.; Shen, M.; and How, J. P. 2020. Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. *arXiv preprint arXiv:2003.01040* .

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tumer, K.; and Agogino, A. K. 2007. Distributed agent-based air traffic flow management. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS.

Varakantham, P.; Adulyasak, Y.; and Jaillet, P. 2014. Decentralized stochastic planning with anonymity in interactions. In *AAAI Conference on Artificial Intelligence*. AAAI Press.

Wiering, M. 2000. Multi-Agent Reinforcement Learning for Traffic Light Control. In *Proceedings of ICML*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Yang, Y.; et al. 2018. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of ICML*.