Coach-Player Multi-Agent Reinforcement Learning for Dynamic Team Composition

Bo Liu
Qiang Liu
Peter Stone
Animesh Garg
Yuke Zhu
Animashree Anandkumar

Abstract
In real-world multi-agent systems, agents with different capabilities may join or leave without altering the team’s overarching goals. Coordinating teams with such dynamic composition is challenging: the optimal team strategy varies with the composition. We propose COPA, a coach-player framework to tackle this problem. We assume the coach has a global view of the environment and coordinates the players, who only have partial views, by distributing individual strategies. Specifically, we 1) adopt the attention mechanism for both the coach and the players; 2) propose a variational objective to regularize learning; and 3) design an adaptive communication method to let the coach decide when to communicate with the players. We validate our methods on a resource collection task, a rescue game, and the StarCraft micromanagement tasks. We demonstrate zero-shot generalization to new team compositions. Our method achieves comparable or better performance than the setting where all players have a full view of the environment. Moreover, we see that the performance remains high even when the coach communicates as little as 13% of the time using the adaptive communication strategy. Code is available at https://github.com/Cranial-XIX/marl-copa.git.

1. Introduction

Cooperative multi-agent reinforcement learning (MARL) is the problem of coordinating a team of agents to perform a shared task. It has broad applications in autonomous vehicle teams (Cao et al., 2012), sensor networks (Choi et al., 2009), finance (Lee et al., 2007), and social science (Leibo et al., 2017). Recent works in multi-agent reinforcement learning (MARL) powered by deep neural networks have shed light on solving challenging problems such as playing StarCraft (Rashid et al., 2018) and soccer (Kurach et al., 2020). Among these methods, centralized training with decentralized execution (CTDE) has gained extensive attention since learning in a centralized way enables better cooperation while executing independently makes the system efficient and scalable (Lowe et al., 2017; Foerster et al., 2018; Son et al., 2019; Mahajan et al., 2019; Wang et al., 2020a). However, most deep CTDE approaches for cooperative MARL are limited to a fixed number of homogeneous agents.

Real-world multi-agent tasks, on the other hand, often involve dynamic teams. For example, in a soccer game, a team receiving a red card has one fewer player. In this case, the team may switch to a more defensive strategy. As another example, consider an autonomous vehicle team for delivery. The control over the team depends on how many vehicles we have, how much load each vehicle permits, as well as the delivery destinations. In both examples, the optimal team strategy varies according to the team composition, i.e., the size of the team and each agent’s capability. In these settings, it is computationally prohibitive to re-train the agents for each new team composition. Thus, it is desirable that the model can generalize zero-shot to new team compositions that are unseen during training.

1Team composition is part of an environmental scenario (de Witt et al., 2019), which also includes other environment entities. The formal definition is in Section 2.1.
Recently, Iqbal et al. (2020) proposed to use the multi-head attention mechanism for modeling a variable number of agents under the CTDE framework. However, the CTDE constraint can be overly restrictive for complex tasks as each agent only has access to its own decisions and partial environmental observations at test time (see Section 3.1 for an example where this requirement forbids learning). The CTDE constraint can be relaxed by introducing communication. To our knowledge, learned communication has only been investigated in homogeneous teams (Foerster et al., 2016). On the other hand, prior works on ad hoc teamwork (Barrett et al., 2014; Grizou et al., 2016; Mirsky et al., 2020) assume a pre-defined communication protocol of which all agents are aware. Moreover, allowing all agents to communicate with each other is too expensive for many scenarios (e.g., battery-powered drones or vehicles). Inspired by real-world sports, we propose to introduce a centralized “coach” agent who periodically distributes strategic information based on the full view of the environment. We call our method COPA (COach-and-PlAyer). To our knowledge, this is the first work that investigates learned communication for ad hoc teams.

We introduce a coach with an omniscient view of the environment, while players only have partial views. We assume that the coach can distribute information to other agents only in limited amounts. We model this communication through a continuous vector, termed as the strategy vector, and it is specific to each agent. We design each agent’s decision module to incorporate the most recent strategy vector from the coach. Inspired by Rakelly et al. (2019) and Wang et al. (2020a), we design an additional variational objective to regularize the learning of the strategy, leading to faster learning and improved performance, and an adaptive communication strategy to minimize communication from the coach to the agents.

We introduce a coach with an omniscient view of the environment, while players only have partial views. We assume that the coach can distribute information to other agents only in limited amounts. We model this communication through a continuous vector, termed as the strategy vector, and it is specific to each agent. We design each agent’s decision module to incorporate the most recent strategy vector from the coach. Inspired by Rakelly et al. (2019) and Wang et al. (2020a), we design an additional variational objective to regularize the learning of the strategy, leading to faster learning and improved performance, and an adaptive communication strategy to minimize communication from the coach to the agents.

We demonstrate that COPA achieves zero-shot generalization to unseen team compositions. The performance stays strong with a communication frequency as low as 13%. Moreover, COPA achieves comparable or even better performance than methods where all players have full views of the environment.

2. Background

In this section, we formally define the learning problem considered in this paper. Then we briefly summarize the idea of value function factorization and a recent improvement that additionally incorporates the attention mechanism, which we leverage in COPA.

