Multi-Agent Coordination via Multi-Level Communication

Anonymous authors
Paper under double-blind review

Abstract

In a multi-agent system, an agent normally can only access parts of the information of the state (partial observability) and the behaviors of others may keep changing (stochasticity) during the training procedure. Agents can obtain more information via communication to better understand the state and the behavior of others. However, the coordination problem still exists since agents sometimes infer incorrect others' actions based on observations. It is also not possible to communicate actions directly at the same time. Otherwise, all agents need to make decisions based on others' actions, leading to circular dependencies. In this paper, we propose a novel multi-level communication scheme, *Sequential Communication* (SeqComm). SeqComm treats agents asynchronously (each agent is assigned a different priority of decision-making, and the higher the priority of decision-making, the higher level the agent is). In addition, we have two communication phases. The negotiation phase is used to determine the priority of decision-making for agents. Agents first communicate hidden states of observations with others. Then, agents communicate and compare the corresponding values of agents' intentions to determine the priority of decision-making. The value of each intention represents the predicted rewards of future behavior without considering others by a learned world model (modeling the environmental dynamics). In the launching phase, the upper-level agents take the lead in making decisions and then communicate their actions with the lower-level agents. Theoretically, we prove the policies learned by SeqComm are guaranteed to improve monotonically and converge. Empirically, we show that SeqComm outperforms existing methods in a variety of cooperative multi-agent tasks.

1 Introduction

The partial observability and stochasticity that are inherent to the nature of multi-agent systems can easily impede the cooperation among agents and lead to catastrophic miscoordination (Ding et al., 2020). Communication has been exploited to help agents obtain extra information during both training and execution to mitigate such problems (Foerster et al., 2016; Sukhbaatar et al., 2016; Peng et al., 2017). Specifically, agents can share their information with others via a trainable communication channel.

Centralized training with decentralized execution (CTDE) (Lowe et al., 2017) is a popular learning paradigm in cooperative multi-agent reinforcement learning (MARL). Although the centralized value function can be learned to evaluate the joint policy of agents, the decentralized policies of agents are essentially independent. Therefore, a coordination problem arises. That is, agents may make sub-optimal actions by mistakenly assuming others' actions when there exist multiple optimal joint actions (Busoniu et al., 2008). Communication allows agents to obtain information about others to avoid miscoordination. However, most existing work only focuses on communicating messages, *e.g.,* the information of agents' current observation or historical trajectory (Jiang & Lu, 2018; Singh et al., 2019; Das et al., 2019; Ding et al., 2020). It is impossible for an agent to acquire other's actions before making decisions since the game model is usually synchronous, *i.e.*, agents make decisions and execute actions simultaneously. Note that the execution rules depend on the game model, but the rules of how agents make decisions depend on the algorithm itself.

A general approach to solving the coordination problem is to make sure that ties between equally good actions are broken by all agents. One simple mechanism for doing so is to know exactly what others will do and adjust the behavior accordingly under a unique ordering of agents and actions (Busoniu et al., 2008). Inspired by this, we reconsider the cooperative game from an asynchronous perspective. In other words, each agent is assigned a priority (*i.e.* order) of decision-making at each step in both training and execution, thus the Stackelberg equilibrium (SE) (Von Stackelberg, 2010) is naturally set up as the learning objective. Specifically, the upper-level agents make decisions before the lower-level agents. Therefore, the lower-level agents can acquire the actual actions of the upper-level agents by communication and make their decisions conditioned on what the upper-level agents would do. Under this setting, the SE is likely to be Pareto superior to the average Nash equilibrium (NE) in games that require a high cooperation level (Zhang et al., 2020). However, *is it necessary to decide a specific priority of decision-making for each agent?* Ideally, the optimal joint policy can be decomposed by any orders (Wen et al., 2019), *e.g.,* $\pi^*(a_1, a_2|s) = \pi^*(a_1|s)\pi^*(a_2|s, a_1) = \pi^*(a_2|s)\pi^*(a_1|s, a_2)$. But during the learning process, it is unlikely for agents to use the optimal actions of other agents for gradient calculation, making it still vulnerable to the relative overgeneralization problem (Wei et al., 2018). It means there is no guarantee that different orders can converge to the same suboptimal. We also claim that the different priorities of decision-making may affect the optimality of the convergence of the learning algorithm in Section 3. Note that relative overgeneralization occurs when a suboptimal NE in the joint space of actions is preferred over an optimal NE because each agent's action in the suboptimal equilibrium is a better choice when matched with arbitrary actions from the cooperative agents.

In this paper, we propose a novel multi-level communication scheme for cooperative MARL, *Sequential Communication* (SeqComm), to enable agents to explicitly coordinate with each other. Specifically, SeqComm has two-phase communication, negotiation phase and launching phase. In the negotiation phase, agents communicate their hidden states of observations with others simultaneously. Then they can generate multiple predicted trajectories, called *intention*, by modeling the environmental dynamics and other agents' actions. In addition, the priority of decision-making is determined by communicating and comparing the corresponding values of agents' intentions. The value of each intention represents the predicted rewards obtained by treating that agent as the first mover of the order sequence. The sequence of others follows the same procedure as aforementioned with the upper-level agents fixed. In the launching phase, the upper-level agents take the lead in decision-making and communicate their actual actions with the lower-level agents. Note that the actual actions will be executed simultaneously in the environment without any changes.

SeqComm is currently built on MAPPO (Yu et al., 2021). Theoretically, we prove the policies learned by SeqComm are guaranteed to improve monotonically and converge. Empirically, we evaluate SeqComm on a set of tasks in multi-agent particle environment (MPE) (Lowe et al., 2017) and StarCraft multi-agent challenge (SMAC) (Samvelyan et al., 2019). In all these tasks, we demonstrate that SeqComm outperforms existing communication-free and communication-based methods. By ablation studies, we confirm that treating agents asynchronously is a more effective way to promote coordination and SeqComm can provide the proper priority of decision-making for agents to develop better coordination.

2 Related Work

Communication. Existing work (Jiang & Lu, 2018; Kim et al., 2019; Singh et al., 2019; Das et al., 2019; Zhang et al., 2019; Jiang et al., 2020; Ding et al., 2020; Konan et al., 2022) in this realm mainly focus on how to extract valuable messages. ATOC (Jiang & Lu, 2018) and IC3Net (Singh et al., 2019) utilize gate mechanisms to decide when to communicate with other agents. Several studies (Das et al., 2019; Konan et al., 2022) employ multi-round communication to fully reason the intentions of others and establish complex collaboration strategies. Social influence (Jaques et al., 2019) uses communication to influence the behaviors of others. I2C (Ding et al., 2020) only communicates with agents that are relevant and influential which are determined by causal inference. However, all these methods focus on how to exploit valuable information from current or past partial observations effectively and properly.

More recently, some studies (Kim et al., 2021; Du et al., 2021; Pretorius et al., 2021) begin to answer the question: can we favor cooperation beyond sharing partial observation? They allow agents to imagine their future states with a world model and communicate those with others. IS (Pretorius et al., 2021), as the representation of this line of research, enables each agent to share its intention with other agents in the form of the encoded imagined trajectory and use the attention module to figure out the importance of the received intentions. However, two concerns arise. On one hand, circular dependencies can lead to inaccurate predicted future trajectories as long as the multi-agent system treats agents synchronously. On the other hand, MARL struggles to extract useful information from numerous messages, not to mention more complex and dubious messages, *i.e.* predicted future trajectories. Unlike these studies, we treat the agents from an asynchronous perspective therefore circular dependencies can be naturally resolved. Furthermore, agents send actions to lower-level agents, making the messages compact as well as informative.

Coordination. The agents are essentially independent decision-makers in execution and may break ties between equally good actions randomly. Thus, in the absence of additional mechanisms, different agents may break ties in different ways, and the resulting joint actions may be suboptimal. Coordination graphs (Guestrin et al., 2002; Böhmer et al., 2020; Wang et al., 2021b) simplify the coordination when the global Q-function can be additively decomposed into local Q-functions that only depend on the actions of a subset of agents. Typically, a coordination graph expresses a higher-order value decomposition among agents. This improves the representational capacity to distinguish other agents' effects on local utility functions, which addresses the miscoordination problems caused by partial observability.

Another general approach to solving the coordination problem is to make sure that ties are broken by all agents in the same way, requiring that random action choices are somehow coordinated or negotiated. Social conventions (Boutilier, 1996) or role assignments (Prasad et al., 1998) encode prior preferences towards certain joint actions and help break ties during action selection. Communication (Fischer et al., 2004; Vlassis, 2007) can be used to negotiate action choices, either alone or in combination with the aforementioned techniques. Our method follows this line of research by utilizing the ordering of agents and actions to break the ties, other than the enhanced representational capacity of the local value function.

Reinforcement Learning in Stackelberg Game. Many previous studies (Könönen, 2004; Sodomka et al., 2013; Greenwald et al., 2003; Zhang et al., 2020) have investigated reinforcement learning in finding Stackelberg equilibrium. Bi-AC (Zhang et al., 2020) is a bi-level actor-critic method that allows agents to have different knowledge bases so that Stackelberg equilibrium (SE) is possible to find. The actions still can be executed simultaneously and distributedly. It empirically studies the relationship between the cooperation level and the superiority of Stackelberg equilibrium to Nash equilibrium. AQL (Könönen, 2004) updates the Q-value by solving the SE in each iteration and can be regarded as the value-based version of Bi-AC.

Existing work mainly focuses on two-agent settings and their order is fixed in advance. However, fixed order can hardly be an optimal solution especially when it comes to large-scale homogeneous agent scenarios. To address this issue, we exploit agents' intentions to dynamically determine the priority of decision-making along the way of interacting with each other.

Multi-Agent Path Finding (MAPF). MAPF aims to plan collision-free paths for multiple agents on a given graph from their given start vertices to target vertices. In MAPF, prioritized planning is deeply coupled with collision avoidance (Van Den Berg & Overmars, 2005; Ma et al., 2019), where collision is used to design constraints or heuristics for planning.

Unlike MAPF, our method couples the priority of decision-making with the learning objective and thus is more general. In addition, the different motivations and problem settings may lead to the incompatibility of the algorithms in the two fields.

3 Problem Formulation

Cost-Free Communication. The decentralized partially observable Markov decision process (Dec-POMDP) can be extended to multi-agent POMDP (Oliehoek et al., 2016) by sharing observations among agents via communication. The joint observations are not necessarily equivalent to the state. However, the joint observations can be used to better represent the state than the single observation if the state is not accessible.

