Supplemental Materials to Submission: # 201
Learning and Exploiting Shaped Reward Models for Large Scale Multiagent RL

1 Derivation of Difference Reward Approximation for Continuous Actions

From the main paper, we have :

$$D_t(i,k) = r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right) - r_{\mathbf{w}}\big((\mathbf{n}_t^S - \mathcal{I}^i + \mathcal{I}^{d_s}), (\mathbf{n}_t^{S\mathcal{P}} - \mathcal{I}^{ik} + \mathcal{I}^{d_s d_k})\big) \tag{1}$$

For stable learning, we normalize the count variables with total population M.

$$D_t(i,k) = r_{\mathbf{w}}\left(\mathbf{n}_t^S/M, \mathbf{n}_t^{S\mathcal{P}}/M\right) - r_{\mathbf{w}}\big((\mathbf{n}_t^S - \mathcal{I}^i + \mathcal{I}^{d_s})/M, (\mathbf{n}_t^{S\mathcal{P}} - \mathcal{I}^{ik} + \mathcal{I}^{d_s d_k})/M\big) \tag{2}$$

where, $\mathcal{I}^{ik}$ and $\mathcal{I}^{d_s d_k}$ are like identity matrix with same dimension as $\mathbf{n}_t^{S\mathcal{P}}$ with all zero entries except value 1 at the index corresponding to (i,k) and default state-noise (d_s, d_k) index respectively. Similarly, $\mathcal{I}^i$ and $\mathcal{I}^{d_s}$ are vectors with all zero entries except value 1 at the index corresponding to i and default state d_s respectively.

As our setting is for large agent population, DR expression (2) when $M \to \infty$ becomes,

$$D_t(i,k) \approx \lim_{M\to\infty} \left[r_{\mathbf{w}}\left(\mathbf{n}_t^S/M, \mathbf{n}_t^{S\mathcal{P}}/M\right) - r_{\mathrm{w}}\big((\mathbf{n}_t^S - \mathcal{I}^i + \mathcal{I}^{d_s})/M, (\mathbf{n}_t^{S\mathcal{P}} - \mathcal{I}^{ik} + \mathcal{I}^{d_s d_k})/M\big)\right] \tag{3}$$

$$= \lim_{\Delta = 1/M \to 0} \left[r_{\mathbf{w}}\big(\tilde{\mathbf{n}}_t^S, \tilde{\mathbf{n}}_t^{S\mathcal{P}}\big) - r_{\mathbf{w}}\big((\tilde{\mathbf{n}}_t^S - \Delta\cdot(\mathcal{I}^i - \mathcal{I}^{d_s})), (\tilde{\mathbf{n}}_t^{S\mathcal{P}} - \Delta\cdot(\mathcal{I}^{ik} - \mathcal{I}^{d_s d_k}))\big)\right] \tag{4}$$

$$= -1 \cdot \lim_{\Delta\to 0} \left[r_{\mathbf{w}}\big((\tilde{\mathbf{n}}_t^S - \Delta\cdot(\mathcal{I}^i - \mathcal{I}^{d_s})), (\tilde{\mathbf{n}}_t^{S\mathcal{P}} - \Delta\cdot(\mathcal{I}^{ik} - \mathcal{I}^{d_s d_k}))\big) - r_{\mathbf{w}}\big(\tilde{\mathbf{n}}_t^S, \tilde{\mathbf{n}}_t^{S\mathcal{P}}\big)\right] \tag{5}$$

By the definition of total differential, above expression becomes:

$$D_t(i,k) \approx -1\cdot(-\Delta)\left(\frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(i)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(d_s)} + \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(i,k)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(d_s,d_k)}\right) \tag{6}$$

$$D_t(i,k) \approx \frac{1}{M}\left(\frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(i)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^S(d_s)} + \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(i,k)} - \frac{\partial r_{\mathbf{w}}\left(\mathbf{n}_t^S, \mathbf{n}_t^{S\mathcal{P}}\right)}{\partial \mathbf{n}_t^{S\mathcal{P}}(d_s,d_k)}\right) \tag{7}$$

2 Policy Gradient for MARL for Discrete Actions

In this section, we review existing policy gradient methods for MARL, and show equivalent gradient expression in our setting where agents are homogeneous (or belong to small number of types), and use a shared policy parameterized by θ.