2.1. Problem Formulation

We model the cooperative multi-agent task under the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek et al., 2016). Specifically, we build on the framework of Dec-POMDPs with entities (de Witt et al., 2019), which uses an entity-based knowledge representation. Here, entities include both controllable agents and other environment landmarks. In addition, we extend the representation to allow agents to have individual characteristics, i.e., skill-level, physical condition, etc. Therefore, a Dec-POMDP with characteristic-based entities can be described as a tuple \((S, U, O, P, R, E, A, C, m, \Omega, \rho, \gamma)\). We represent the space of entities by \(E\). For any \(e \in E\), the entity \(e\) has a \(d_e\) dimensional state representation \(s_e \in \mathbb{R}^{d_e}\). The global state is therefore the set \(s = \{s_e | e \in E\} \in S\). A subset of the entities are controllable agents \(a \in A \subseteq E\). For both agents and non-agent entities, we differentiate them based on their
characteristics \( c^e \in C \). For example, \( c^e \) can be a continuous vector that consists of two parts such that only one part can be non-zero. That is, if \( e \) is an agent, the first part can represent its skill-level or physical condition, and if \( e \) is a non-agent entity, the second part can represent its entity type. A scenario is a multiset of entities \( c = \{ c^e | e \in E \} \) \( \in \Omega \) and possible scenarios are drawn from the distribution \( \rho(c) \). In other words, scenarios are unique up to the composition of the team and that of world entities: a fixed scenario \( c \) maps to a normal Dec-POMDP with the fixed multiset of entities \( \{ e | c^e \in c \} \).

Given a scenario \( c \), at each environment step, each agent \( a \) can observe a subset of entities specified by \( c^a \) for emphasis. The objective used to train the neural network is: 

\[
Q^\theta_{\text{tot}}(\tau_t, u_t) \approx \max_u Q^\theta(\tau_t, u, c) \in \mathbb{R}
\]

The objective is to maximize the discounted cumulative reward over time: 

\[
G = \mathbb{E}_{\gamma} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]
\]

Here, \( \mathbb{E} \) is a replay buffer that stores previously generated off-policy data. \( Q^\theta_{\text{tot}} \) is the target network parameterized by a delayed copy of \( \theta \) for stability (Mnih et al., 2015).

### 2.2. Value Function Factorization and Attention QMIX

Factorizing the action-value function \( Q \) into per-agent value functions is a popular approach in centralized training and decentralized execution. Specifically, Rashid et al. (2018) propose QMIX, which factorizes \( Q^\theta_{\text{tot}}(\tau_t, u_t) \) into \( Q^\theta(\tau_t, u^a, c^e | a \in A) \) and combines them via a mixing network such that \( \forall a \frac{\partial Q^\theta}{\partial c^a} \ge 0 \). This condition guarantees that individual optimal action \( u^a \) is also the best action for the team. As a result, during execution, the mixing network can be removed and agents can act independently according to their own \( Q^\theta \). Attention QMIX (A-QMIX) (Iqbal et al., 2020) augments the QMIX algorithm with an attention mechanism to deal with an indefinite number of agents/entities. In particular, for each agent, the algorithm applies the multi-head attention (MHA) layer (Vaswani et al., 2017) to summarize the information of the other entities. This information is used for both encoding the agent’s state and adjusting the mixing network. Specifically, the agent’s observation \( o \) is represented by two matrices: the entity state matrix \( X^e \) and the observability matrix \( M \). Assume that in the current scenario \( c \), there exists \( n_e \) entities, \( n_a \) of which are controllable agents, then \( X^e \in \mathbb{R}^{n_e \times d_e \times n_a} \) endows all entities, with the first \( n_a \) rows representing the agents. \( M \in \{0, 1\}^{n_a \times n_e} \) is a binary observability mask and \( M_{ij} = m(a^i, c^j) \) indicates whether agent \( i \) observes entity \( j \). \( X^e \) is first passed through an encoder, i.e., a single-layer feed-forward network, the output of which we refer to as \( \mathcal{H} \). Denote the \( k \)-th row of \( X \) as \( h_k \), then for the \( i \)-th agent, the MHA layer then takes \( h_i \) as the query and \( \{ h_j | M_{ij} = 1 \} \) as the keys to compute a latent representation of \( a^i \)'s observation. For the mixing network, the same MHA layer takes as input \( X^e \) and the full observation matrix \( M^a \), where \( M_{ij}^a = 1 \) if both \( e^i \) and \( e^j \) exist in the scenario \( c \), and outputs the encoded global representation for each agent. These encoded representations are then used to generate the mixing network. For more details, we refer readers to Appendix B of the original A-QMIX paper. While A-QMIX in principle applies to the dynamic team composition problem, it is restricted to fully decentralized execution with partial observation. We leverage the attention modules from A-QMIX but additionally investigate how to efficiently take advantage of global information by introducing a coach.

Iqbal et al. (2020) proposes an extended version of A-QMIX, called Randomized Entity-wise Factorization for Imagined Learning (REFIL), which randomly breaks up the team into two disjoint parts to further decompose the \( Q^\theta_{\text{tot}} \). The authors demonstrate with the imaginary grouping, REFIL outperforms A-QMIX on a gridworld resource allocation task and a modified StarCraft environment. As we show in
the experiment section, we find that REFIL does not always improve over A-QMIX while doubling the computation resource. For this reason, our method is mainly based on A-QMIX, but extending it to REFIL is straightforward.