Pynadath & Tambe (2002) showed that under cost-free communication, agents would share optimal messages for mutual interest. If the communication cost is high, there is a balance between delivering all the useful messages for greater benefits and keeping the amount of communication as low as possible. In addition, analyzing this extreme case gives us some understanding of the benefit of communication, even if the results do not apply across all domains. However, even under multi-agent POMDP where agents can get joint observations, coordination problems can still arise (Busoniu et al., 2008). Suppose the centralized critic has learnt actions pairs $[a_1, a_2]$ and $[b_1, b_2]$ that are equally optimal. Without any prior information, the individual policies π_1 and π_2 learned from the centralized critic can break the ties randomly and may choose a_1 and b_2, respectively.

Multi-Agent Sequential Decision-Making. We consider fully cooperative multi-agent tasks that are modeled as multi-agent POMDP, where n agents interact with the environment according to the following procedure, which we refer to as *multi-agent sequential decision-making.*

At each timestep t, assume the priority (*i.e.* order) of decision-making for all agents is given and each priority level has only one agent (*i.e.*, agents make decisions one by one). Note that the smaller the level index, the higher the priority of decision-making is. The agent at each level k gets its own observation o_t^k drawn from the state s_t, and receives messages $\boldsymbol{m}_t^{-k}$ from all other agents, where $\boldsymbol{m}_t^{-k} \triangleq \{\{o_t^1, a_t^1\}, \dots, \{o_t^{k-1}, a_t^{k-1}\}, o_t^{k+1}, \dots, o_t^n\}$. Equivalently, $\boldsymbol{m}_t^{-k}$ can be written as $\{\boldsymbol{o}_t^{-k}, \boldsymbol{a}_t^{1:k-1}\}$, where $\boldsymbol{o}_t^{-k}$ denotes the joint observations of all agents except k (in practice, agents communicate the hidden states/encodings of observations), and $\boldsymbol{a}_t^{1:k-1}$ denotes the joint actions of agents 1 to $k-1$. For the agent at the first level (*i.e.*, $k=1$), $\boldsymbol{a}_t^{1:k-1} = \varnothing$. Then, the agent determines its action a_t^k sampled from its policy $\pi_k(\cdot|o_t^k, \boldsymbol{m}_t^{-k})$ or equivalently $\pi_k(\cdot|\boldsymbol{o}_t, \boldsymbol{a}_t^{1:k-1})$ and sends it to the lower-level agents. After all agents have determined their actions, they perform the joint actions $\boldsymbol{a}_t$, which can be seen as sampled from the joint policy $\boldsymbol{\pi}(\cdot|s_t)$ *factorized* as $\prod_{k=1}^n \pi_k(\cdot|\boldsymbol{o}_t, \boldsymbol{a}_t^{1:k-1})$, in the environment and get a shared reward $r(s_t, \boldsymbol{a}_t)$ and the state transitions to next state s' according to the transition probability $p(s'|s_t, \boldsymbol{a}_t)$. All agents aim to maximize the expected return $\sum_{t=0}^{\infty} \gamma^t r_t$, where γ is the discount factor. The state-value function and action-value function of the level-k agent are defined as follows:

$$V_{\pi_k}(s, \boldsymbol{a}^{1:k-1}) \triangleq \mathop{\mathbb{E}}_{\substack{s_{1:\infty} \\ \boldsymbol{a}_0^{k:n} \sim \boldsymbol{\pi}_{k:n} \\ \boldsymbol{a}_{1:\infty} \sim \boldsymbol{\pi}}} [\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \boldsymbol{a}_0^{1:k-1} = \boldsymbol{a}^{1:k-1}]$$

$$Q_{\pi_k}(s, \boldsymbol{a}^{1:k}) \triangleq \mathop{\mathbb{E}}_{\substack{s_{1:\infty} \\ \boldsymbol{a}_0^{k+1:n} \sim \boldsymbol{\pi}_{k+1:n} \\ \boldsymbol{a}_{1:\infty} \sim \boldsymbol{\pi}}} [\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \boldsymbol{a}_0^{1:k} = \boldsymbol{a}^{1:k}].$$

For the setting of multi-agent sequential decision-making discussed above, we have the following proposition.

Proposition 1. *If all the agents update their policy with individual TRPO (Schulman et al., 2015) sequentially in multi-agent sequential decision-making, then the joint policy of all agents are guaranteed to improve monotonically and converge.*

Proof. The proof is given in Appendix B. □

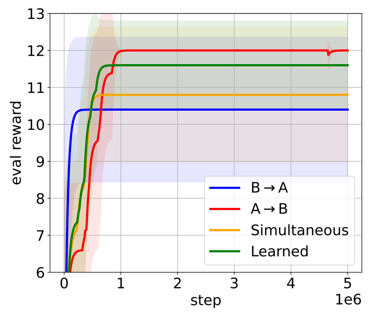
Proposition 1 indicates that SeqComm has the performance guarantee regardless of the priority of decision-making in multi-agent sequential decision-making. However, the priority of decision-making indeed affects the optimality of the converged joint policy, and we have the following claim.

Claim 1. *The different priorities of decision-making may affect the optimality of the convergence of the learning algorithm due to the relative overgeneralization problem.*

We use a one-step matrix game as an example, as illustrated in Figure 1(a), to demonstrate the influence of the priority of decision-making on the learning process. Due to relative overgeneralization (Wei et al., 2018), agent B tends to choose b_2 or b_3. Specifically, b_2 or b_3 in the suboptimal equilibrium is a better choice than b_1 in the optimal equilibrium when matched with arbitrary actions from agent A. Therefore, as shown in Figure 1(b), $B \rightarrow A$ (*i.e.*, agent B makes decisions before A, and A's policy conditions on the action of B) and *Simultaneous* (*i.e.*, two agents make decisions simultaneously and independently) are easily trapped into local optima. However, things can be different if agent A goes first, as $A \rightarrow B$ achieves the optimum. As long as agent A does not suffer from relative overgeneralization, it can help agent B get rid of local optima by narrowing down the search space of B. Besides, a policy that determines the priority of decision-making can be learned under the guidance of the state-value function, denoted as *Learned*. It obtains better performance

		Agent B		
		b_1	b_2	b_3
Agent A	a_1	12	6	6
	a_2	-6	8	0
	a_3	-6	0	8

(a) payoff matrix of the game



(b) evaluations of different methods

Figure 1: (a) Payoff matrix for a one-step game. There are multiple local optima. (b) Evaluations of different methods for the game in terms of the mean reward and standard deviation of ten runs. $A \rightarrow B$, $B \rightarrow A$, *Simultaneous*, and *Learned* represent that agent A makes decisions first, agent B makes decisions first, two agents make decisions simultaneously, and there is another learned policy determining the priority of decision making, respectively. MAPPO (Yu et al., 2021) is used as the backbone.

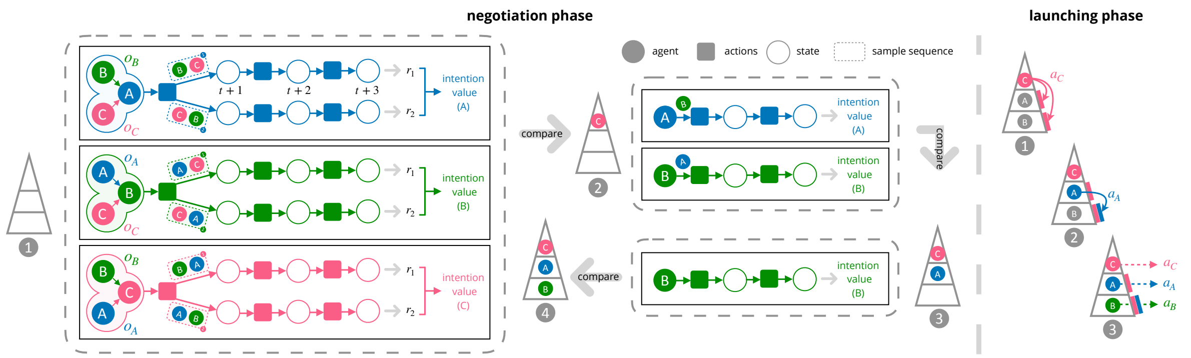


Figure 2: Overview of SeqComm. SeqComm has two communication phases, the negotiation phase (*left*) and the launching phase (*right*). In the negotiation phase, agents communicate hidden states of observations with others and obtain their own intentions. The priority of decision-making is determined by sharing and comparing the value of all the intentions. In the launching phase, the agents who hold the upper-level positions will make decisions prior to the lower-level agents. Besides, their actions will be shared with anyone who has not yet made decisions.

than $B \rightarrow A$ and *Simultaneous*, which indicates that dynamically determining the order during policy learning can be beneficial as we do not know the optimal priority in advance.

Remark 1. *The priority (*i.e.* order) of decision-making affects the optimality of the converged joint policy in multi-agent sequential decision-making, thus it is critical to determine the order. However, learning the order directly requires an additional centralized policy in execution, which is not generalizable in a scenario where the number of agents varies. Moreover, its learning complexity exponentially increases with the number of agents, making it infeasible in many cases.*

4 Sequential Communication

In this paper, we cast our eyes in another direction and resort to the world model. Ideally, we can randomly sample candidate order sequences, evaluate them under the world model (see Section 4.1), and choose the order sequence that is deemed the most promising under the true dynamic. SeqComm is designed based on this principle to determine the priority of decision-making via communication.

In SeqComm, communication is separated into phases serving different purposes and multi-round communication in one phase is possible. One is the *negotiation* phase for agents to determine the priority of decision-making. Another is the *launching* phase for agents to act conditioning on actual actions upper-level agents will take to implement *explicit coordination via communication.* The overview of SeqComm is illustrated in Figure 2. Each SeqComm agent consists of a policy, a critic, and a world model, as illustrated in Figure 3, and the parameters of all networks are shared across agents (Gupta et al., 2017).

4.1 Negotiation Phase

In the negotiation phase, the observation encoder first takes o_t as input and outputs a hidden state h_t to compress the information, which is used to communicate with others. Note that many studies (Ding et al., 2020; Jiang & Lu, 2018) found that redundant messages may impair the learning process empirically. In more detail, the model can converge slowly or sometimes lead to a worse sub-optimal. Agents then determine the priority of decision-making by *intentions* which is established and evaluated based on the world model.