Policy gradient [15, 16, 10, 11] methods are popular approaches to solve RL tasks. In single agent setting, the policy parameter θ is optimized by performing gradient ascent on the parameter space to maximize the expected discounted return. The gradient expression can be written as [15]:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_{0:\infty}, a_{0:\infty}} \Big[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot R_t \Big] \tag{8}$$

The natural extension of policy gradient method in multiagent setting is to take the gradient of the objective w.r.t. each agent's policy parameter θ^m [8]:

$$\nabla_{\theta^m} J(\boldsymbol{\pi}_\theta) = \mathbb{E}_{\boldsymbol{s}_{0:\infty}, \boldsymbol{a}_{0:\infty}} \Big[\sum_{t=0}^{\infty} \nabla_{\theta^m} \log \pi^m_{\theta^m}(a^m_t \mid z^m_t) \cdot R_t \Big] \tag{9}$$

For simplicity, we have assumed that the policy is conditioned on the agent's last observation z^m_t; finite-state controllers which summarize an agent's observation history can also be used [8].

In some variants of the above policy gradient method, such as [2], each agent approximates the return R_t in (9) using a parameterized action-value function Q_w which can be trained using TD(λ) [14]. Such approaches are also known as multiagent *actor-critic* methods [5] as an agent's policy π^m is the actor, and Q_w is the critic to approximate the action-value function.

In our setting, since agents are homogeneous (or belong to a small number of types), the policy π^m is also shared among agents. This is a common technique used in MARL algorithms to speed up convergence [3, 17, 7]. Since the state also includes the type information, the shared policy can still generate different behaviors based on the type. Furthermore, another common assumption in such homogeneous agent setting is that an agent's observation z^m_t depends only on the agent's local state s^m_t and the aggregate state count table or $z^m_t = o(s^m_t, \mathbf{n}^S_t)$ [7, 17, 12]. As an example, in the air traffic control domain, the aircraft agent in a sector can observe its current sector and the count of other aircrafts in neighboring sectors. The observation does not depend on the identity of aircrafts, only on their aggregate statistic. By exploiting this aggregation-based symmetry and a shared policy, we can also aggregate the gradient in (9). Let $\langle \boldsymbol{s}_t, \boldsymbol{a}_t \rangle = \langle s^m_t, a^m_t \rangle_{m=1:M}$ be the joint state-action pair for agents at time t, and $\mathbf{n}^S_t$, $\mathbf{n}^{SA}_t$ be the corresponding count table vectors. The aggregate gradient over all the agents is given as:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\boldsymbol{s}_{0:\infty}, \boldsymbol{a}_{0:\infty}} \Big[\sum_{t=0}^{\infty} \sum_{s \in S} \sum_{a \in A} \sum_{m=1}^{M} \mathbb{I}\left[s^m_t = s, a^m_t = a\right] \times \nabla_\theta \log \pi_\theta\big(a \mid z_t = o(s, \mathbf{n}^S_t)\big) \cdot R_t \Big] \tag{10}$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\boldsymbol{s}_{0:\infty}, \boldsymbol{a}_{0:\infty}} \Big[\sum_{t=0}^{\infty} \sum_{s \in S} \sum_{a \in A} \mathrm{n}_t(s, a) \cdot \nabla_\theta \log \pi_\theta(a \mid z_t) \cdot R_t \Big] \tag{11}$$

Although the above gradient can be estimated using sampling, credit assignment remain a challenge as returns R_t are estimated from the global rewards. As a result, it will take very high number of samples to determine which agents actions are important for high returns. To address this issue, we next show how to compute difference rewards, and returns R_t can then be compute from difference rewards, rather than global system rewards to address the credit assignment problem.

Once difference rewards are computed as shown in the main paper, it is straightforward to integrate them in a policy gradient equation (11). We first compute the DR based empirical return as:

$$R_t^{dr} = \sum_{i=0}^{\infty} \gamma^i \left(\sum_{s \in S} \sum_{a \in A} \mathrm{n}_{t+i}(s,a) \cdot D_{t+i}(s,a) \right) \tag{12}$$

We then replace the empirical return R_t in the policy gradient expression (11) with DR based return R_t^{dr}, which has better credit assignment signal than R_t. Notice that difference rewards are highly versatile. Instead of a policy gradient setting, they can also be integrated in a multiagent Q-learning context [9] or multiagent actor-critic methods [6]. We defer such investigations as part of the future work.