3. Method

Here we introduce a novel coach-player architecture to incorporate global information for adapting the team-level strategy across different scenarios $c$. We first introduce the coach agent that coordinates base agents with global information by broadcasting strategies periodically. Then we present the learning objective and an additional variational objective to regularize the training. We finish by introducing a method to reduce the broadcast rate and provide analysis to support it.

3.1. On the importance of global information

As the optimal team strategy varies according to the scenario $c$, which includes the team composition, it is important for the team to be aware of any scenario changes as soon as they happen. In an extreme example, consider a multi-agent problem where every agent has its skill-level represented as a Gaussian distribution from which we calculate the log-probability of the strategy, once the dimension $z$ is the strategy, once the team strategy is therefore represented as $z$. Specifically, we assume $z \sim \mathcal{N}(\mu, \Sigma)$, and $(\mu = \{\mu_a | a \in A\}, \Sigma = \{\Sigma_a | a \in A\}) = f_\phi(s; c)$.

For the $T$ steps after receiving $z^0$, agent $a$ will act conditioned on $z^0$. Specifically, within an episode, at time $t_k \in \{t | v \equiv 0 (\text{mod} \ T)\}$, the coach observes the global state $s_k$, and computes and distributes the strategies $z_k$ to all agents. From time $t \in [t_k, t_k + T − 1]$, any agent $a$ will act according to its individual action-value $Q^a_t (\tau^a_t, \cdot | z_t^a, c^a)$.

Denote $\hat{t} = \max \{v | v \equiv 0 (\text{mod} \ T) \ 	ext{and} \ v \leq t\}$, the most recent time step when the coach distributed strategies. The mean square Bellman error objective in (1) becomes

$$\mathcal{L}_{RL}(\theta, \phi) = \mathbb{E}_{(c, \tau, u, s, s_{t+1}) \sim \mathcal{D}} \left[ \left( \tau_1 + \gamma \max_{u'} Q^\text{tot}_{\theta}(\tau_{t+1}, u'|z_{t+1}; c) - Q^\text{tot}_{\theta}(\tau_t, u_t | z_t; c) \right)^2 \right],$$

where $z_t \sim f_\phi(s_t; c)$, $z_{t+1} \sim f_\phi(s_{t+1}; c)$, and $\hat{\phi}$ denotes the parameters of the target network for the coach’s strategy predictor $\hat{f}$. We build our network on top of A-QMIX but use a separate multi-head attention (MHA) layer to encode the global state that the coach observes. For the mixing network, we also use the coach’s output from the MHA layer ($h_{\text{team}}$) for mixing the individual $Q^a$ to form the team $Q^\text{team}$. The entire architecture is illustrated in Figure 2.

3.3. Regularizing with variational objective

Inspired by recent work that applied variational inference to regularize the learning of a latent space in reinforcement learning (Rakelly et al., 2019; Wang et al., 2020a), we also introduce a variational objective to stabilize the training. Intuitively, an agent’s behavior should be consistent with its assigned strategy. In other words, the received strategy should be identifiable from the agent’s future trajectory. Therefore, we propose to maximize the mutual information between the strategy and the agent’s future observation-action pairs $c^a = (a_{t+1}, a_{t+2}, \ldots, a_{t+T-1})$. We maximize the following variational lower bound:

$$I(z^a_t; \zeta^a_t, s_t) \geq \mathbb{E}_{s_t, z^a_t, \zeta^a_t} \left[ \log q_\xi(z^a_t | \zeta^a_t, s_t) \right] + H(z^a_t | s_t). \quad (2)$$

We defer the full derivation to Appendix A. Here $H(\cdot)$ denotes the entropy and $q_\xi$ is the variational distribution parameterized by $\xi$. We further adopt the Gaussian factorization as in Rakelly et al. (2019), i.e.

$$q_\xi(z^a_t | \zeta^a_t, s_t) \propto q_\xi^{(1)}(z^a_t | s_t, u^a_t) \prod_{k=t+1}^{t+T-1} q_\xi^{(k)}(z^a_k | \zeta^a_k, u^a_k),$$

where each $q_\xi^{(i)}$ is a Gaussian distribution. So $q_\xi$ predicts the $\mu^a_t$ and $\Sigma^a_t$ of a multivariate normal distribution from which we calculate the log-probability of
Theorem 1. Denote the optimal action-value and value functions by \( Q^a_\theta \) and \( V^a_\theta \). Denote the action-value and value functions corresponding to receiving new strategies every time from the coach by \( Q^{tot} \) and \( V^{tot} \), and those corresponding to following the strategies distributed according to (3) as \( \hat{Q} \) and \( \hat{V} \). i.e. \( \hat{V}(\tau_t|z_t; c) = \max_a \hat{Q}(\tau_t, a; z_t; c) \). Assume for any trajectory \( \tau_t \), actions \( u_t \), current state \( s_t \), the most recent state the coach distributed strategies \( s_t \), and the players’ characteristics \( c \), \( \|Q^{tot}(\tau_t, u_t, f(s_t); c) - Q^{a}_\theta(s_t, u_t; c)\|_2 \leq \kappa \), and for any strategies \( z^a_t, z_2^a, |Q^{tot}(\tau_t, u_t|z^a_t, z^{a_2}; c) - Q^{tot}(\tau_t, u_t|z^a_t, z^{a_2}; c)| \leq \eta \|z^a_t - z^{a_2}_t\|_2 \). If the used team strategies \( z_t \) satisfy \( \forall a, t, \|z^a_t - z^{a_2}_t\|_2 \leq \beta \), then we have

\[
\|V^{a}_\theta(s_t; c) - \hat{V}(\tau_t|z_t; c)\|_\infty \leq \frac{2(n_a \eta \gamma \beta + \kappa)}{1 - \gamma},
\]

where \( n_a \) is the number of agents and \( \gamma \) is the discount factor.