World Model. The world model is needed to predict and evaluate future trajectories. SeqComm, unlike previous works (Kim et al., 2021; Du et al., 2021; Pretorius et al., 2021), can utilize received hidden states of other agents in the first round of communication to model more precise environment dynamics for the explicit coordination in the next round of communication. Once an agent can access other agents' hidden states, it shall have adequate information to estimate their actions since all agents are homogeneous and parameter-sharing. Therefore, the world model $\mathcal{M}(\cdot)$ takes as input the joint hidden states $\boldsymbol{h}_t = \{h_t^1, \dots, h_t^n\}$ and actions $\boldsymbol{a}_t$, and predicts the next joint observations and reward. In practice, before the inputs pass into the world model, the attention module $\mathrm{AM_w}$ is utilized to process the input.

$$\hat{\boldsymbol{o}}_{t+1}, \hat{r}_{t+1} = \mathcal{M}_i(\mathrm{AM_w}(\boldsymbol{h}_t, \boldsymbol{a}_t)).$$

The reason that we adopt the attention module is to entitle the world model to be generalizable in the scenarios where additional agents are introduced or existing agents are removed.

Priority of Decision-Making. The intention is the key element to determine the priority of decision-making. The notion of intention is described as an agent's future behavior in previous works (Rabinowitz et al., 2018; Raileanu et al., 2018; Kim et al., 2021). However, we define the *intention* as an agent's future behavior *without considering others.*

As mentioned before, an agent's intention considering others can lead to circular dependencies and cause miscoordination. By our definition, the intention of an agent should be depicted as all future trajectories considering that agent as the first mover and ignoring the others. However, there are many possible future trajectories as the priority of the rest agents is *unfixed.* In practice, we use the Monte Carlo method to estimate the intention value based on all future trajectories. Note that it is uniform across priorities for unfixed agents. Each order should be treated equally since we do not have any prior for the distribution.

Taking agent i at timestep t to illustrate, it firstly considers itself as the first-mover and produces its action only based on the joint hidden states, $\hat{a}_t^i \sim \pi_i(\cdot|\mathrm{AM_a}(\boldsymbol{h}_t, \emptyset)$, where we again use an attention module $\mathrm{AM_a}$ to handle the input. For the order sequence of lower-level agents, we randomly sample a set of order sequences from unfixed agents. Assume agent j is the second-mover, agent i models j's action by considering the upper-level action following its own policy $\hat{a}_t^j \sim \pi_i(\cdot|\mathrm{AM_a}(\boldsymbol{h}_t, \hat{a}_t^i))$. The same procedure is applied to predict the actions of all other agents following the sampled order sequence. Based on the joint hidden states and predicted actions, the next joint observations $\hat{\boldsymbol{o}}_{t+1}$ and corresponding reward $\hat{r}_{t+1}$ can be predicted by the world model. The length of the predicted future trajectory is H and it can then be written as $\tau^t = \{\hat{\boldsymbol{o}}_{t+1}, \hat{\boldsymbol{a}}_{t+1}, \dots, \hat{\boldsymbol{o}}_{t+H}, \hat{\boldsymbol{a}}_{t+H}\}$ by repeating the procedure aforementioned and the value of one trajectory is defined as the return of that trajectory $v_{\tau^t} = (\sum_{t'=t+1}^{t+H} \gamma^{t'-t-1} \hat{r}_{t'})/H$. In addition, the intention value is defined as the average value of F future trajectories with different sampled order sequences. The choice of F is a tradeoff between the computation overhead and the accuracy of the estimation.

After all the agents have computed their intentions and the corresponding value, they again communicate their intention values to others. Then agents would compare and choose the agent with the highest intention value to be the first mover. The priority of lower-level decision-making follows the same procedure with the upper-level agents fixed. Note that some agents are required to communicate intention values with others multiple times until the priority of decision-making is finally determined.

4.2 Launching Phase

As for the launching phase, agents communicate to obtain additional information to make decisions. Apart from the received hidden states from the last phase, we allow agents to get what *actual* actions the upper-level agents will take in execution, while other studies can only infer others' actions by opponent modeling (Rabinowitz et al., 2018; Raileanu et al., 2018) or communicating intentions (Kim et al., 2021). Therefore, miscoordination can be naturally avoided and a better cooperation strategy is possible since lower-level agents can adjust their behaviors accordingly. A lower-level agent i make a decision following the policy $\pi_i(\cdot|\mathrm{AM_a}(\boldsymbol{h}_t, \boldsymbol{a}_t^{upper}))$, where $\boldsymbol{a}_t^{upper}$ means received actual actions from all upper-level agents. As long as the agent has decided its action, it will send its action to all other lower-level agents by the communication channel. *Note that the actions are executed simultaneously and distributedly in execution, though agents make decisions sequentially.*

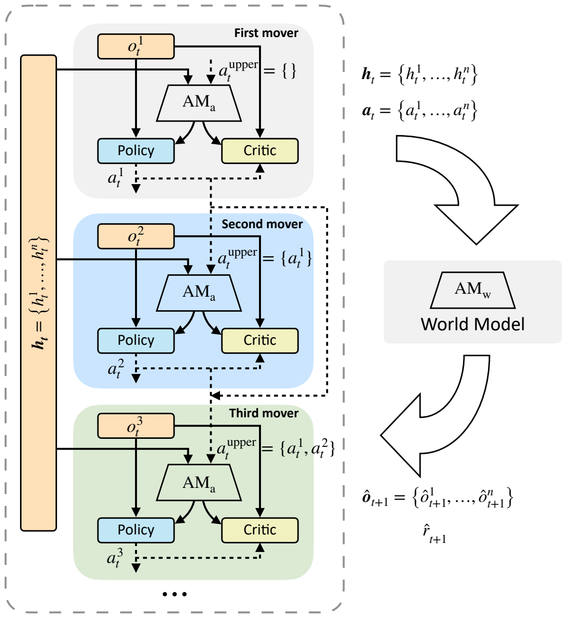

First mover
Second mover
Third mover
o_t^1
o_t^2
o_t^3
$a_t^{\text{upper}} = \{\}$
$a_t^{\text{upper}} = \{a_t^1\}$
$a_t^{\text{upper}} = \{a_t^1, a_t^2\}$
$\mathrm{AM_a}$
Policy
Critic
a_t^1
a_t^2
a_t^3
$\boldsymbol{h}_t = \{h_t^1, \dots, h_t^n\}$
$\boldsymbol{a}_t = \{a_t^1, \dots, a_t^n\}$
$\mathrm{AM_w}$
World Model
$\hat{\boldsymbol{o}}_{t+1} = \{\hat{o}_{t+1}^1, \dots, \hat{o}_{t+1}^n\}$
$\hat{r}_{t+1}$

Figure 3: Architecture of SeqComm. The critic and policy of each agent take input as its own observation and received messages. The world model takes as input the joint hidden states and predicted joint actions.

4.3 Local Communication

SeqComm is more suitable for latency-tolerate MARL tasks, *e.g.,* power dispatch (minutes) (Wang et al., 2021a), inventory management (hours) (Feng et al., 2021), maritime transportation (days) (Li et al., 2019)). However, in some real applications, broadcast communication can be unrealistic and incur lots of communication overhead. Therefore, we provide another version of SeqComm in scenarios where agents can only communicate with nearby agents (agents within a limited communication range).

In more detail, the agent first calculates its own intention value based only on the hidden states of nearby agents. After comparing with the intention values of nearby agents, the agent can determine the upper-level and lower-level nearby agents. Once the agent receives the actual actions of all the upper-level nearby agents, it decides on its action and communicates it with the lower-level nearby agents. Unlike the previous version of SeqComm, agents cannot distinguish the detailed order sequence of the upper-level nearby agents since their communication ranges may not overlap. Therefore, the intention values are calculated and communicated among agents for *only one time.* The local communication version greatly reduces communication overhead, making it more suitable for many real applications.

For more details of the algorithms, please refer to the Appendix F for the pseudo-code.

4.4 Theoretical Analysis

As intention values determine the priority of decision-making, SeqComm is likely to choose different orders *at different timesteps* during training. However, we have the following proposition that theoretically guarantees the performance of the learned joint policy under SeqComm.

Proposition 2. *The monotonic improvement and convergence of the joint policy in SeqComm are independent of the priority of decision-making of agents at each timestep.*

Proof. The proof is given in Appendix B. □

The priority of decision-making is chosen under the world model, thus the compounding errors in the world model can result in discrepancies between the predicted returns of the same order under the world model and the true dynamics. We then analyze the monotonic improvement for the joint policy under the world model based on Janner et al. (2019).

Theorem 1. *Let the expected total variation between two transition distributions be bounded at each timestep as* $\max_t \mathbb{E}_{s\sim\boldsymbol{\pi}_{\beta,t}}[D_{TV}(p(s'|s,\boldsymbol{a})||\hat{p}(s'|s,\boldsymbol{a}))] \le \epsilon_m$*, and the policy divergences at level* k *be bounded as* $\max_{s,\boldsymbol{a}^{1:k-1}} D_{TV}(\pi_{\beta,k}(a^k|s,\boldsymbol{a}^{1:k-1})||\pi_k(a^k|s,\boldsymbol{a}^{1:k-1})) \le \epsilon_{\pi_k}$*, where* $\boldsymbol{\pi}_\beta$ *is the data collecting policy for the model and* $\hat{p}(s'|s,\boldsymbol{a})$ *is the transition distribution under the model. Then the model return* $\hat{\eta}$ *and true return* η *of the policy* $\boldsymbol{\pi}$ *are bounded as:*

$$\hat{\eta}[\boldsymbol{\pi}] \ge \eta[\boldsymbol{\pi}] - \underbrace{[\frac{2\gamma r_{\max}(\epsilon_m + 2\sum_{k=1}^{n}\epsilon_{\pi_k})}{(1-\gamma)^2} + \frac{4r_{\max}\sum_{k=1}^{n}\epsilon_{\pi_k}}{(1-\gamma)}]}_{C(\epsilon_m,\boldsymbol{\epsilon}_{\pi_{1:n}})}.$$

Proof. The proof is given in Appendix C. □

Remark 2. *Theorem 1 provides a useful relationship between the compounding errors and the policy update. As long as we improve the return under the true dynamic by more than the gap,* $C(\epsilon_m, \boldsymbol{\epsilon}_{\pi_{1:n}})$*, we can guarantee the policy improvement under the world model. If no such policy exists to overcome the gap, it implies the model error is too high, that is, there is a large discrepancy between the world model and true dynamics. Thus the order sequence obtained under the world model is not reliable. Such an order sequence is almost the same as a random one. Though a random order sequence also has the theoretical guarantee of Proposition 2, we will show in Section 5.2 that a random order sequence leads to a poor local optimum empirically.*

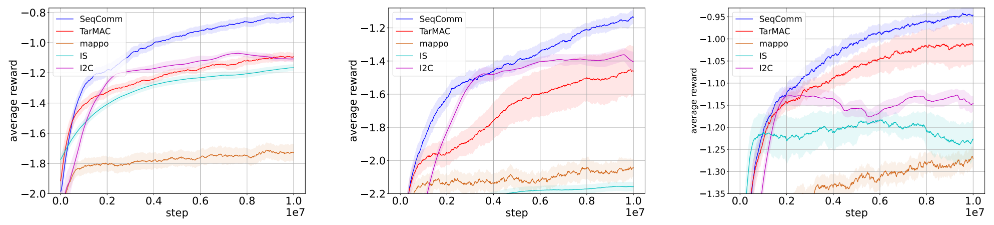
5 Experiments

SeqComm is currently instantiated based on MAPPO (Yu et al., 2021). We evaluate SeqComm on three tasks in multi-agent particle environment (MPE) (Lowe et al., 2017) and some maps in StarCraft multi-agent challenge (SMAC) (Samvelyan et al., 2019).