3 Air Traffic Control Domain

In this section, we present the multiagent model for our motivating application of air-traffic control. We assume a total of M aircraft agents moving in a planning area which are divided into multiple sectors $z \in Z$. Each aircraft follows a predefined trajectory while moving in a sector.

State: Let $s_t^m = (z, \boldsymbol{d})$ denote the state of an aircraft m at time t, where $z \in Z$ is the current sector of the aircraft, $\boldsymbol{d} = \langle d_i \rangle_{i=1:N}$ is a distance vector of aircraft m from it's N nearest neighboring aircrafts, with each element d_i as the distance to the aircraft $i \in [1, N]$ in nautical miles.

Reward: The reward function of air traffic control problem is same as in [1]. The algorithms only observe the numerical reward obtained from the simulator.

Action: Let a_t^m denote the aircraft m's action at time t. We define the action as speed change, let $A = [-50\%, -20\%, 0, +20\%, +50\%]$ be the action space e.g $a_t^m = +20\%$ means 20% increase from the current speed and $a_t^m = 0$ means no speed change i.e the aircraft continues with it's current speed. We also have minimum and maximum speed $[v_{min}, v_{max}]$ constraints on the speed.

Air Traffic Simulator: For our work, we use a python based free open source simulator, BlueSky Air Traffic Control Simulator [4] as the RL environment. At each time step, the simulator takes aircraft speed as the input and simulate the next state.

Aggregate Statistics: Let $\boldsymbol{s}_t, \boldsymbol{a}_t = \langle s_t^m, a_t^m, m = 1 : M \rangle$ be the joint state-action pair for all agent at time t, now we define aggregate statistics:

- $\mathrm{n}_t(z, \boldsymbol{d}) = \sum_{m=1}^{M} \mathbb{I}[s_t^m = (z_t^m = z, \boldsymbol{d}_t^m = \boldsymbol{d})], \forall z, \forall \boldsymbol{d}$, counts the number of aircraft which are in sector z and have similar distance vector $\boldsymbol{d}$ from neighboring aircrafts at time t.

- $\mathrm{n}_t(z, \boldsymbol{d}, v) = \sum_{m=1}^{M} \mathbb{I}[s_t^m = (z_t^m = z, \boldsymbol{d}_t^m = \boldsymbol{d}), a_t^m = v], \forall z, \forall \boldsymbol{d}$,

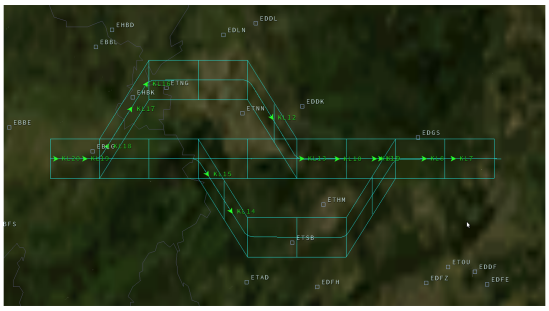
$\forall v \in A$, counts the number of aircraft which are in sector z, have similar distance vector $\boldsymbol{d}$ from neighboring aircrafts and taking action v at time t.

Let, $\mathbf{n}_t^s = \langle \mathrm{n}_t(z, \boldsymbol{d}) \rangle_{\forall z, \forall \boldsymbol{d}}$ and $\mathbf{n}_t^{sa} = \langle \mathrm{n}_t(z, \boldsymbol{d}, v) \rangle_{\forall z, \forall \boldsymbol{d}, \forall v}$ be the state and state-action count table respectively, and $\mathbf{n}_t = \langle \mathbf{n}_t^s, \mathbf{n}_t^{sa} \rangle$ be the count table vector at time t.