We defer the proof to Appendix B. The method described in (3) satisfies the condition in Theorem 1 and therefore when \( \beta \) is small, distributing strategies according to (3) results in a bounded drop in performance.

4. Experiments

We design the experiments to 1) verify the effectiveness of the coach agent; 2) investigate how performance varies with the interval \( T \); 3) test if the variational objective is useful; and 4) understand how much the performance drops by adopting the method specified by (3). We test our idea on a resource collection task built on the multi-agent particle environment (Lowe et al., 2017), a multi-agent rescue game, and customized micromanagement tasks in StarCraft.


4.1. Resource Collection

In Resource Collection, a team of agents coordinates to collect different resources spread out on a square map with width 1.8. There are 4 types of entities: the resources, the agents, the home, and the invader. We assume there are 3 types of resources: (r)ed, (g)reen and (b)lue. In the world, always 6 resources appear with 2 of each type. Each agent has 4 characteristics \( c_x^a \) represents how efficiently agent \( x \) collects the resource \( a \), and \( v \) is the agent’s max moving speed. The agent’s job is to collect as many resources as possible, bring them home, and catch the invader if it appears. If \( a \) collects \( x \), the team receives a reward of \( 10 \cdot c_x^a \) as reward. Agents can only hold one resource at a time. After collecting it, the agent must bring the resource home before going out to collect more. Bringing a resource home yields a reward of 1. Occasionally the invader appears and goes directly to the home. Any agent catching the invader will have 4 reward. If the invader reaches the home, the team is penalized by a −4 reward. Each agent has 5 actions: accelerate up / down / left / right and deceleration, and it observes anything within a distance of 0.2. The maximum episode length is 145 time steps. In training, we allow scenarios to have 2 to 4 agents, and for each agent, \( c_x^a, c_x^b, c_x^g \) are drawn uniformly from \( \{0.1, 0.5, 0.9\} \) and the max speed \( v^a \) is drawn from \( \{0.3, 0.5, 0.7\} \). We design 3 testing tasks: 5-agent task, 6-agent task, and a varying-agent task. For each task, we generate 1000 different scenarios c. Each scenario includes \( n_g \) agents, 6 resources, and an invader. For agents, \( c_x^a, c_x^b, c_x^g \) are chosen uniformly from the interval \([0.1, 0.9]\) and \( v^a \) from \([0.2, 0.8]\). For a particular scenario in the varying agent task, starting from 4 agents, the environment randomly adds or drops an agent every \( \nu \) steps as long as the number of agents remains in \([2, 6]\). \( \nu \) is drawn from the uniform distribution \( U(8, 12) \). See Figure 3 for an example run of the learned policy.

Effectiveness of Coach Figure 4(a) shows the training curve when the communication interval is set to \( T = 4 \). The black solid line is a hand-coded greedy algorithm where agents always go for the resource they are best at collecting, and whenever the invader appears, the closest agent goes for it. We see that without global information, A-QMIX and REFIL are significantly worse than the hand-coded baseline. Without the coach, we let all agents have the global view every \( T \) steps in A-QMIX (periodic) but it barely improves over A-QMIX. A-QMIX (full) is fully observable, i.e., all agents have global view. Without the variational objective, COPA performs comparably against A-QMIX (full). With the variational objective, it becomes even better than A-QMIX (full). The results confirm the importance of global coordination and the coach-player hierarchy.

Communication Interval To investigate how performance varies with \( T \), we train with different \( T \) chosen from \([2, 4, 8, 12, 16, 20, 24]\) in Figure 4(b). The performance peaks at \( T = 4 \), counterintuitively smaller that smaller \( T \) is better. This result suggests that the coach is most useful when it can cause the agents to behave smoothly/consistently over time.

Zero-shot Generalization We apply the learned model with \( T = 4 \) to the 3 testing environments. Results are provided in Table 1. The communication frequency is calculated according to the fully centralized setting. For instance, when \( T = 4 \) and \( \beta = 0 \), it results in an average 25% centralization frequency.

| Method         | Env. (n = 5) | Env. (n = 6) | Env. (varying n) | \( f \) |
|----------------|--------------|--------------|------------------|--------|
| Random Policy  | 6.9          | 10.4         | 2.3              | N/A    |
| Greedy Expert  | 115.3        | 142.4        | 71.6             | N/A    |
| REFIL          | 90.5±1.5     | 109.3±1.8    | 61.5±0.9         | 0      |
| A-QMIX         | 96.9±2.1     | 115.1±2.1    | 66.2±1.6         | 0      |
| A-QMIX (periodic) | 93.1±20.4 | 104.2±22.6 | 68.9±12.6 | 0.25 |
| A-QMIX (full)  | 157.4±8.5    | 179.6±9.8    | 114.3±6.2        | 1      |
| COPA (\( \beta = 0 \)) | 175.6±19 | 203.2±2.5 | 124.9±0.9 | 0.25 |
| COPA (\( \beta = 2 \)) | 174.4±1.7 | 200.3±1.6 | 122.8±1.5 | 0.18 |
| COPA (\( \beta = 3 \)) | 168.8±1.7 | 195.4±1.8 | 120.0±1.6 | 0.13 |
| COPA (\( \beta = 5 \)) | 149.3±1.4 | 174.7±1.7 | 104.7±1.6 | 0.08 |
| COPA (\( \beta = 8 \)) | 109.4±3.6 | 130.6±4.0 | 80.6±2.0 | 0.04 |