For these experiments, we compare SeqComm (broadcast communication) against the following communication-free and communication-based baselines:

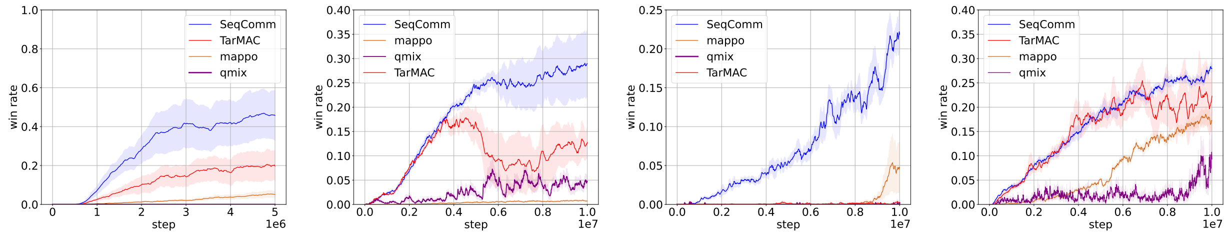
- MAPPO (Yu et al., 2021), a multi-agent version of PPO Schulman et al. (2017), achieving good performance in cooperative multi-agent tasks.
- QMIX (Rashid et al., 2018), a popular and high-performing CTDE method based on monotonic value decomposition.
- IS (Kim et al., 2021), where agents communicate predicted future trajectories (observations and actions) and predictions are made by the environment model.
- TarMAC (Das et al., 2019), where agents use the attention module to focus more on important incoming messages (the encoding of observation). In addition, TarMAC also adopts broadcast communication.
- I2C (Ding et al., 2020), where agents infer one-to-one communication to reduce the redundancy of messages (also conditioned on observations).

In the experiments, SeqComm and baselines are parameter-sharing for fast convergence (Gupta et al., 2017; Terry et al., 2020). We have fine-tuned the baselines for a fair comparison. The world model in MPE is pre-trained with data generated from any pre-trained algorithm for convenience. However, in the more complex and high-dimensional SMAC environment, the world model is trained from scratch and kept fine-tuned in the learning process. Please refer to the Appendix F for the hyperparameter settings. All results are presented in terms of the mean and standard deviation of five runs with different random seeds.



(a) PP (b) CN (c) KA

Figure 4: Learning curves of SeqComm and baselines in PP, CN, and KA. We run 1e^7 steps in all three MPE tasks.



(a) 6h_vs_8z (b) 8m_vs_9m (c) MMM2 (d) 10m_vs_11m

Figure 5: Learning curves of SeqComm and baselines in four SMAC customized maps.

5.1 Results

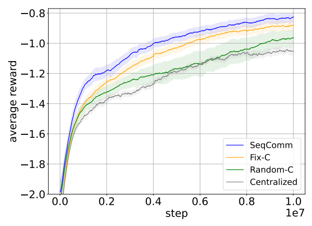
MPE. We experiment on predator-prey (PP), cooperative navigation (CN), and keep-away (KA) in MPE. In PP, five predators (agents) try to capture three prey. In CN, five agents try to occupy five landmarks. In KA, three attackers (agents) try to occupy three landmarks, however, there are three defenders to push them away.

In all three tasks, the size of agents is set to be larger than the original settings so that collisions occur more easily, following the settings in previous work (Kim et al., 2021). In addition, agents cannot observe any other agents, and this makes the task more difficult and communication more important. We can observe similar modifications in previous work (Foerster et al., 2016; Ding et al., 2020). After all, we want to demonstrate the superiority over communication-based baselines, and communication-based methods are more suitable for scenarios with limited vision. More details about experimental settings are available in Appendix E.

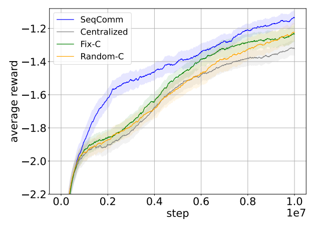
Performance. Figure 4 shows the learning curves of all the methods in terms of the mean reward averaged over timesteps in PP, CN, and KA. We can see that SeqComm converges to the highest mean reward compared with all the baselines. In more detail, all communication-based methods outperform MAPPO, indicating the necessity of communication in these difficult tasks. Apart from MAPPO, IS performs the worst since it may access inaccurate predicted information due to the circular dependencies. The substantial improvement SeqComm over I2C and TarMAC may be attributed to that SeqComm allows agents to get more valuable action information for explicit coordination. Note that QMIX is omitted in the comparison for clear presentation since previous work (Yu et al., 2021) have shown QMIX and MAPPO exhibit similar performance in various MPE tasks.

The agents learned by SeqComm show sophisticated coordination strategies induced by the priority of decision-making, which can be witnessed by the visualization of agent behaviors. More details are given in Appendix D.

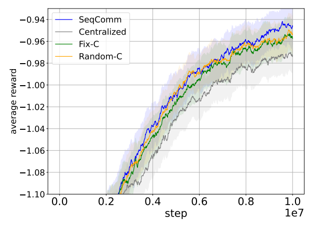
SMAC. We also evaluate SeqComm against the baselines on four customized maps in SMAC: 6h_vs_8z, MMM2, 10m_vs_11m, and 8m_vs_9m. We have made some minor changes to the observation part of agents to make it more difficult since MAPPO has already achieved impressive results on default settings. Specifically, the sight range of agents is reduced from 9 to 2, and agents cannot perceive any information about their allies even if they are within the sight range. NDQ (Wang et al., 2020) adopts a similar change on its



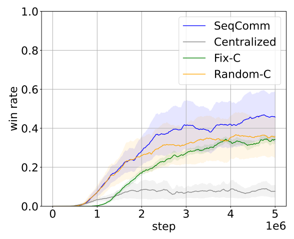
(a) PP



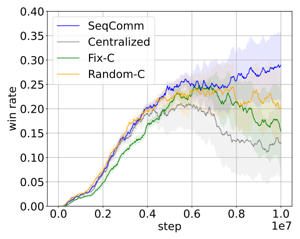
(b) CN



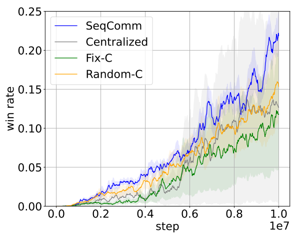
(c) KA



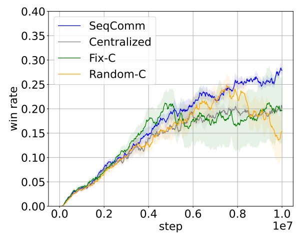
(d) 6h_vs_8z



(e) 8m_vs_9m



(f) MMM2



(g) 10m_vs_11m

Figure 6: Ablation studies in all tasks. Fix-C: the priority of decision-making is fixed throughout one episode. Random-C: the priority of decision-making is determined randomly. Centralized: agents get joint observations during training and execution.

customized maps to increase the difficulty of action coordination and demonstrates that the miscoordination problem is widespread in multi-agent learning. The rest settings remain the same as the default. Besides, the value function in QMIX also takes input as the observations of all agents for a fair comparison.

Performance. The learning curves of SeqComm and the baselines in terms of the win rate are illustrated in Figure 5. IS and I2C fail in these tasks and get a zero win rate because these two methods are built on MADDPG. However, MADDPG cannot work well in SMAC, especially when we reduce the sight range of agents, which is also supported by other studies (Papoudakis et al., 2021). SeqComm and TarMAC converge to better performances than MAPPO and QMIX, which demonstrates the benefit of communication. Moreover, SeqComm still outperforms TarMAC, consistent with the previous results.

5.2 Ablation Studies

Priority of Decision-Making. We compare SeqComm with three ablation baselines: the priority of decision-making is fixed throughout one episode, denoted as *Fix-C*, the priority of decision-making is determined randomly at each timestep, denoted as *Random-C*, and agents access the joint observations during training and execution, denoted as *Centralized*. Note that *Centralized* does not have the agent priority.

As depicted in Figure 6, SeqComm achieves a higher mean reward or win rate than Fix-C, Random-C, and Centralized in almost all the tasks. These results verify the importance of the priority of decision-making and the necessity to continuously adjust it during one episode. It is also demonstrated that SeqComm can provide an improved priority of decision-making.

As discussed in Section 4.4, although Fix-C and Random-C also have the theoretical guarantee, they converge to poor local optima in practice. Moreover, Fix-C and Random-C show better performance than Centralized in most tasks. This result accords with the hypothesis that the SE is likely to be Pareto superior to the average NE in games that require a high cooperation level.

Additionally, the learned policy of SeqComm can generalize well to the same task with a different number of agents in MPE, which is detailed in Appendix D.

Local Communication. In this part, we evaluate the local communication version of SeqComm in SMAC. Agents can only communicate with nearby agents (agents within their communication range). The number of nearby agents is restricted from 4 to 5. We compare SeqComm with only two ablation baselines, Random-C and Centralized, since Fix-C is unavailable in local communication settings.