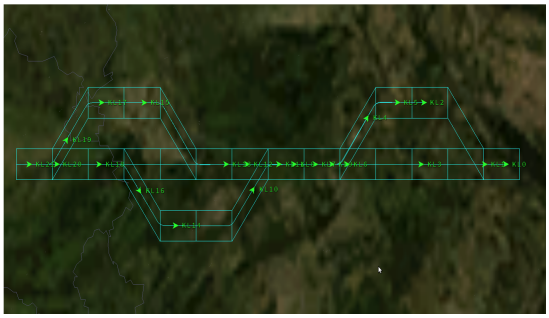
Traffic Control Policy: In order to control air-traffic, we have $\boldsymbol{\pi} = \{\pi_{\theta^z}(\bullet), \forall z \in Z\}$, $\pi_{\theta^z}(\bullet)$ is a parameterized θ^z policy function for each sector $z \in Z$. When an aircraft m enters a sector z, it samples the action $a_t^m \sim \pi_{\theta^z}(o^m)$, where o^m is the local observation by aircraft m.

State discretization: We discretize the distance d_i into six discrete buckets as follows:

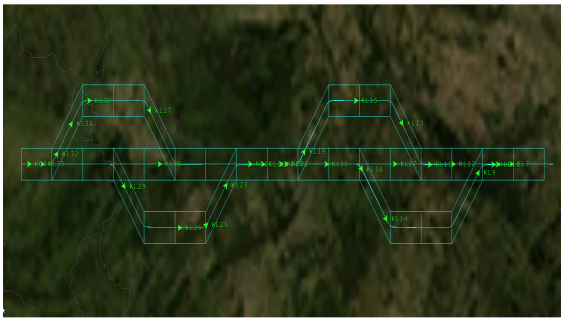
$$d_i = \begin{cases} 2 & , d_i < 3 \\ 3.5 & , 3 \leq d_i < 4 \\ 5 & , 4 \leq d_i < 6 \\ 7 & , 6 \leq d_i < 8 \\ 9 & , 8 \leq d_i < 10 \\ 10 & , 10 \leq d_i \end{cases}$$



(a) 20 Sectors



(b) 25 Sectors



(c) 30 Sectors

Figure 1: Maps for synthetic data experiments

3.1 Experimental settings:

For all experimental settings(including cooperative navigation experiments), we use 5 random seeds, and average values are reported.

Synthetic instances map: We define planning area(or map) for synthetic experiments as set of adjoining polygons. We call each polygon as a sector. The shape and structure of planning area is extended from the synthetic problem instance used in [1]. The map in fig(1a) with 20 sectors is used for the experiments with varying arrival rate(Fig. 3 in main paper) and varying agent population(Fig. 4 in main paper). The maps in fig(1b) and fig(1c) are used for experiments with varying number of sectors(Fig. 5) in the main paper.

3.2 Approaches:

DIFF-RW : DIFF-RW is our proposed approximate difference reward based approach. For the air traffic domain, we have a separate policy network for each sector. We use a fully-connected feed-forward neural network with 2 hidden layers for each policy network. Each layer has 100 hidden units. We have another single global neural network for approximating the reward function. Within the same neural network, we decompose it into small sub networks. Each subnetwork corresponds to each sectors. Each subnetwork has 2 hidden layers with 100 hidden units on each layer. For each hidden layer we use leky-relu as activation function. We use Adam optimizer with learning rate 1e-3. We use no speed change action or max speed change action as hyperparameter for default actions in air traffic domain. Similarly for default state we use mid-point index of the state count table.

AT-BASELINE : AT-BASELINE is one of the baseline approach we use in our experiments. It is proposed in [1]. We follow the same hyperparameter settings used in their work.

MTMF : MTMF is another baseline approach proposed in [13]. For this we have only one global critic network. The network architecture is same as reward network structure in DIFF-RW . We use Adam optimizer with learning rate 1e-3. For this baseline, we use Boltzman policy as described in in [13].