Table 1. Mean episodic reward on unseen environments with more agents and dynamic team composition. Results are computed from 5 models trained with different seeds. Communication frequency \( (f) \) is compared to communicating with all agents at every step.
As we increase $\beta$ to suppress the distribution of strategies, we see that the performance shows no significant drop until 13\% centralization frequency. Moreover, we apply the same model to 3 environments that are dynamic to different extents (see Figure 5). In the more static environment, resources are always spawned at the same locations. In the medium environment, resources are spawned randomly but there is no invader. The more dynamic environment is the 3rd environment in Table 1 where the team is dynamic in composition and there exists the invader.

In Figure 5, the x-axis is normalized according to the communication frequency $f$ when $\beta = 0$, and the y-axis is normalized by the corresponding performance. As expected, performance drops faster in more dynamic environments as we decrease $f$.

### 4.2. Rescue Game

Search-and-rescue is a natural application for multi-agent systems. In this section we apply COPA to different rescue games. In particular, we consider a $10 \times 10$ grid-world, where each grid contains a building. At any time step, each building is subject to catch a fire. When a building $b$ is on fire, it has an emergency level $c^b \sim U(0, 1)$. Within the world, at most 10 buildings will be on fire at the same time. To protect the buildings, we have $n$ ($n$ is a random number from 2 to 8) robots who are the surveillance firefighters. Each robot $a$ has a skill-level $c^a \in [0, 2]$. A robot has 5 actions, moving up/down/left/right and put out the fire. If $a$ is at a building on fire and chooses to put out the fire, the emergency level will be reduced to $c^b \leftarrow \max(c^b - c^a, 0)$. At each time step $t$, the overall-emergency is given by $c^B_t = \sum_b (c^b)^2$ since we want to penalize greater spread of the fire. The reward is defined as $r_t = c^B_t - \alpha - c^B_t$, the amount of spread that the team prevents. During training, we sample $n$ from $3 - 5$ and these robots are spawned randomly across the world. Each agent’s skill-level is sampled from $[0.2, 0.5, 1.0]$. Then a random number of $3 - 6$ buildings will catch a fire. During testing, we enlarge $n$ to $2 - 8$ agents and sample up to 8 buildings on fire. We summarize the results in the Figure 6.

Interestingly, we find that A-QMIX/REFIL with full views perform worse than A-QMIX with partial views. We conjecture this is because the full view complicates learning. On the other hand, COPA consistently outperforms all baselines even with a communication frequency as low as 0.15. In addition, we conduct an ablation on the sight range of the players in Figure 7. As we increase the sight range of the players from 1 to 5, where 5 is equivalent to granting full views to all players,
the performance on test scenarios drops. This result raises the possibility that agents and players might benefit from having different views of the environment.

4.3. StarCraft Micromanagement Tasks

Finally, we apply COPA on the more challenging StarCraft multi-agent challenge (SMAC) (Samvelyan et al., 2019). SMAC involves controlling a team of units to defend a team of enemy units. We test on the two settings, 3-8sz and 3-8MMM, from Iqbal et al. (2020). Here, 3-8sz means the scenario consists of symmetric teams of between 3 to 8 units, with each agent being either a Zealot or a Stalker. Similarly, 3-8MMM has between 0 and 2 Medics and 3 to 6 Marines/Marauders. In Iqbal et al. (2020), both teams are grouped together and all agents can fully observe their teammates. To make the task more challenging, we modified both environments such that the agent team is randomly divided into 2 to 4 groups while the enemy team is organized into 1 or 2 groups. Each group is spawned together at a random place. Specifically, the center of any enemy group is drawn uniformly on the boundary of a circle of radius 7, while the center of any group of the controlled agents is drawn uniformly from within the circle of radius 6. Then we grant each unit a sight range of 3. The learning curves are shown in Figure 8.

As we find that on SMAC environments, the imaginary objective proposed in (Iqbal et al., 2020) further improves the performance of A-QMIX, we also incorporate the imaginary objective for COPA, denoted by COPA(im). Then we compare COPA(im) with A-QMIX, REFIL, and REFIL with the omniscient view at the same maximum frequency (e.g. $T = 4$) that COPA distributes the strategies, denoted by REFIL(periodic). COPA outperforms A-QMIX and REFIL without the omniscient views by a large margin. Moreover, COPA performs slightly better than REFIL (periodic). But note that COPA can dynamically control the communication frequency which REFIL(periodic) cannot. We further provide the results under different communication frequencies in Table 2. COPA’s performance remains strong with the communication frequency as low as 0.04 and 0.10 respectively on 3-8sz and 3-8MMM environments.