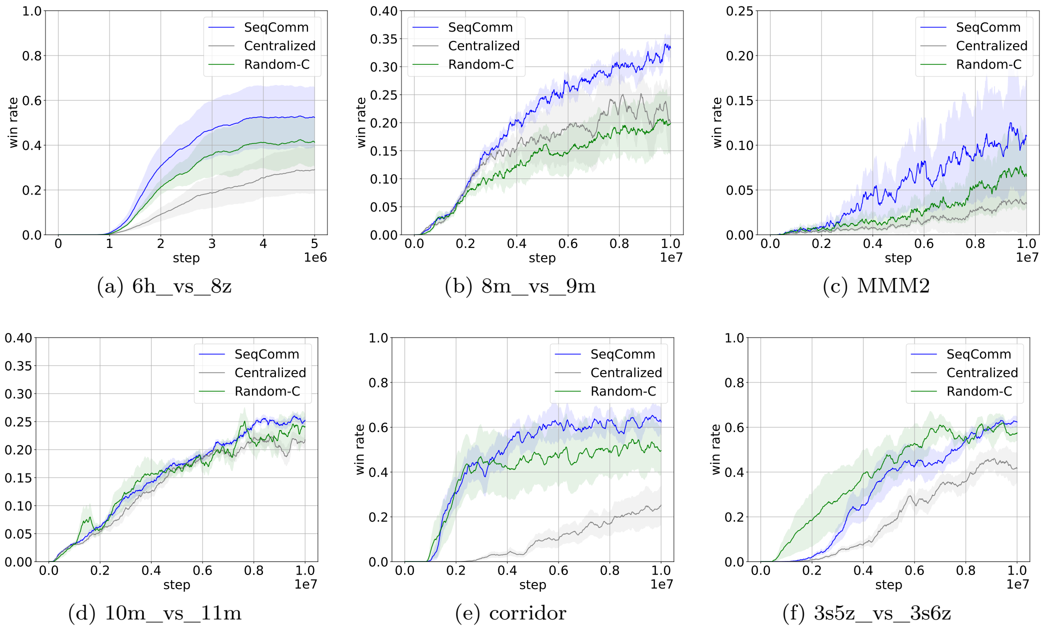


Figure 7: Learning curves of SeqComm in four SMAC customized maps under local communication settings.

In Figure 7, we can find out that SeqComm gets the highest win rate in all the maps. This reveals that SeqComm adapts effectively to local communication settings and communicating actual actions in a determined order still helps even in a reduced communication range.

The results show the potential of SeqComm for real applications. For some maps, it is possible that the local communication version outperforms the original version since the broadcast communication may incur some redundant information which may impair the learning process (Jiang & Lu, 2018).

We also carry out ablation studies on communication range in MPE tasks. Note that communication range means how many nearest neighbors each agent is allowed to communicate with, following the settings in previous work (Ding et al., 2020). We reduce the communication range of SeqComm from 4 to 2 and 0. As there are only three agents on a team in KA, it is omitted in this study. The results are shown in Figure 8. Communication-based agents (SeqComm with range 2 and range 4) perform better than communication-free agents (SeqComm with range 0), which accords with the results of many previous studies. More importantly, the better performance of SeqComm with a communication range 2 than the corresponding TarMAC again demonstrates the effectiveness of our method even in reduced communication ranges.

However, as the communication range decreases from 4 to 2, there is no performance reduction in these two MPE tasks. On the contrary, the agents with communication range 2 perform the best. It accords with the results in I2C (Ding et al., 2020) and ATOC (Jiang & Lu, 2018) that redundant information can impair the learning process sometimes. In other settings, this conclusion might not be true. Moreover, since under our communication scheme agents can obtain more in-

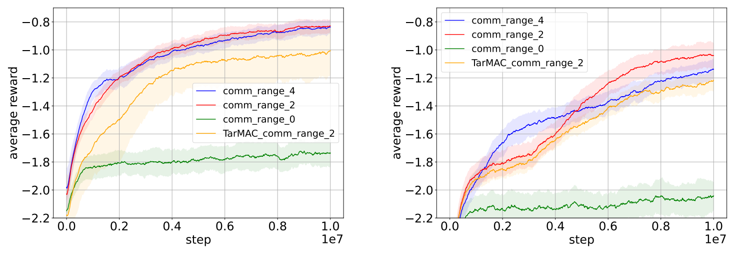


Figure 8: Ablation studies under local communication in PP (left) and CN (right).

formation, *i.e.* the actual actions of others, it is more reasonable that SeqComm can still outperform other methods in reduced communication ranges.

6 Conclusions

We have proposed SeqComm that enables agents well and explicitly coordinate with each other. SeqComm from an asynchronous perspective allows agents to make decisions sequentially. A two-phase communication scheme has been adopted for determining the priority of decision-making and transferring messages accordingly. Empirically, it is demonstrated that SeqComm outperforms baselines in a variety of cooperative multi-agent scenarios.

References

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning (ICML)*, 2020.

Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38 (2):156–172, 2008.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning (ICML)*, 2019.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yali Du, Yifan Zhao, Meng Fang, Jun Wang, Gangyan Xu, and Haifeng Zhang. Learning predictive communication by imagination in networked system control, 2021.

Mingxiao Feng, Guozi Liu, Li Zhao, Lei Song, Jiang Bian, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Multi-agent reinforcement learning with shared resource in inventory management. 2021.

Felix Fischer, Michael Rovatsos, and Gerhard Weiss. Hierarchical reinforcement learning in communication-mediated multiagent coordination. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Amy Greenwald, Keith Hall, and Roberto Serrano. Correlated q-learning. In *A comprehensive survey of multiagent reinforcement learning*, 2003.

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representation (ICLR)*, 2020.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations (ICLR)*, 2021.

Sachin Konan, Esmaeil Seraj, and Matthew Gombolay. Iterated reasoning with mutual information in cooperative and byzantine decentralized teaming. In *International Conference on Learning Representations (ICLR)*, 2022.

Ville Könönen. Asymmetric multiagent reinforcement learning. *Web Intelligence and Agent Systems: An international journal*, 2(2):105–121, 2004.

Xihan Li, Jia Zhang, Jiang Bian, Yunhai Tong, and Tie-Yan Liu. A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. *arXiv preprint arXiv:1903.00714*, 2019.

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 (NeurIPS)*, 2017.

Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

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

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks, 2021.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

MV Nagendra Prasad, Victor R Lesser, and Susan E Lander. Learning organizational roles for negotiated search in a multiagent system. *International Journal of Human-Computer Studies*, 48(1):51–67, 1998.

Arnu Pretorius, Scott Cameron, Andries Petrus Smit, Elan van Biljon, Lawrence Francis, Femi Azeez, Alexandre Laterre, and Karim Beguir. Learning to communicate through imagination with model-based deep multi-agent reinforcement learning, 2021.

David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *J. Artif. Intell. Res.*, 16:389–423, 2002.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International Conference on Machine Learning (ICML)*, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In *International Conference on Learning Representations (ICLR)*, 2019.

Eric Sodomka, Elizabeth Hilliard, Michael Littman, and Amy Greenwald. Coco-q: Learning in stochastic games with side payments. In *International Conference on Machine Learning (ICML)*, 2013.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Justin K. Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, Benjamin Black, and Dinesh Manocha. Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

Jur P Van Den Berg and Mark H Overmars. Prioritized motion planning for multiple robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

Nikos Vlassis. A concise introduction to multiagent systems and distributed artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 1(1):1–71, 2007.

Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representation (ICLR)*, 2020.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. *arXiv preprint arXiv:2106.02886*, 2021b.

Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning. In *AAAI Spring Symposium Series*, 2018.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and W Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Haifeng Zhang, Weizhe Chen, Zeren Huang, Minne Li, Yaodong Yang, Weinan Zhang, and Jun Wang. Bi-level actor-critic for multi-agent coordination. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A Appendix

B Proofs of Proposition 1 and Proposition 2

Lemma 1 (Agent-by-Agent PPO). *If we update the policy of each agent i with TRPO Schulman et al. (2015) (or approximately PPO) when fixing all the other agent's policies, then the joint policy will improve monotonically.*

Proof. We consider the joint surrogate objective in TRPO $L_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}})$ where $\boldsymbol{\pi}_{\text{old}}$ is the joint policy before updating and $\boldsymbol{\pi}_{\text{new}}$ is the joint policy after updating.

Given that $\pi_{\text{new}}^{-i} = \pi_{\text{old}}^{-i}$, we have:

$$\begin{aligned}
L_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}) &= \mathbb{E}_{a\sim\boldsymbol{\pi}_{\text{new}}}[A_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a})] \\
&= \mathbb{E}_{a\sim\boldsymbol{\pi}_{\text{old}}}[\frac{\boldsymbol{\pi}_{\text{new}}(\boldsymbol{a}|s)}{\boldsymbol{\pi}_{\text{old}}(\boldsymbol{a}|s)} A_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a})] \\
&= \mathbb{E}_{a\sim\boldsymbol{\pi}_{\text{old}}}[\frac{\pi^i_{\text{new}}(a^i|s)}{\pi^i_{\text{old}}(a^i|s)} A_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a})] \\
&= \mathbb{E}_{a^i\sim\pi^i_{\text{old}}}\left[\frac{\pi^i_{\text{new}}(a^i|s)}{\pi^i_{\text{old}}(a^i|s)} \mathbb{E}_{a^{-i}\sim\pi^{-i}_{old}}[A_{\boldsymbol{\pi}_{\text{old}}}(s,a^i,a^{-i})]\right] \\
&= \mathbb{E}_{a^i\sim\pi^i_{\text{old}}}\left[\frac{\pi^i_{\text{new}}(a^i|s)}{\pi^i_{\text{old}}(a^i|s)} A^i_{\boldsymbol{\pi}_{\text{old}}}(s,a^i)\right] \\
&= L_{\pi^i_{\text{old}}}(\pi^i_{\text{new}}),
\end{aligned}$$

where $A^i_{\boldsymbol{\pi}_{\text{old}}}(s,a^i) = \mathbb{E}_{a^{-i}\sim\pi^{-i}_{\text{old}}}[A_{\boldsymbol{\pi}_{\text{old}}}(s,a^i,a^{-i})]$ is the individual advantage of agent i, and the third equation is from the condition $\pi_{\text{new}}^{-i} = \pi_{\text{old}}^{-i}$.