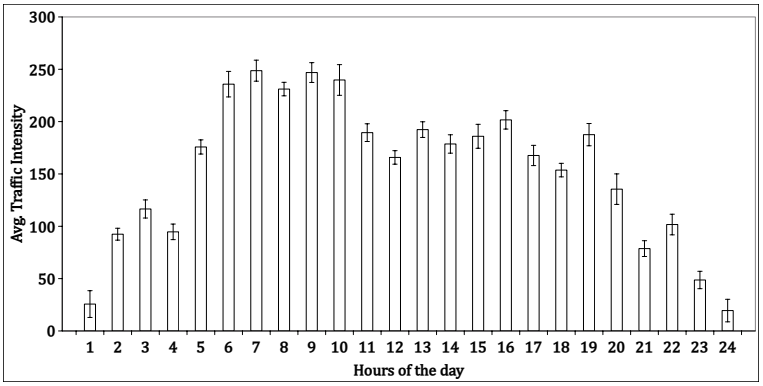
MTMF-Cont : MTMF-Cont has same network structure as MTMF .

MCAC : MCAC is another baseline approach proposed in [7]. For this baseline, we have similar neural network structure as DIFF-RW . The only difference is, instead of reward function approximation. We have global action-value function approximation.

LOCAL-DR : For this baseline, each agent have it's local reward function approximator. All agents have same neural network architecture. For each neural network function, we use a fully-connected feed-forward neural network with 2 hidden layers. Each layer has 100 hidden units. For each hidden layer we use leky-relu as activation function. We use Adam optimizer with learning rate 1e-3. Each agent also have local policy network. The policy network architecture is same as that of DIFF-RW .

AT-DR : This baseline has same neural network architecture of reward function and policy function as DIFF-RW .

DR-Cont : For DR-Cont we use soft-actor critic(SAC) implementation in spinningup code base. We use the default neural network hyper-parameters in codebase itself. The reward function approximator architecture is same as DIFF-RW . We use Adam optimizer with learning rate 1e-3.



(a)

Figure 2: Traffic Intensity (average over 30 days)

3.3 Real data experimental settings

Each data record contains aircraft information such as aviation callsigns, time stamp, lat-long position, speed, heading. We have records for every few seconds, resulting in around 10 millions records in the Heathrow airspace. First we select the planning region as airspace surrounding the airport within approximately 100km radius. Then we divide the whole region into multiple grid sectors with grid size (50km$\times$60km). Many aircrafts fly over the region at different altitudes. Some aircrafts fly at 30k feet range, which is considered as cruising altitude. At such altitude, there are not many conflicts. On the other hand, at very low altitudes, e.g. around 2k feet, aircrafts are very close to the airport, and controlled by the airport ATC(air traffic control). Therefore, in our evaluation scenarios, we consider only those aircrafts whose altitudes are between 5k feet to 20k feet range.

Figure (2a) shows traffic intensity of aircrafts in our planning area flying at altitudes between the 5k - 20k feet range. The x-axis shows hour of the day and y-axis represent the average traffic intensity in terms of aircraft counts. We observe that there is peak hour formed for around 5 hours period between 6th-10th hour. We train and test our approach during this peak hour period only.

4 Cooperative Navigation Domain

To discretize the state space in cooperative navigation domain, we use 2 tiles. Each tile is divided into (5x5=25) Grids. The two baselines used in this domain are MADDPG and MAAC. Implementation of both baselines are used from the code link provided in the respective paper.

MF: MF is another baseline we use. For this we have only one global critic network. We use a fully-connected feed-forward neural network with 2 hidden layers. Each layer has 100 hidden units. For each hidden layer we use leky-relu as activation function. We use Adam optimizer with learning rate 1e-3.

DIFF-RW : DIFF-RW is our approach. Since in this domain reward is non-decomposable. We use two networks one for each policy and reward function approximation. The neural network architecture for each networks is same. a fully-connected feed-forward neural network with 2 hidden layers.