![Figure 8](image)

**Figure 8.** Training curves of the win rate on the 3-8sz and 3-8MMM environments. COPA(im) indicates COPA applied with the additional imaginary grouping objective as in REFIL (Iqbal et al., 2020). Each experiment is conducted over 3 independent runs. The shaded area indicates the standard error of the mean.

| Method          | 3-8sz Win Rate % | 3-8 MMM Win Rate % |
|-----------------|------------------|---------------------|
| REFIL (periodic)| 12.90 ± 3.36     | 5.78 ± 0.23         |
| COPA ($\beta = 0$) | 26.07 ± 3.03 | 11.62 ± 1.23        |
| COPA ($\beta = 1$) | 26.61 ± 2.73 | 10.42 ± 1.63        |
| COPA ($\beta = 2$) | 22.65 ± 1.34 | 10.35 ± 0.84        |
| COPA ($\beta = 3$) | 14.25 ± 1.11  | 9.01 ± 1.17         |

**Table 2.** Performance on 3-8sz and 3-8MMM versus the communication frequency ($f$) controlled by $\beta$. Results are computed from 3 models on 500 test episodes.

5. Related Work

In addition to the related work highlighted in Section 2, here we review other relevant works in four categories.

Centralized Training with Decentralized Execution

Centralized training with decentralized execution (CTDE) assumes agents execute independently but uses the global information for training. A branch of methods investigates factorizable $Q$ functions (Sunehag et al., 2017; Rashid et al., 2018; Mahajan et al., 2019; Son et al., 2019) where the team $Q$ is decomposed into individual utility functions. Some other methods adopt the actor-critic method where only the critic is centralized (Foerster et al., 2017; Howe et al., 2017). However, most CTDE methods by structure require fixed-size teams and are often applied to homogeneous teams.

Methods for Dynamic Team Compositions

Several recent works in transfer learning and curriculum learning attempt to transfer policy of small teams for larger teams (Carion et al., 2019; Shu & Tian, 2019; Agarwal et al., 2019; Wang et al., 2020b; Long et al., 2020). These works mostly consider teams with different numbers of homogeneous agents. Concurrently, Iqbal et al. (2020) and Zhang et al. (2020) also consider using the attention mechanism to deal with a variable number of agents. But both methods are fully decentralized and we focus on studying how to incorporate global information for ad hoc teams when useful.

Ad Hoc Teamwork and Networked Agents

Standard ad hoc teamwork research mainly focuses on the single ad hoc agent and assumes no control over the teammates (Genter et al., 2011; Barrett & Stone, 2012). Recently, Grizou et al. (2016) and Mirsky et al. (2020) consider communication between ad hoc agents but assume the communication protocol is pre-defined. Decentralized networked agents assume...
information can propagate among agents and their neighbors (Kar et al., 2013; Macua et al., 2014; Foerster et al., 2016; Suttle et al., 2019; Zhang et al., 2018). However, to our knowledge, networked agents have not been applied to teams with dynamic compositions yet.

Hierarchical Reinforcement Learning Hierarchical RL/MARL decomposes the task into hierarchies: a meta-controller selects either a temporal abstracted action (Bacon et al., 2017), called an option, or a goal state (Vezhnevets et al., 2017) for the base agents. Then the base agents shift their purposes to finish the assigned option or reach the goal. Therefore usually the base agents have different learning objective from the meta-controller. Recent deep MARL methods also demonstrate role emergence (Wang et al., 2020a) or skill emergence (Yang et al., 2019). But the inferred role/skill is only conditioned on the individual trajectory. The coach in our method uses global information to determine the strategies for the base agents. To our knowledge, we are the first to apply such a hierarchy for teams with varying numbers of heterogeneous agents.

6. Conclusion
We investigated a new setting of multi-agent reinforcement learning problems, where both the team size and members’ capabilities are subject to change. To this end, we proposed a coach-player framework, COPA, where the coach coordinates with a global view but players execute independently with local views and the coach’s strategy. We developed a variational objective to regularize the learning of the strategies and introduces an intuitive method to suppress unnecessary distribution of strategies. Results on three different environments demonstrate that the coach is important for learning when the team composition might change. COPA achieves strong zero-shot generalization performance with relatively low communication frequency. Interesting future directions include investigating better communication strategies between the coach and the players and introducing multiple coach agents with varying levels of views.

References
Agarwal, A., Kumar, S., and Sycara, K. Learning transferable cooperative behavior in multi-agent teams. arXiv preprint arXiv:1906.01202, 2019.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

Barrett, S. and Stone, P. An analysis framework for ad hoc teamwork tasks. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), June 2012. URL http://www.cs.utexas.edu/users/ai-lab?AAMAS12-Barrett.

Barrett, S., Agmon, N., Hazon, N., Kraus, S., and Stone, P. Communicating with unknown teammates. In ECAI, pp. 45–50, 2014.

Cao, Y., Yu, W., Ren, W., and Chen, G. An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial informatics, 9(1):427–438, 2012.

Carion, N., Usunier, N., Synnaeve, G., and Lazaric, A. A structured prediction approach for generalization in cooperative multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, pp. 8130–8140, 2019.

Choi, J., Oh, S., and Horowitz, R. Distributed learning and cooperative control for multi-agent systems. Automatica, 45(12):2802–2814, 2009.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

de Witt, C. S., Foerster, J., Farquhar, G., Torr, P., Boehmer, W., and Whiteson, S. Multi-agent common knowledge reinforcement learning. In Advances in Neural Information Processing Systems, pp. 9927–9939, 2019.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926, 2017.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

Foerster, J. N., Assael, Y. M., De Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:1605.06676, 2016.

Genter, K., Agmon, N., and Stone, P. Role-based ad hoc teamwork. In Proceedings of the Plan, Activity, and Intent Recognition Workshop at the Twenty-Fifth Conference on Artificial Intelligence (PAIR-11), August 2011. URL http://www.cs.utexas.edu/users/ai-lab?PAIR11-katie.

Grizou, J., Barrett, S., Stone, P., and Lopes, M. Collaboration in ad hoc teamwork: ambiguous tasks, roles, and communication. In AAMAS Adaptive Learning Agents (ALA) Workshop, Singapore, 2016.
Iqbal, S., de Witt, C. A. S., Peng, B., Böhmer, W., Whiteson, S., and Sha, F. Ai-qnix: Attention and imagination for dynamic multi-agent reinforcement learning. *arXiv preprint arXiv:2006.04222*, 2020.

Kar, S., Moura, J. M., and Poor, H. V. Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+innovations. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, 2013.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kurach, K., Raichuk, A., Stańczyk, P., Zajęc, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4501–4510, 2020.

Lee, J. W., Park, J., Jangmin, O., Lee, J., and Hong, E. A multiagent approach to q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6):864–877, 2007.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

Long, Q., Zhou, Z., Gupta, A., Fang, F., Wu, Y., and Wang, X. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*, 2020.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390, 2017.

Macua, S. V., Chen, J., Zazo, S., and Sayed, A. H. Distributed policy evaluation under multiple behavior strategies. *IEEE Transactions on Automatic Control*, 60(5):1260–1274, 2014.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pp. 7613–7624, 2019.

Medsker, L. and Jain, L. C. *Recurrent neural networks: design and applications*. CRC press, 1999.

Mirska, R., Macke, W., Wang, A., Yedidsion, H., and Stone, P. A penny for your thoughts: The value of communication in ad hoc teamwork. *Good Systems-Published Research*, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340, 2019.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Shu, T. and Tian, Y. M<sup>3</sup>rl: Mind-aware multi-agent management reinforcement learning, 2019.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1905.05408*, 2019.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Suttle, W., Yang, Z., Zhang, K., Wang, Z., Basar, T., and Liu, J. A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning. *arXiv preprint arXiv:1903.06372*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

Wang, T., Dong, H., Lesser, V., and Zhang, C. Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020a.
Wang, W., Yang, T., Liu, Y., Hao, J., Hao, X., Hu, Y., Chen, Y., Fan, C., and Gao, Y. From few to more: Large-scale dynamic multiagent curriculum learning. In *AAAI*, pp. 7293–7300, 2020b.

Yang, J., Borovikov, I., and Zha, H. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *arXiv preprint arXiv:1912.03558*, 2019.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. Fully decentralized multi-agent reinforcement learning with networked agents. *arXiv preprint arXiv:1802.08757*, 2018.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Multi-agent collaboration via reward attribution decomposition. *arXiv preprint arXiv:2010.08531*, 2020.

Zhu, P., Li, X., Poupart, P., and Miao, G. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.
A. Regularizing with variational objective

We provide the full derivation of Equation (2) in the following:

\[
I(z^a_t; \zeta^a_t, s_t) = \mathbb{E}_{s_t, z^a_t, \zeta^a_t} \left[ \log \frac{p(z^a_t|\zeta^a_t, s_t)}{p(z^a_t|s_t)} \right] \quad \text{//by the definition of mutual information}
\]

\[
= \mathbb{E}_{s_t, z^a_t, \zeta^a_t} \left[ \log q_{\xi}(z^a_t|\zeta^a_t, s_t) \right] + \text{KL} \left( \frac{p(z^a_t|\zeta^a_t, s_t)}{p(z^a_t|s_t)} \right)
\]

\[
\geq \mathbb{E}_{s_t, z^a_t, \zeta^a_t} \left[ \log q_{\xi}(z^a_t|\zeta^a_t, s_t) \right] \quad \text{//since KL(\cdot, \cdot) \geq 0}
\]

\[
= \mathbb{E}_{s_t, z^a_t, \zeta^a_t} \left[ \log q_{\xi}(z^a_t|\zeta^a_t, s_t) \right] + H(z^a_t|s_t).
\]

B. Proof of Theorem 1

In this section we provide the proof for Theorem 1.