With the result of TRPO, we have the following conclusion:

$$\begin{aligned}
J(\pi_{\text{new}}) - J(\pi_{\text{old}}) &\geq L_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}) - \text{C}D^{\max}_{\text{KL}}(\boldsymbol{\pi}_{\text{new}}||\boldsymbol{\pi}_{\text{old}}) \\
&= L_{\pi^i_{\text{old}}}(\pi^i_{\text{new}}) - \text{C}D^{\max}_{\text{KL}}(\pi^i_{\text{new}}||\pi^i_{\text{old}}) \quad (\text{from } \pi_{\text{new}}^{-i} = \pi_{\text{old}}^{-i})
\end{aligned}$$

This means the individual objective is the same as the joint objective so the monotonic improvement is guaranteed. □

Then we can show the proof of Proposition 1.

Proof. We will build a new MDP $\tilde{M}$ based on the original MDP. We keep the action space $\tilde{A} = A = \times_{i=1}^n A^i$, where A^i is the original action space of agent i. The new state space contains multiple layers. We define $\tilde{S}^k = S \times (\times_{i=1}^k A^i)$ for $k = 1, 2, \cdots, n-1$ and $\tilde{S}^0 = S$, where S is the original state space. Then a new state $\tilde{s}^k \in \tilde{S}^k$ means that $\tilde{s}^k = (s, a^1, a^2, \cdots, a^k)$. The total new state space is defined as $\tilde{S} = \cup_{i=0}^{n-1}\tilde{S}^i$. Next we define the transition probability $\tilde{P}$ as following:

$$\begin{aligned}
\tilde{P}(\tilde{s}'|\tilde{s}^k, a^{k+1}, a^{-(k+1)}) &= \mathbb{1}\left(\tilde{s}' = (\tilde{s}^k, a^{k+1})\right), \ k < n-1 \\
\tilde{P}(\tilde{s}'|\tilde{s}^k, a^{k+1}, a^{-(k+1)}) &= \mathbb{1}\left(\tilde{s}' \in \tilde{S}^0\right) P(\tilde{s}'|\tilde{s}^k, a^{k+1}), \ k = n-1.
\end{aligned}$$

In another word, the new state space is composed by $N+1$ layers where the state in the layer k is composed by the original state and the actions of the first k agents. The new transition probability means that the state in the layer k can only transition to the state in the layer $k+1$ with the corresponding action, and the state in the layer $n-1$ will transition to the layer 0 with the probability P in the original MDP. The reward function $\tilde{r}$ is defined as following:

$$\tilde{r}(\tilde{s},\boldsymbol{a}) = \mathbb{1}\left(\tilde{s} \in \tilde{S}_0\right) r(\tilde{s},\boldsymbol{a}).$$

This means the reward is only obtained when the state in layer 0 and the value is the same as the original reward function. Now we obtain the total definition of the new MDP $\tilde{M} = \{\tilde{S}, \tilde{A}, \tilde{P}, \tilde{r}, \gamma\}$.

Then we claim that if all agents learn in multi-agent sequential decision-making by PPO, they are actually taking agent-by-agent PPO in the new MDP $\tilde{M}$. To be precise, one update of multi-agent sequential decision-making in the original MDP M equals to a round of update from agent 1 to agent n by agent-by-agent PPO in the new MDP $\tilde{M}$. Moreover, the total reward of a round in the new MDP $\tilde{M}$ is the same as the reward in one timestep in the original MDP M. With this conclusion and Lemma 1, we complete the proof.

□

The proof of Proposition 2 can be seen as a corollary of the proof of Proposition 1.

Proof. From Lemma 1 we know that the monotonic improvement of the joint policy in the new MDP $\tilde{M}$ is guaranteed for each update of one single agent's policy. So even if the different round of updates in the new MDP $\tilde{M}$ is with different order of the decision-making, the monotonic improvement of the joint policy is still guaranteed. Finally, from the proof of Proposition 1, we know that the monotonic improvement in the new MDP $\tilde{M}$ equals to the monotonic improvement in the original MDP M. These complete the proof. □

C Proofs of Theorem 1

Lemma 2 (TVD of the joint distributions). *Suppose we have two distribution $p_1(x, y) = p_1(x)p_1(x|y)$ and $p_2(x, y) = p_2(x)p_2(x|y)$. We can bound the total variation distance of the joint as:*

$$D_{TV}(p_1(x, y)||p_2(x, y)) \leq D_{TV}(p_1(x)||p_2(x)) + \max_x D_{TV}(p_1(y|x)||p_2(y|x))$$

Proof. See (Janner et al., 2019) (Lemma B.1). □

Lemma 3 (Markov chain TVD bound, time-varing). *Suppose the expected KL-divergence between two transition is bounded as $\max_t \mathbb{E}_{s\sim p_{1,t}(s)} D_{KL}(p_1(s'|s)||p_2(s'|s)) \leq \delta$, and the initial state distributions are the same $p_{1,t=0}(s) = p_{2,t=0}(s)$. Then the distance in the state marginal is bounded as:*

$$D_{TV}(p_{1,t}(s)||p_{2,t}(s)) \leq t\delta$$

Proof. See (Janner et al., 2019) (Lemma B.2). □

Lemma 4 (Branched Returns Bound). *Suppose the expected KL-divergence between two dynamics distributions is bounded as $\max_t \mathbb{E}_{s\sim p_{1,t}(s)}[D_{TV}(p_1(s'|s, \boldsymbol{a})||p_2(s'|s, \boldsymbol{a}))]$, and the policy divergences at level k are bounded as $\max_{s,\boldsymbol{a}^{1:k-1}} D_{TV}(\pi_1(a^k|s, \boldsymbol{a}^{1:k-1})||\pi_2(a^k|s, \boldsymbol{a}^{1:k-1})) \leq \epsilon_{\pi_k}$. Then the returns are bounded as:*

$$|\eta_1 - \eta_2| \leq \frac{2r_{\max}\gamma(\epsilon_m + \sum_{k=1}^n \epsilon_{\pi_k})}{(1-\gamma)^2} + \frac{2r_{\max}\sum_{k=1}^n \epsilon_{\pi_k}}{1-\gamma},$$

where $r_{\max}$ is the upper bound of the reward function.

Proof. Here, η_1 denotes the returns of $\boldsymbol{\pi}_1$ under dynamics $p_1(s'|s, \boldsymbol{a})$, and η_2 denotes the returns of $\boldsymbol{\pi}_2$ under dynamics $p_2(s'|s, \boldsymbol{a})$. Then we have

$$\begin{aligned}
|\eta_1 - \eta_2| &= |\sum_{s,\boldsymbol{a}}(p_1(s, \boldsymbol{a}) - p_2(s, \boldsymbol{a}))r(s, \boldsymbol{a})| \\
&= |\sum_t \sum_{s,\boldsymbol{a}} \gamma^t(p_{1,t}(s, \boldsymbol{a}) - p_{2,t}(s, \boldsymbol{a}))r(s, \boldsymbol{a})| \\
&\leq \sum_t \sum_{s,\boldsymbol{a}} \gamma^t|p_{1,t}(s, \boldsymbol{a}) - p_{2,t}(s, \boldsymbol{a})|r(s, \boldsymbol{a}) \\
&\leq r_{\max}\sum_t \sum_{s,\boldsymbol{a}} \gamma^t|p_{1,t}(s, \boldsymbol{a}) - p_{2,t}(s, \boldsymbol{a})|.
\end{aligned}$$

By Lemma 2, we get

$$\begin{aligned}
\max_s D_{TV}(\pi_1(\boldsymbol{a}|s)||\pi_2(\boldsymbol{a}|s)) &\leq \max_{s,a_1} D_{TV}(\pi_1(\boldsymbol{a}^{-1}|s,a^1)||\pi_2(\boldsymbol{a}^{-1}|s,a^1)) \\
&+ \max_s D_{TV}(\pi_1(a^1|s)||\pi_2(a^1|s)) \\
&\leq \cdots \\
&\leq \sum_{k=1}^{n} \max_{s,\boldsymbol{a}^{1:k-1}} D_{TV}(\pi_1(a^k|s,\boldsymbol{a}^{1:k-1})||\pi_2(a^k|s,\boldsymbol{a}^{1:k-1})) \\
&\leq \sum_{k=1}^{n} \epsilon_{\pi_k}.
\end{aligned}$$

We then apply Lemma 3, using $\delta = \epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k}$ (via Lemma 3 and 2) to get

$$\begin{aligned}
D_{TV}(p_{1,t}(s)||p_{2,t}(s)) &\leq t \max_t E_{s\sim p_{1,t}(s)} D_{TV}(p_{1,t}(s'|s)||p_{2,t}(s'|s)) \\
&\leq t \max_t E_{s\sim p_{1,t}(s)} D_{TV}(p_{1,t}(s',\boldsymbol{a}|s)||p_{2,t}(s',\boldsymbol{a}|s)) \\
&\leq t(\max_t E_{s\sim p_{1,t}(s)} D_{TV}(p_{1,t}(s'|s,\boldsymbol{a})||p_{2,t}(s'|s,\boldsymbol{a})) \\
&+ \max_t E_{s\sim p_{1,t}(s)} \max_s D_{TV}(\boldsymbol{\pi}_{1,t}(\boldsymbol{a}|s)||\boldsymbol{\pi}_{2,t}(\boldsymbol{a}|s))) \\
&\leq t(\epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k})
\end{aligned}$$

And we also get $D_{TV}(p_{1,t}(s,\boldsymbol{a})||p_{2,t}(s,\boldsymbol{a})) \leq t(\epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k}) + \sum_{k=1}^{n} \epsilon_{\pi_k}$ by Lemma 2. Thus, by plugging this back, we get:

$$\begin{aligned}
|\eta_1 - \eta_2| &\leq r_{\max} \sum_t \sum_{s,\boldsymbol{a}} \gamma^t |p_{1,t}(s,\boldsymbol{a}) - p_{2,t}(s,\boldsymbol{a})| \\
&\leq 2r_{\max} \sum_t \gamma^t (t(\epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k}) + \sum_{k=1}^{n} \epsilon_{\pi_k}) \\
&\leq 2r_{\max}(\frac{\gamma(\epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k}))}{(1-\gamma)^2} + \frac{\sum_{k=1}^{n} \epsilon_{\pi_k}}{1-\gamma})
\end{aligned}$$

□

Then we can show the proof of Theorem 1.