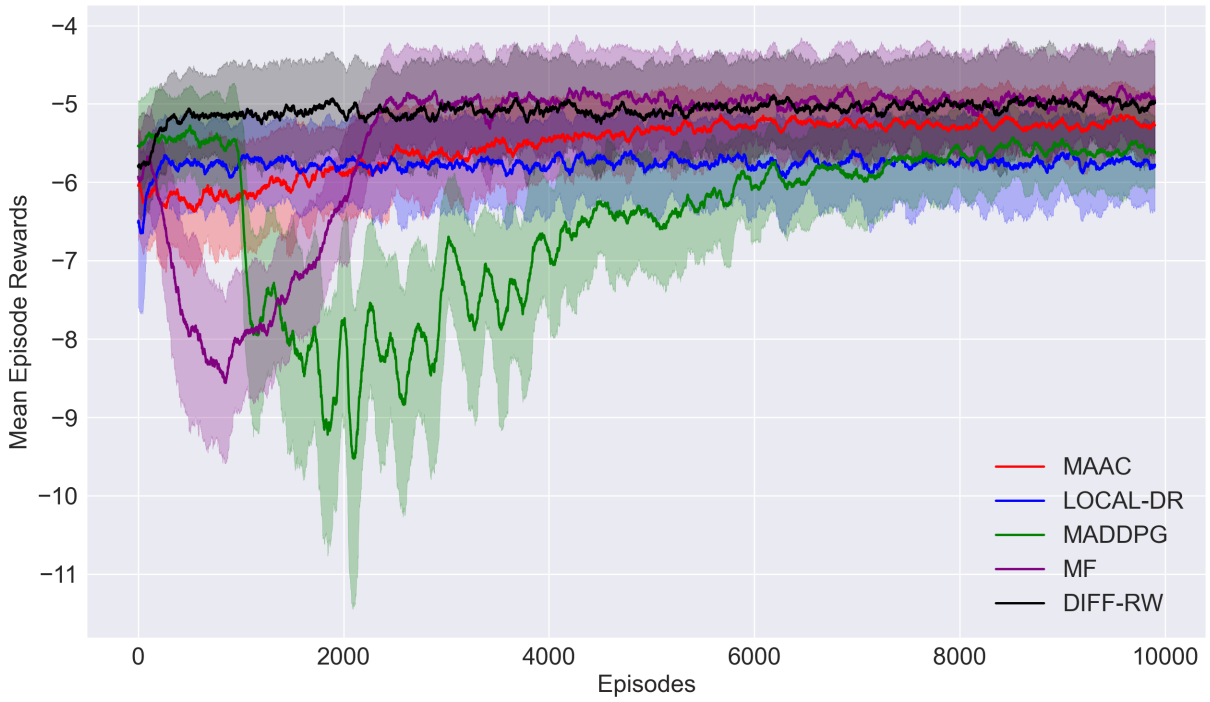


Figure 3: Learning Curve

Each layer has 100 hidden units. For each hidden layer we use leky-relu as activation function. We use Adam optimizer with learning rate 1e-3. In difference reward approximation, we use no action as default action. No action or not performing any operations is also part of action space. For default state we use the mid cell of each tile.

LOCAL-DR : Each agent has two networks, one for policy and another for reward function approximation. The network architecture is same as DIFF-RW . We use Adam optimizer with learning rate 1e-3.

References

[1] Marc Brittain and Peng Wei. Autonomous air traffic controller: A deep multi-agent reinforcement learning approach. In *ICML Workshop Reinforcement Learning for Real Life, arXiv preprint arXiv:1905.01303*. PMLR, 2019.

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

[3] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Lecture Notes in Computer Science*, pages 66–83, 2017.

[4] Jacco M Hoekstra and Joost Ellerbroek. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of the 7th International Conference on Research in Air Transportation*, volume 131, page 132. FAA/Eurocontrol USA/Europe, 2016.

[5] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*, pages 1008–1014. MIT Press, 2000.

[6] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390. Curran Associates, Inc., 2017.

[7] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Credit assignment for collective multiagent rl with global rewards. In *Advances in Neural Information Processing Systems*, pages 8102–8113, 2018.

[8] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to Cooperate via Policy Search. In *Conference in Uncertainty in Artificial Intelligence*, pages 489–496, 2000.

[9] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80, pages 4292–4301. PMLR, 2018.

[10] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. JMLR.org, 2015.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[12] Arambam James Singh, Akshat Kumar, and Hoong Chuin Lau. Hierarchical multiagent reinforcement learning for maritime traffic management. In *19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1278–1286. IFAAMAS, 2020.

[13] Sriram Ganapathi Subramanian, Pascal Poupart, Matthew E. Taylor, and Nidhi Hegde. Multi type mean field reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 411–419. IFAAMAS, 2020.

[14] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[15] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063. The MIT Press, 2000.

[16] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[17] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80, pages 5567–5576. PMLR, 2018.