Theorem 1. Denote the optimal action-value and value functions by \(Q^\text{tot} \) and \(V^\text{tot} \). Denote the action-value and value functions corresponding to receiving new strategies every time from the coach by \(Q^\text{tot} \) and \(V^\text{tot} \), and thus corresponding to following the strategies distributed according to (3) as \(\bar{Q} \) and \(\bar{V} \), i.e. \(\bar{V}(\tau_t|\tilde{z}_i; c) = \max_a \bar{Q}(\tau_t, u|\tilde{z}_i; c)\). Assume for any trajectory \(\tau_t \), actions \(u_t \), current state \(s_t \), the most recent state the coach distribute strategies \(s_t \), and the players’ characteristics \(c \), \(|Q^\text{tot}(\tau_t, u_t, f(s_t); c) - Q^\text{tot}(s_t, u_t; c)| \leq \kappa \), and for any strategies \(z^a_t, z^b_t \), \(|Q^\text{tot}(\tau_t, u_t|z^a_t, z^{-a}; c) - Q^\text{tot}(\tau_t, u_t|z^b_t, z^{-a}; c)| \leq \eta \|z^a_t - z^b_t\|_2 \). If the used team strategies \(\tilde{z}_i \) satisfies \(\forall a, \|z^a_t - z^\text{tot}_a\|_2 \leq \beta \), then we have

\[
\|V^\text{tot}_t(s_t; c) - \bar{V}(\tau_t|\tilde{z}_i; c)\|_\infty \leq \frac{2(n_a \eta \beta + \kappa)}{1 - \gamma},
\]

where \(n_a \) is the number of agents and \(\gamma \) is the discount factor.

To summarize, the assumptions assume that the learned action-value function \(Q^\text{tot} \) approximates the optimal \(Q^*_a \) well and has bounded Lipschitz constant with respect to individual action-value functions. Moreover, we assume the individual action-value functions also have bounded Lipschitz constant with respect to the strategies.

Proof. According to Assumption 2, if \(\|\tilde{z}_i - z^a_t\|_2 \leq \beta \) for all \(a \), then

\[
|Q^\text{tot}(\tau_t, u_t|\tilde{z}_i, c) - Q^\text{tot}(\tau_t, u_t|z^a_t, c)| \leq \sum_{a, 1 \leq i \leq n_a} \eta_1 \eta_2 \|\tilde{z}_i - z^a_t\|_2 \leq n_a \eta_1 \eta_2 \beta.
\]

For notation convenience, we ignore the superscript of \(\text{tot} \) and the condition on \(c \). For a state \(s \), denote the action the learned policy take as \(u^\dagger \), i.e. \(u^\dagger \equiv \text{argmax}_a Q(\tau, u) \). Similarly we can define \(u^* \) and \(\bar{u} \) as the action one would take according to the optimal \(Q^*_a \) and the action-value \(Q \) estimated using the old strategy. From Assumption 1, we know that

\[
Q^*_a(s, u^\dagger) \geq Q(\tau, u^\dagger) - \kappa \geq Q(\tau, u^*) - \kappa \geq Q^*_a(s, u^*) - 2\kappa.
\]

Therefore taking \(u^\dagger \) will result in at most \(2\kappa \) performance drop at this single step. Similarly, denote \(\epsilon_0 = n_a \eta_1 \eta_2 \beta \), then

\[
Q(\tau, \bar{u}) \geq Q(\tau, \bar{u}) - \epsilon_0 \geq \bar{Q}(\tau, u^\dagger) - \epsilon_0 \geq Q(\tau, u^\dagger) - 2\epsilon_0.
\]

Hence \(Q^*_a(s, \bar{u}) \geq Q^*_a(s, u^*) - 2(\epsilon_0 + \kappa) \). Note that this means taking the action \(\bar{u} \) in the place of \(u^* \) at state \(s \) will result in at most \(2(\epsilon_0 + \kappa) \) performance drop. This conclusion generalizes to any step \(t \). Therefore, if at each single step the performance is bounded within \(2(\epsilon_0 + \kappa) \), then overall the performance is within \(2(\epsilon_0 + \kappa)/(1 - \gamma) \).

C. Training Details

For both Resource Collection and Rescue Game, we set the max total number of training steps to 5 million. Then we use the exponentially decayed \(\epsilon \)-greedy algorithm as our exploration policy, starting from \(\epsilon_0 = 1.0 \) to \(\epsilon_n = 0.05 \). We parallelize the
environment with 8 threads for training. Experiments are run on the GeForce RTX 2080 GPUs. We provide the algorithm hyper-parameters in Table 3.

For StarCraft Micromanagement, we follow the same setup from (Iqbal et al., 2020) and train all methods on the 3-8sz and 3-8MMM maps for 12 millions steps. To regularize the learning, we use $\lambda_1 = 0.00005$ and $\lambda_2 = 0.000005$ for both maps. For all experiments, we set the default period before centralization to $T = 4$.

### Table 3. Hyper-parameters in Resource Collection and Rescue Game.

| Name         | Description                                                      | Value          |
|--------------|------------------------------------------------------------------|----------------|
| $|D|$          | replay buffer size                                              | 100000         |
| $n_{\text{head}}$ | number of heads in multi-head attention                        | 4              |
| $n_{\text{thread}}$ | number of parallel threads for running the environment       | 8              |
| $dh$         | the hidden dimension of all modules                             | 128            |
| $\gamma$    | the discount factor                                             | 0.99           |
| $lr$        | learning rate                                                   | 0.0003         |
| $\alpha$    | $\alpha$ value in RMSprop                                       | 0.99           |
| $\epsilon$ | $\epsilon$ value in RMSprop                                      | 0.00001        |
| $n_{\text{batch}}$ | batch size                                                    | 256            |
| grad clip   | clipping value of gradient                                      | 10             |
| target update frequency | how frequent do we update the target network                  | 200 updates    |
| $\lambda_1$ | $\lambda_1$ in variational objective                           | 0.001          |
| $\lambda_2$ | $\lambda_2$ in variational objective                           | 0.0001         |