Proof. Let $\boldsymbol{\pi}_\beta$ denote the data collecting policy. We use Lemma 4 to bound the returns, but it will require bounded model error under the new policy $\boldsymbol{\pi}$. Thus, we need to introduce $\boldsymbol{\pi}_\beta$ by adding and subtracting $\eta[\boldsymbol{\pi}_\beta]$, to get:

$$\hat{\eta}[\boldsymbol{\pi}] - \eta[\boldsymbol{\pi}] = \hat{\eta}[\boldsymbol{\pi}] - \eta[\boldsymbol{\pi}_\beta] + \eta[\boldsymbol{\pi}_\beta] - \eta[\boldsymbol{\pi}].$$

we can bound L_1 and L_2 both using Lemma 4 by using $\delta = \sum_{k=1}^{n} \epsilon_{\pi_k}$ and $\delta = \epsilon_m + \sum_{k=1}^{n} \epsilon_{\pi_k}$ respectively, and obtain:

$$L_1 \geq -\frac{2\gamma r_{\max} \sum_{k=1}^{n} \epsilon_{\pi_k}}{(1-\gamma)^2} - \frac{2r_{\max} \sum_{k=1}^{n} \epsilon_{\pi_k}}{(1-\gamma)}$$

$$L_2 \geq -\frac{2\gamma r_{\max}(\epsilon_{\pi_m} + \sum_{k=1}^{n} \epsilon_{\pi_k})}{(1-\gamma)^2} - \frac{2r_{\max} \sum_{k=1}^{n} \epsilon_{\pi_k}}{(1-\gamma)}.$$

Adding these two bounds together yields the conclusion. □

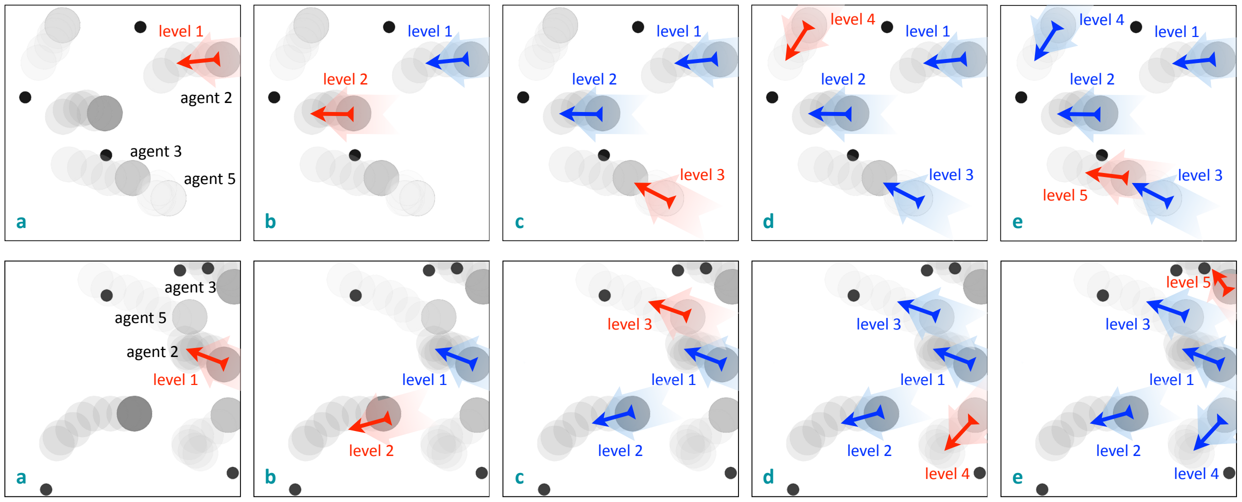


Figure 9: Illustration of learned priority of decision making in PP (*upper panel*) and CN (*lower panel*). Preys (landmarks) are viewed in black and predators (agents) are viewed in grey in PP (CN). From *a* to *e*, shown is the priority order. The smaller the level index, the higher priority of decision-making is.

Table 1: Mean reward in different tasks, averaged over timesteps, with 200 test trials.

	Fix-C	SeqComm
3-agent in CN	-0.83 ± 0.17	$\mathbf{-0.76} \pm 0.08$
7-agent in CN	-1.79 ± 0.15	$\mathbf{-1.57} \pm 0.10$
7-agent in PP	-1.89 ± 0.45	$\mathbf{-1.31} \pm 0.60$

D Additional Experiments

D.1 Illustration of Learned Priority of Decision-Making

Figure 9 (upper panel from *a* to *e*) shows the priority order of decision-making determined by SeqComm in PP. Agent 2 that is far away from other preys and predators is chosen to be the first-mover. If agents want to encircle and capture the preys, the agents (*e.g.*, agent 2 and 5) that are on the periphery of the encircling circle should hold upper-level positions since they are able to decide how to narrow the encirclement. In addition, agent 3 makes decisions prior to agent 5 so that collision can be avoided after agent 5 obtains the intention of agent 3.

For CN, as illustrated in Figure 9 (lower panel from *a* to *e*), agent 2 is far away from all the landmarks and all other agents are in a better position to occupy landmarks. Therefore, agents 2 is chosen to be the first-mover, which is similar to the phenomenon observed in PP. Once it has determined the target to occupy, other agents (agent 5 and 3) can adjust their actions accordingly and avoid conflict of goals. Otherwise, if agent 5 makes a decision first and chooses to occupy the closest landmark, then agent 2 has to approach to a further landmark which would take more steps.

D.2 Generalization

Generalization to different numbers of agents has always been a key problem in MARL. For most algorithms in communication, once the model is trained in one scenario, it is unlikely for agents to maintain relatively competitive performance in other scenarios with different numbers of agents. However, as we employ attention modules to process communicated messages so that agents can handle messages of different lengths. In addition, the module used to determine the priority of decision-making is also not restricted by the number of agents. Thus, we investigate whether SeqComm generalizes well to different numbers of agents in CN and PP.

For both tasks, SeqComm is trained on 5-agent settings. Then, we test SeqComm in 3-agent and 7-agent settings of CN and 7-agent setting of PP. We use Fix-C trained *directly* on these test tasks to illustrate the performance of SeqComm. Note that the quantity of both landmarks and preys is adjusted according to the number of agents in CN and PP. The test results are shown in Table 1. SeqComm exhibits the superiority in CN and PP, demonstrating that SeqComm may have a good generalization to the number of agents. A thorough study of the generalization of SeqComm is left to future work.

E Experimental Settings

In cooperative navigation, there are 5 agents and the size of each is 0.15. They need to occupy 5 landmarks with a size of 0.05. The acceleration of agents is 7. In predator-prey, the number of predators (agents) and prey is set to 5 and 3, respectively, and their sizes are 0.15 and 0.05. The acceleration is 5 for predators and 7 for prey. In keep away, the number of attackers (agents) and defenders is set to 3, and their sizes are respectively 0.15 and 0.05. Besides, the acceleration is 6 for attackers and 4 for defenders. The three landmarks are located at $(0.00, 0.30)$, $(0.25, -0.15)$, and $(-0.25, -0.15)$. Note that each agent is allowed to communicate with all other agents in all three tasks. The team reward is similar across tasks. At a timestep t, it can be written as $r^t_{\text{team}} = -\sum_{i=1}^{n} d^t_i + C^t r_{\text{collision}}$, where d^t_i is the distance of landmark/prey i to its nearest agent/predator, C^t is the number of collisions (when the distance between two agents is less than the sum of their sizes) occurred at timestep t, and $r_{\text{collision}} = -1$. In addition, agents act discretely and have 5 actions (stay and move up, down, left, right). The length of each episode is 20, 30, and 20 in cooperative navigation, predator-prey, and keep-away, respectively.

F Implementation Details

F.1 Algorithm

In this part, we provide the pseudo-code of SeqComm as below:

Algorithm 1 Negotiation Phase

```
Require: Number of agents $N$
  $\mathcal{P} = [\ ]$: already determined priority
  $\mathcal{A} = \{1, 2, ..., N\}$: remaining agents
  /* Agents communicate the hidden state $\boldsymbol{h}$ of their observations with each other*/
  for $i = 1, 2, ..., N$ do
    for $j$ in $\mathcal{A}$ do
      Compute agent $j$'s intention value $v_j$ via Algorithm 2
    end for
    /* Agents in $\mathcal{A}$ communicate the intention values with each other*/
    Set $p_i$ to be the agent $j$ with the maximum $v_j$
    Append $p_i$ to $\mathcal{P}$ and remove it from $\mathcal{A}$
  end for
```

Algorithm 2 Intention Value Calculation of Agent a

```
Require: Already determined priority $\mathcal{P}$, remaining agents $\mathcal{A}$, number of sampling trajectories $F$, length of
  predicted future trajectory $H$, policy $\pi$ and attention module $\text{AM}_\text{a}$, world model $\mathcal{M}$ and attention module
  $\text{AM}_\text{w}$, discount factor $\gamma$
  for $i = 1, 2, ..., F$ do
    Randomly shuffle $\mathcal{A} \setminus \{a\}$ to sample a decision-making priority $\mathcal{P}_{\mathcal{A}\setminus\{a\}}$ of the remaining agents except
  agent $a$
    for $j = 0, 1, ..., H-1$ do
      $\hat{\boldsymbol{a}}^{upper} = \{\}$: predicted actions from all upper-level agents
      for $k$ in Concat$(\mathcal{P}, [a], \mathcal{P}_{\mathcal{A}\setminus\{a\}})$ do
        Sample $\hat{a}^k$ following $\pi(\cdot|\text{AM}_\text{a}(\boldsymbol{h}_{t+j}, \boldsymbol{a}^{upper}))$
```

```
            Append $\hat{a}^k$ to $\hat{\boldsymbol{a}}^{upper}$
        end for
        Rollout one step with the world model $\hat{\boldsymbol{o}}_{t+j+1}, \hat{r}_{t+j+1} = \mathcal{M}(\mathrm{AM_w}(\boldsymbol{h}_{t+j}, \boldsymbol{a}^{upper}))$
    end for
    Compute the return of the trajectory $v_i = \sum_{t'=t+1}^{t+H} \gamma^{t'-t-1}\hat{r}_{t'}$
end for
Compute the average return $v = \frac{1}{F}\sum_{i=1}^{F} v_i$
```

Algorithm 3 Launching Phase

```
Require: Decision-making priority $\mathcal{P}$, policy $\pi$ and $\mathrm{AM_a}$
  $\boldsymbol{a}_t^{upper} = \{\}$: actions from all upper-level agents
  for $i$ in $\mathcal{P}$ do
      Sample $a_t^i$ following $\pi_i(\cdot|\mathrm{AM_a}(\boldsymbol{h}_t, \boldsymbol{a}_t^{upper}))$
      Append $a_t^i$ to $\boldsymbol{a}_t^{upper}$
      /* Send $\boldsymbol{a}_t^{upper}$ to the lower agent*/
  end for
  Interact with the environment with $\boldsymbol{a}_t$
```

We also provide the pseudo-code of the local communication version as below:

Algorithm 4 Local Negotiation Phase of Agent a

```
Require: Neighbouring agents $\mathcal{N}$
  /* Agents communicate the hidden state $\boldsymbol{h}$ of their observations with neighbouring agents*/
  Compute local intention $v_a$ via Algorithm 5
  /* Send $v_a$ to neighbouring agents and receive $\{v_i\}_{i\in\mathcal{N}}$ from them */
  Set upper-level neighbouring agents $\mathcal{N}^{upper} = \{i \mid v_i > v_a, i \in \mathcal{N}\}$
  Set lower-level neighbouring agents $\mathcal{N}^{lower} = \{i \mid v_i < v_a, i \in \mathcal{N}\}$
```

Algorithm 5 Local Intention Value Calculation of Agent a

```
Require: Neighbouring agents $\mathcal{N}$, number of sampling trajectories $F$, length of predicted future trajectory
  $H$, policy $\pi$ and $\mathrm{AM_a}$, world model $\mathcal{M}$ and $\mathrm{AM_w}$, discount factor $\gamma$
  for $i = 1, 2, ..., F$ do
      Randomly shuffle $\mathcal{N}$ to sample a local decision-making priority $\mathcal{P}_{\mathcal{N}}$
      for $j = 0, 1, ..., H-1$ do
          $\hat{\boldsymbol{a}}^{upper} = \{\}$: predicted actions from all upper-level agents
          for $k$ in Concat$([a], \mathcal{P}_{\mathcal{N}})$ do
              Sample $\hat{a}^k$ following $\pi(\cdot|\mathrm{AM_a}(\boldsymbol{h}_{t+j}, \boldsymbol{a}^{upper}))$
              Append $\hat{a}^k$ to $\hat{\boldsymbol{a}}^{upper}$
          end for
          Rollout one step with the world model $\hat{\boldsymbol{o}}_{t+j+1}, \hat{r}_{t+j+1} = \mathcal{M}(\mathrm{AM_w}(\boldsymbol{h}_{t+j}, \boldsymbol{a}^{upper}))$
      end for
      Compute the return of the trajectory $v_i = \sum_{t'=t+1}^{t+H} \gamma^{t'-t-1}\hat{r}_{t'}$
  end for
  Compute the average return $v = \frac{1}{F}\sum_{i=1}^{F} v_i$
```

Algorithm 6 Local Launching Phase of Agent a

```
Require: Upper-level neighbouring agents $\mathcal{N}^{upper}$, lower-level neighbouring agents $\mathcal{N}^{lower}$, policy $\pi$ and
  $\mathrm{AM_a}$
  /* Receive upper-level actions $\boldsymbol{a}_t^{upper}$ from all upper-level neighbouring agents $\mathcal{N}^{upper}$ /*
  Sample $a_t^i$ following $\pi_i(\cdot|\mathrm{AM_a}(\boldsymbol{h}_t, \boldsymbol{a}_t^{upper}))$
  /* Send $a_t^i$ to all lower-level neighbouring agents $\mathcal{N}^{lower}$ */
  Interact with the environment with $\boldsymbol{a}_t$
```

Table 2: Hyperparameters for predator-prey, cooperative navigation, keep-away

Hyperparameter	SeqComm	Random-C	Fix-C	TarMAC	I2C	IS
discount (γ)			0.95,0.95,0.95			
batch size	–	–	–	–	800	1024
buffer capacity	–	–	–	–	1e^6	
number of processes		16,16,16			–	–
learning rate		1.5e^{-5}, 1e^{-5}, 4e^{-5}			1e^{-2}, 1e^{-3}, 1e^{-3}	1e^{-2}
H	10,10,20	–	–	–	–	–
F	$2, 2, 1$		–	–	–	–

F.2 Architecture and Hyperparameters

Our models, including SeqComm, Fix-C, and Random-C are trained based on MAPPO. The critic and policy network are realized by two fully connected layers. As for the attention module, key, query, and value have one fully connected layer each. The size of the hidden layers is 100. Tanh functions are used as nonlinearity. For I2C, we use their official code with default settings of basic hyperparameters and networks. As there is no released code of IS and TarMAC, we implement IS and TarMAC by ourselves, following the instructions mentioned in the original papers (Kim et al., 2021; Das et al., 2019).

For the world model, observations and actions are firstly encoded by a fully connected layer. The output size for the observation encoder is 48, and the output size for the action encoder is 16. Then the outputs of the encoder will be passed into the attention module with the same structure aforementioned. Finally, we use a fully connected layer to decode. In these layers, Tanh is used as the nonlinearity.

Table 2 summarize the hyperparameters used by SeqComm and the baselines in the MPE.

For SMAC, SeqComm, Random-C, Fix-C, and Centralized are based on the same architecture, and the hyperparameters stay the same. For MMM2 and 6z_vs_8z, the learning rate is 5e^{-5}, while for 10m_vs_11m, corridor, and 3s4z_vs_3s5z, learning rate is 7e^{-5}. For 8m_vs_9m, the learning rate is 3e^{-5}. H and F is set to 5 and 1, respectively. However, 20 and 2 is a better combination of H and F if computing resources are sufficient.

For TarMAC, the learning rate is 5e^{-5} for MMM2 and 6z_vs_8z. The learning rate is 7e^{-5} for other maps. TarMAC adopts MAPPO as the backbone and two-round communication mechanism.

For MAPPO, the learning rate is 5e^{-5} for MMM2 and 6z_vs_8z, and 7e^{-5} for 8m_vs_9m and 10m_vs_11m. For these four methods, the mini_batch is set to 1. As for other hyperparameters, we follow the default settings of the official code (Yu et al., 2021).

For QMIX, the learning rate is 5e^{-5}. The ϵ is 1 and the batch size is 32. The buffer size is 5e^3. For others, we follow the default settings of link https://github.com/starry-sky6688/MARL-Algorithms.git.

F.3 Attention Module

Attention module (AM) is applied to process messages in the world model, critic network, and policy network. AM consists of three components: query, key, and values. The output of AM is the weighted sum of values, where the weight of the value is determined by the dot product of the query and the corresponding key.

For AM in the world model denoted as $\mathrm{AM_w}$, agent i gets messages $\boldsymbol{m}_t^{-i} = \boldsymbol{h}_t^{-i}$ from all other agents at timestep t in negotiation phase, and predicts a query vector q_t^i following $\mathrm{AM}^i_{\mathrm{w},q}(h_t^i)$. The query is used to compute a dot product with keys $\boldsymbol{k}_t = [k_t^1, \cdots, k_t^n]$. Note that k_t^j is obtained by the message from agent j following $\mathrm{AM}^i_{\mathrm{a},k}(h_t^j)$ for $j \neq i$, and k_t^i is from $\mathrm{AM}^i_{\mathrm{neg},k}(h_t^i)$. Besides, it is scaled by $1/\sqrt{d_k}$ followed by a softmax to obtain attention weights α for each value vector:

$$\alpha_i = \text{softmax}\left[\frac{{q_t^i}^T k_t^1}{\sqrt{d_k}} \cdots \underbrace{\frac{{q_t^i}^T k_t^j}{\sqrt{d_k}}}_{\alpha_{ij}} \cdots \frac{{q_t^i}^T k_t^n}{\sqrt{d_k}}\right] \tag{1}$$

The output of attention module is defined as: $c_t^i = \sum_{j=1}^n \alpha_{ij} v_t^j$, where v_t^j is obtained from messages or its own hidden state of observation following $\text{AM}_{\text{w},v}^i(\cdot)$.

As for AM in the policy and critic network denoted as AM_a , agent i gets additional messages from upper-level agent in the launching phase. The message from upper-level and lower-level agent can be expanded as $\boldsymbol{m}_t^{upper} = [\boldsymbol{h}_t^{upper}, \boldsymbol{a}_t^{upper}]$ and $\boldsymbol{m}_t^{lower} = [\boldsymbol{h}_t^{lower}, 0]$, respectively. In addition, the query depends on agent's own hidden state of observation h_t^i, but keys and values are only from messages of other agents.

F.4 Training

The training of SeqComm is an extension of MAPPO. The observation encoder e, the critic V, and the policy π are respectively parameterized by θ_e, θ_v, θ_π. Besides, the attention module AM_a is parameterized by θ_a and takes as input the agent's hidden state, the messages (hidden states of other agents) in the negotiation phase, and the messages (the actions of upper-level agents) in launching phase. Let $\mathcal{D} = \{\tau_k\}_{k=1}^K$ be a set of trajectories by running policy in the environment. Note that we drop time t in the following notations for simplicity.

The value function is fitted by regression on mean-squared error:

$$\mathcal{L}(\theta_v, \theta_a, \theta_e) = \frac{1}{KT} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T-1} \left\| V(\text{AM}_\text{a}(e(\boldsymbol{o}), \boldsymbol{a}^{upper})) - \hat{R} \right\|_2^2 \tag{2}$$

where $\hat{R}$ is the discount rewards-to-go.

We update the policy by maximizing the PPO-Clip objective:

$$\mathcal{L}(\theta_\pi, \theta_a, \theta_e) = \frac{1}{KT} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T-1} \min\left(\frac{\pi(a|\text{AM}_\text{a}(e(\boldsymbol{o}), \boldsymbol{a}^{upper}))}{\pi_{old}(a|\text{AM}_\text{a}(e(\boldsymbol{o}), \boldsymbol{a}^{upper}))} A_{\pi_{old}}, g(\epsilon, A_{\pi_{old}})\right) \tag{3}$$

where $g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A \leq 0 \end{cases}$, and $A_{\pi_{old}}(\boldsymbol{o}, \boldsymbol{a}^{upper}, a)$ is computed using the GAE method.

The world model $\mathcal{M}$ is parameterized by θ_w is trained as a regression model using the training data set $\mathcal{S}$. It is updated with the loss:

$$\mathcal{L}(\theta_w) = \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r \in \mathcal{S}} \left\| (\boldsymbol{o}', r) - \mathcal{M}(\text{AM}_\text{w}(e(\boldsymbol{o}), \boldsymbol{a})) \right\|_2^2. \tag{4}$$

We trained our model on one GeForce GTX 1050 Ti and Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz.