A. Derivation of Meta-Multiagent Policy Gradient Theorem

Theorem 1. (Meta-Multiagent Policy Gradient Theorem) *For any stochastic game $\mathcal{M}_n$, the gradient of the meta-value function for agent i at state s_0 with respect to current policy parameters ϕ_0^i evolving in the environment along with other peer agents using initial parameters $\boldsymbol{\phi_0^{-i}}$ is:*

$$\nabla_{\phi_0^i} V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0,\phi_0^i) = \mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{\ell+1}}}|\boldsymbol{\phi_{\ell+1}})}\big[G^i(\tau_{\boldsymbol{\phi_{\ell+1}}})$$
$$\big(\underbrace{\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)}_{\text{Current Policy}} + \underbrace{\textstyle\sum_{\ell'=0}^{\ell}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\phi_{\ell'+1}^i)}_{\text{Own Learning}} + \underbrace{\textstyle\sum_{\ell'=0}^{\ell}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\boldsymbol{\phi_{\ell'+1}^{-i}})}_{\text{Peer Learning}}\big)\big]\Big].$$

Proof. We begin our derivation from the meta-value function defined in Equation (2) and expand it with the state-action value and joint actions, assuming the conditional independence between agents' actions (Wen et al., 2019):

$$\begin{aligned} V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0,\phi_0^i) =& \mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\mathbb{E}_{\tau_{\boldsymbol{\phi_{\ell+1}}}}\big[G^i(\tau_{\boldsymbol{\phi_{\ell+1}}})\big]\Big] = \mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[V^i_{\boldsymbol{\phi_{\ell+1}}}(s_0)\Big] \\ =& \mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big], \end{aligned} \tag{6}$$

where $Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})$ is the state-action value under the joint policy with parameters $\boldsymbol{\phi_{\ell+1}}$ at state s_0 with joint action $\boldsymbol{a_0}$.

In Equation (6), we note that both $\boldsymbol{\phi_{1:\ell}^i}$ and $\boldsymbol{\phi_{1:\ell}^{-i}}$ depend on ϕ_0^i. Considering the joint update from $\boldsymbol{\phi_0}$ to $\boldsymbol{\phi_1}$, for simplicity, we can write the gradients in the inner-loop (see Equation (3)) based on the multiagent stochastic policy gradient theorem (Wei et al., 2018):

$$\begin{aligned} \nabla_{\phi_0^{-i}}\mathbb{E}_{p(\tau_{\boldsymbol{\phi_0}}|\boldsymbol{\phi_0})}\Big[G^i(\tau_{\boldsymbol{\phi_0}})\Big] &= \sum_s\rho_{\boldsymbol{\phi_0}}(s)\sum_{a^i}\nabla_{\phi_0^i}\pi(a^i|s,\phi_0^i)\sum_{\boldsymbol{a^{-i}}}\boldsymbol{\pi}(\boldsymbol{a^{-i}}|s,\boldsymbol{\phi_0^{-i}})Q^i_{\boldsymbol{\phi_0}}(s,\boldsymbol{a}), \\ \nabla_{\boldsymbol{\phi_0^{-i}}}\mathbb{E}_{p(\tau_{\boldsymbol{\phi_0}}|\boldsymbol{\phi_0})}\Big[\boldsymbol{G^{-i}}(\tau_{\boldsymbol{\phi_0}})\Big] &= \sum_s\rho_{\boldsymbol{\phi_0}}(s)\sum_{\boldsymbol{a^{-i}}}\nabla_{\boldsymbol{\phi_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a^{-i}}|s,\boldsymbol{\phi_0^{-i}})\sum_{a^i}\pi(a^i|s,\phi_0^i)\boldsymbol{Q^{-i}_{\phi_0}}(s,\boldsymbol{a}), \end{aligned} \tag{7}$$

where $\rho_{\boldsymbol{\phi_0}}$ denotes the stationary distribution under the joint policy with parameters $\boldsymbol{\phi_0}$. Importantly, the inner-loop gradients for an agent i and its peers are a function of ϕ_0^i. Hence, the updated joint policy parameter $\boldsymbol{\phi_1}$ depends on ϕ_0^i. Following Equation (7), the successive inner-loop optimization until $\boldsymbol{\phi_{\ell+1}}$ results in dependencies between ϕ_0^i and $\boldsymbol{\phi_{1:\ell+1}^i}$ and between ϕ_0^i and $\boldsymbol{\phi_{1:\ell+1}^{-i}}$ (see Figure 1b).

Having identified which terms are dependent on ϕ_0^i, we continue from Equation (6) and derive the gradient of the meta-value function with respect to ϕ_0^i by applying the product rule:

$$\begin{aligned} \nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0,\phi_0^i) &= \nabla_{\phi_0^i}\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big]\Big] \\ &= \nabla_{\phi_0^i}\Big[\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}}p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}^i},\boldsymbol{\phi_{0:\ell}^{-i}})\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big] \\ &= \underbrace{\nabla_{\phi_0^i}\Big[\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}}p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}^i},\boldsymbol{\phi_{0:\ell}^{-i}})\Big]\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})+}_{\text{Term A}} \\ &\quad \underbrace{\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}}p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}^i},\boldsymbol{\phi_{0:\ell}^{-i}})\Big[\sum_{a_0^i}\nabla_{\phi_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\Big]\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})+}_{\text{Term B}} \\ &\quad \underbrace{\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}}p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}^i},\boldsymbol{\phi_{0:\ell}^{-i}})\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\Big[\sum_{\boldsymbol{a_0^{-i}}}\nabla_{\phi_0^i}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})\Big]Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})+}_{\text{Term C}} \\ &\quad \underbrace{\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}}p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}^i},\boldsymbol{\phi_{0:\ell}^{-i}})\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})\Big[\nabla_{\phi_0^i}Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big]}_{\text{Term D}}. \end{aligned} \tag{8}$$

We first focus on the derivative of the trajectories $\tau_{\boldsymbol{\phi_{0:\ell}}}$ in Term A:

$$\begin{aligned}
\nabla_{\phi_0^i}\Big[\sum_{\tau_{\boldsymbol{\phi_{0:\ell}}}} p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi^i_{0:\ell}},\boldsymbol{\phi^{-i}_{0:\ell}})\Big] &= \nabla_{\phi_0^i}\Big[\sum_{\tau_{\boldsymbol{\phi_0}}} p(\tau_{\boldsymbol{\phi_0}}|\phi_0^i,\boldsymbol{\phi_0^{-i}})\sum_{\tau_{\boldsymbol{\phi_1}}} p(\tau_{\boldsymbol{\phi_1}}|\phi_1^i,\boldsymbol{\phi_1^{-i}})\times\ldots\times\sum_{\tau_{\boldsymbol{\phi_\ell}}} p(\tau_{\boldsymbol{\phi_\ell}}|\phi_\ell^i,\boldsymbol{\phi_\ell^{-i}})\Big]\\
&= \Big[\sum_{\tau_{\boldsymbol{\phi_0}}}\nabla_{\phi_0^i} p(\tau_{\boldsymbol{\phi_0}}|\phi_0^i,\boldsymbol{\phi_0^{-i}})\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{0\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}})+\\
&\quad\Big[\sum_{\tau_{\boldsymbol{\phi_1}}}\nabla_{\phi_1^i} p(\tau_{\boldsymbol{\phi_1}}|\phi_1^i,\boldsymbol{\phi_1^{-i}})\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{1\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}})+\ldots+\\
&\quad\Big[\sum_{\tau_{\boldsymbol{\phi_\ell}}}\nabla_{\phi_\ell^i} p(\tau_{\boldsymbol{\phi_\ell}}|\phi_\ell^i,\boldsymbol{\phi_\ell^{-i}})\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{\ell\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}}),
\end{aligned}\tag{9}$$

where the probability of collecting a trajectory under the joint policy with parameters $\boldsymbol{\phi_\ell}$ is given by:

$$p(\tau_{\boldsymbol{\phi_\ell}}|\phi_\ell^i,\boldsymbol{\phi_\ell^{-i}}) = p(s_0)\prod\nolimits_{t=0}^{H}\pi(a_t^i|s_t,\phi_\ell^i)\boldsymbol{\pi}(\boldsymbol{a_t^{-i}}|s_t,\boldsymbol{\phi_\ell^{-i}})\mathcal{P}(s_{t+1}|s_t,\boldsymbol{a_t}).\tag{10}$$

Using Equation (10) and the log-derivative trick, Equation (9) can be further expressed as:

$$\begin{aligned}
&\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_0}}|\boldsymbol{\phi_0})}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{0\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}})+\\
&\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_1}}|\boldsymbol{\phi_1})}\nabla_{\phi_0^i}\Big(\log\pi(\tau_{\boldsymbol{\phi_1}}|\phi_1^i)+\log\boldsymbol{\pi}(\tau_{\boldsymbol{\phi_1}}|\boldsymbol{\phi_1^{-i}})\Big)\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{1\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}})+\ldots+\\
&\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell})}\nabla_{\phi_0^i}\Big(\log\pi(\tau_{\boldsymbol{\phi_\ell}}|\phi_\ell^i)+\log\boldsymbol{\pi}(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell^{-i}})\Big)\Big]\prod\nolimits_{\forall\ell'\in\{0,\ldots,\ell\}\backslash\{\ell\}}\sum_{\tau_{\boldsymbol{\phi_{\ell'}}}} p(\tau_{\boldsymbol{\phi_{\ell'}}}|\phi_{\ell'}^i,\boldsymbol{\phi_{\ell'}^{-i}})
\end{aligned}\tag{11}$$

where the summations of the log-terms, such as $\nabla_{\phi_0^i}\Big(\log\pi(\tau_{\boldsymbol{\phi_\ell}}|\phi_\ell^i)+\log\boldsymbol{\pi}(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell^{-i}})\Big)$ are *inherently* included due to the sequential dependencies between ϕ_0^i and $\boldsymbol{\phi_{1:\ell}}$. We use the result of Equation (11) and organize terms to arrive at the following expression for Term A in Equation (8):

$$\begin{aligned}
\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\Big(&\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)+\sum_{\ell'=0}^{\ell-1}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\phi_{\ell'+1}^i)+\sum_{\ell'=0}^{\ell-1}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\boldsymbol{\phi_{\ell'+1}^{-i}})\Big)\times\\
&\sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big].
\end{aligned}\tag{12}$$

Coming back to Term B-D in Equation (8), repeatedly unrolling the derivative of the Q-function $\nabla_{\phi_0^i}Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})$ by following Sutton & Barto (1998) yields:

$$\begin{aligned}
&\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\sum_s\rho_{\boldsymbol{\phi_{\ell+1}}}(s)\sum_{a^i}\nabla_{\phi_0^i}\pi(a^i|s,\phi_{\ell+1}^i)\sum_{\boldsymbol{a^{-i}}}\boldsymbol{\pi}(\boldsymbol{a^{-i}}|s,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi}+1}(s,\boldsymbol{a})\Big]+\\
&\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\sum_s\rho_{\boldsymbol{\phi_{\ell+1}}}(s)\sum_{\boldsymbol{a^{-i}}}\nabla_{\phi_0^i}\boldsymbol{\pi}(\boldsymbol{a^{-i}}|s,\boldsymbol{\phi_{\ell+1}^{-i}})\sum_{a^i}\pi(a^i|s,\phi_{\ell+1}^i)Q^i_{\boldsymbol{\phi_{\ell+1}}}(s,\boldsymbol{a})\Big],
\end{aligned}\tag{13}$$

which adds the consideration of future joint policy $\boldsymbol{\phi_{\ell+1}}$ to Equation (12). Finally, we summarize Equations (12) and (13) together and express in expectations:

$$\begin{aligned}
\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0,\phi_0^i) &= \mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{\ell+1}}}|\boldsymbol{\phi_{\ell+1}})}\big[G^i(\tau_{\boldsymbol{\phi_{\ell+1}}})\\
&\Big(\underbrace{\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)}_{\text{Current Policy}}+\underbrace{\sum\nolimits_{\ell'=0}^{\ell}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\phi_{\ell'+1}^i)}_{\text{Own Learning}}+\underbrace{\sum\nolimits_{\ell'=0}^{\ell}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\boldsymbol{\phi_{\ell'+1}^{-i}})}_{\text{Peer Learning}}\Big)\big]\Big].
\end{aligned}$$

□

B. Derivation of Meta-Policy Gradient Theorem

Remark 1. *Meta-PG can be considered as a special case of Meta-MAPG when assuming that other agents' learning in the environment is independent of the meta-agent's behavior.*

Proof. The framework by Al-Shedivat et al. (2018) makes the implicit assumption that there exist no sequential dependencies between the future parameters of other agents $\boldsymbol{\phi_{1:L}^{-i}}$ and ϕ_0^i. This assumption implies that the peers' policy updates in Equation (7) are not a function of a meta-agent's policy. As a result, the gradients of the peers' log-terms with respect to ϕ_0^i in Appendix A, such as $\nabla_{\phi_0^i}\Big(\log\boldsymbol{\pi}(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell^{-i}})\Big)$, become zero. Specifically, Term A in Equation (8) simplifies to:

$$\begin{aligned}\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\Big(\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)+\sum_{\ell'=0}^{\ell-1}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\phi_{\ell'+1}^i)\Big)\times\\ \sum_{a_0^i}\pi(a_0^i|s_0,\phi_{\ell+1}^i)\sum_{\boldsymbol{a_0^{-i}}}\boldsymbol{\pi}(\boldsymbol{a_0^{-i}}|s_0,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi_{\ell+1}}}(s_0,\boldsymbol{a_0})\Big].\end{aligned} \tag{14}$$

Similarly, Term B-D in Equation (8) become:

$$\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\sum_s\rho_{\boldsymbol{\phi_{\ell+1}}}(s)\sum_{a^i}\nabla_{\phi_0^i}\pi(a^i|s,\phi_{\ell+1}^i)\sum_{\boldsymbol{a^{-i}}}\boldsymbol{\pi}(\boldsymbol{a^{-i}}|s,\boldsymbol{\phi_{\ell+1}^{-i}})Q^i_{\boldsymbol{\phi}+1}(s,\boldsymbol{a})\Big]. \tag{15}$$

Finally, summarizing Equations (14) and (15) together, and expressing in expectations results in Meta-PG:

$$\begin{aligned}\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0,\phi_0^i)=\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{0:\ell}}}|\boldsymbol{\phi_{0:\ell}})}\Big[\mathbb{E}_{p(\tau_{\boldsymbol{\phi_{\ell+1}}}|\boldsymbol{\phi_{\ell+1}})}\big[G^i(\tau_{\boldsymbol{\phi_{\ell+1}}})\\ \big(\underbrace{\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_0}}|\phi_0^i)}_{\text{Current Policy}}+\underbrace{\textstyle\sum_{\ell'=0}^{\ell}\nabla_{\phi_0^i}\log\pi(\tau_{\boldsymbol{\phi_{\ell'+1}}}|\phi_{\ell'+1}^i)}_{\text{Own Learning}}\big)\big]\Big].\end{aligned}$$

□

C. Stateless Zero-Sum Game Details

Derivation of Meta-MAPG. In the stateless zero-sum game, a meta-agent i and an opponent j maximize simple value functions $V^i_{\boldsymbol{\phi_\ell}}=\phi_\ell^i\phi_\ell^j$ and $V^j_{\boldsymbol{\phi_\ell}}=-\phi_\ell^i\phi_\ell^j$ respectively, where $\phi_\ell^i,\phi_\ell^j\in\mathbb{R}$. We note that this domain has the episode horizon H of 1 with a deterministic and continuous action space, where the action is equivalent to the policy parameter: $a_\ell^i=\phi_\ell^i$ and $a_\ell^j=\phi_\ell^j$. Because there are no stochastic factors in this domain, the meta-value function defined in Equation (2) can be simplified without the expectations:

$$V^i_{\boldsymbol{\phi_{0:\ell+1}}}(\phi_0^i)=V^i_{\boldsymbol{\phi_{\ell+1}}}=\phi^i_{\ell+1}\phi^j_{\ell+1}. \tag{16}$$

Assuming the maximum chain length L of 1 for clarity, the inner-loop updates from $\boldsymbol{\phi_0}$ to $\boldsymbol{\phi_1}$ are:

$$\begin{aligned}\phi_1^i&=\phi_0^i+\alpha\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_0}}=\phi_0^i+\alpha\nabla_{\phi_0^i}\big[\phi_0^i\phi_0^j\big]=\phi_0^i+\alpha\phi_0^j,\\ \phi_1^j&=\phi_0^j+\alpha\nabla_{\phi_0^j}V^j_{\boldsymbol{\phi_0}}=\phi_0^j+\alpha\nabla_{\phi_0^j}\big[-\phi_0^i\phi_0^j\big]=\phi_0^j-\alpha\phi_0^i,\end{aligned} \tag{17}$$

where α is the inner-loop learning rate. Then, the meta-multiagent policy gradient can be directly computed without using the log-derivative trick:

$$\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:1}}}=\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_1}}=\nabla_{\phi_0^i}\big[\phi_1^i\phi_1^j\big]=\big[\nabla_{\phi_0^i}\phi_1^i\big]\phi_1^j+\big[\nabla_{\phi_0^i}\phi_1^j\big]\phi_1^i=\phi_1^j-\alpha\phi_1^i. \tag{18}$$

During meta-training, the initial policy parameter ϕ_0^i will be updated with the outer-loop learning rate β:

$$\phi_0^i:=\phi_0^i+\beta\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:1}}}=\phi_0^i+\beta\big[\phi_1^j-\alpha\phi_1^i\big]. \tag{19}$$

Derivation of Meta-PG. For Meta-PG (Al-Shedivat et al., 2018), the framework assumes that there is no dependency between ϕ_0^i and ϕ_1^j. Thus, the term $\big[\nabla_{\phi_0^i}\phi_1^j\big]$ in Equation (18) becomes zero, resulting in the meta-update of:

$$\phi_0^i:=\phi_0^i+\beta\nabla_{\phi_0^i}V^i_{\boldsymbol{\phi_{0:1}}}=\phi_0^i+\beta\phi_1^j. \tag{20}$$

Hyperparameter. We used the following hyperparameters: 1) randomly sampled initial opponent policy parameter from -1 to 1 (i.e., $p(\boldsymbol{\phi_0^{-i}})=[-1,1]$), 2) $\alpha=0.75$, and 3) $\beta=0.01$.

D. Meta-MAPG Algorithm

Algorithm 1 Meta-Learning at Training Time

```
Require: $p(\boldsymbol{\phi_0^{-i}})$: Distribution over peer agents' initial policy
    parameters; $\boldsymbol{\alpha}, \beta$: Learning rates
 1: Randomly initialize $\phi_0^i$
 2: while $\phi_0^i$ has not converged do
 3:     Sample a meta-train batch of $\boldsymbol{\phi_0^{-i}} \sim p(\boldsymbol{\phi_0^{-i}})$
 4:     for each $\boldsymbol{\phi_0^{-i}}$ do
 5:         for $\ell = 0, \ldots, L$ do
 6:             Sample and store trajectory $\tau_{\boldsymbol{\phi_\ell}} \sim p(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell})$
 7:             Compute $\boldsymbol{\phi_{\ell+1}} = f(\boldsymbol{\phi_\ell}, \tau_{\boldsymbol{\phi_\ell}}, \boldsymbol{\alpha})$ from inner-loop opti-
                mization (Equation (3))
 8:         end for
 9:     end for
10:     Update $\phi_0^i \leftarrow \phi_0^i + \beta \sum_{\ell=0}^{L-1} \nabla_{\phi_0^i} V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0, \phi_0^i)$ based on
        Equation (4)
11: end while
```

Algorithm 2 Meta-Learning at Execution Time

```
Require: $p(\boldsymbol{\phi_0^{-i}})$: Distribution over peer agents' initial policy
    parameters; $\boldsymbol{\alpha}$: Learning rates; Optimized $\phi_0^{i*}$
1: Initialize $\phi_0^i \leftarrow \phi_0^{i*}$
2: Sample a meta-test batch of $\boldsymbol{\phi_0^{-i}} \sim p(\boldsymbol{\phi_0^{-i}})$
3: for each $\boldsymbol{\phi_0^{-i}}$ do
4:     for $\ell = 0, \ldots, L$ do
5:         Sample trajectory $\tau_{\boldsymbol{\phi_\ell}} \sim p(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell})$
6:         Compute $\boldsymbol{\phi_{\ell+1}} = f(\boldsymbol{\phi_\ell}, \tau_{\boldsymbol{\phi_\ell}}, \alpha)$ from inner-loop opti-
           mization (Equation (3))
7:     end for
8: end for
```

D.1. Meta-MAPG with Opponent Modeling

Algorithm 3 Meta-Learning at Training Time with OM

```
Require: $p(\boldsymbol{\phi_0^{-i}})$: Distribution over peer agents' initial policy
    parameters; $\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}^{-i}, \hat{\boldsymbol{\eta}}^{-i}, \beta$: Learning rates
 1: Randomly initialize $\phi_0^i$
 2: while $\phi_0^i$ has not converged do
 3:     Sample a meta-train batch of $\boldsymbol{\phi_0^{-i}} \sim p(\boldsymbol{\phi_0^{-i}})$
 4:     for each $\boldsymbol{\phi_0^{-i}}$ do
 5:         Randomly initialize $\hat{\boldsymbol{\phi}}_0^{-i}$
 6:         for $\ell = 0, \ldots, L$ do
 7:             Sample and store trajectory $\tau_{\boldsymbol{\phi_\ell}} \sim p(\tau_{\boldsymbol{\phi_\ell}}|\boldsymbol{\phi_\ell})$
 8:             Approximate $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i} = f(\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}, \tau_{\boldsymbol{\phi_\ell}}, \hat{\boldsymbol{\eta}}^{-i})$ using oppo-
                nent modeling (Algorithm 4)
 9:             Compute $\boldsymbol{\phi_{\ell+1}} = f(\boldsymbol{\phi_\ell}, \tau_{\boldsymbol{\phi_\ell}}, \boldsymbol{\alpha})$ from inner-loop opti-
                mization (Equation (3))
10:             Compute $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell+1}}^{-i} = f(\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}, \tau_{\boldsymbol{\phi_\ell}}, \hat{\boldsymbol{\alpha}}^{-i})$ from inner-loop
                optimization (Equation (3))
11:         end for
12:     end for
13:     Update $\phi_0^i \leftarrow \phi_0^i + \beta \sum_{\ell=0}^{L-1} \nabla_{\phi_0^i} V^i_{\boldsymbol{\phi_{0:\ell+1}}}(s_0, \phi_0^i)$ based on
        Equation (4) and $\hat{\boldsymbol{\phi}}_{\boldsymbol{1:L}}^{-i}$
14: end while
```

Algorithm 4 Opponent Modeling

```
1: function Opponent Modeling($\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}, \tau_{\boldsymbol{\phi_\ell}}, \hat{\boldsymbol{\eta}}^{-i}$)
2:     while $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}$ has not converged do
3:         Compute log-likelihood $\mathcal{L}_{\text{likelihood}} = f(\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}, \tau_{\boldsymbol{\phi_\ell}})$ based
           on Equation (21)
4:         Update $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i} \leftarrow \hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i} + \hat{\boldsymbol{\eta}}^{-i} \nabla_{\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}} \mathcal{L}_{\text{likelihood}}$
5:     end while
6:     return $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}$
7: end function
```

We explain Meta-MAPG with opponent modeling (OM) for settings where a meta-agent cannot access the policy parameters of its peers during meta-training. Our decentralized meta-training method in Algorithm 3 replaces the other agents' true policy parameters $\boldsymbol{\phi}_{\boldsymbol{1:L}}^{-i}$ with inferred parameters $\hat{\boldsymbol{\phi}}_{\boldsymbol{1:L}}^{-i}$ in computing the peer learning gradient. Specifically, we follow Foerster et al. (2018a) for opponent modeling and estimate $\hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}$ from $\tau_{\boldsymbol{\phi_\ell}}$ using log-likelihood $\mathcal{L}_{\text{likelihood}}$ (Line 8 in Algorithm 3):

$$\mathcal{L}_{\text{likelihood}} = \sum_{t=0}^{H} \log \boldsymbol{\pi}^{-i}(\boldsymbol{a}_t^{-i}|s_t, \hat{\boldsymbol{\phi}}_{\boldsymbol{\ell}}^{-i}), \tag{21}$$

where $s_t, \boldsymbol{a}_t^{-i} \in \tau_{\boldsymbol{\phi_\ell}}$. A meta-agent can obtain $\hat{\boldsymbol{\phi}}_{\boldsymbol{1:L}}^{-i}$ by iteratively applying the opponent modeling procedure until the maximum chain length of L. We also apply the inner-loop update with the Differentiable Monte-Carlo Estimator (DiCE) (Foerster et al., 2018c) to the inferred policy parameters of peer agents (Line 10 in Algorithm 3). By applying DiCE, we can save the sequential dependencies between ϕ_0^i and updates to the policy parameters of peer agents $\hat{\boldsymbol{\phi}}_{\boldsymbol{1:L}}^{-i}$ in a computation graph and compute the peer learning gradient efficiently via automatic-differentiation (Line 13 in Algorithm 3).

E. Additional Implementation Details

E.1. Network Structure

Our neural networks for the policy and value function consist of a fully-connected input layer with 64 units followed by a single-layer LSTM with 64 units and a fully-connected output layer. We reset the LSTM states to zeros at the beginning of trajectories and retain them until the end of episodes. The LSTM policy outputs a probability for the categorical distribution in the iterated games (i.e., IPD, RPS). For the 2-Agent HalfCheetah domain, the policy outputs a mean and variance for the Gaussian distribution. We empirically observe that no parameter sharing between the policy and value network results in more stable learning than sharing the network parameters.

E.2. Optimization

We detail additional important notes about our implementation:

• We apply the linear feature baseline (Duan et al., 2016a) and generalized advantage estimation (GAE) (Schulman et al., 2016) during the inner-loop and outer-loop optimization, respectively, to reduce the variance in the policy gradient.

• We use DiCE (Foerster et al., 2018c) to compute the peer learning gradient efficiently. Specifically, we apply DiCE during the inner-loop optimization and save the sequential dependencies between ϕ_0^i and $\boldsymbol{\phi}_{\boldsymbol{1:L}}^{\boldsymbol{-i}}$ in a computation graph. Because the computation graph has the sequential dependencies, we can compute the peer learning gradient by the backpropagation of the meta-value function via the automatic-differentiation toolbox.

• Learning from diverse peers can potentially cause conflicting gradients and unstable learning. In IPD, for instance, a strategy to adapt against cooperating peers can be completely opposite to the adaptation strategy against defecting peers, resulting in conflicting gradients. To address this potential issue, we use the projecting conflicting gradients (PCGrad) (Yu et al., 2020) during the outer-loop optimization. We also have tested the baseline methods with PCGrad.

• We use a distributed training to speed up the meta-optimization. Each thread interacts with a Markov chain of policies until the chain horizon and then computes the meta-optimization gradients using Equation (4). Then, similar to Mnih et al. (2016), each thread asynchronously updates the shared meta-agent's policy and value network parameters.

F. Additional Baseline Details

We train all adaptation methods based on a meta-training set until convergence. We then measure the adaptation performance on a meta-testing set using the best-learned policy determined by a meta-validation set.

F.1. Meta-PG

We have improved the Meta-PG baseline itself beyond its implementation in the original work (Al-Shedivat et al., 2018) to further isolate the importance of the peer learning gradient term. Specifically, compared to Al-Shedivat et al. (2018), we make the following theoretical contributions to build on:

1) **Underlying problem statement:** Al-Shedivat et al. (2018) bases their problem formulation off that of multi-task / continual single-agent RL. In contrast, ours is based on a general stochastic game between n agents (Shapley, 1953).

2) **A Markov chain of joint policies:** Al-Shedivat et al. (2018) treats an evolving peer agent as an external factor, resulting in the absence of the sequential dependencies between a meta-agent's current policy and the peer agents' future policies in the Markov chain. However, our important insight is that the sequential dependencies exist in general multiagent settings as the peer agents are also learning agents based on trajectories by interacting with a meta-agent (see Figure 1b).

3) **Meta-objective:** The meta-objective defined in Al-Shedivat et al. (2018) is based on single-agent settings. In contrast, our meta-objective is based on general multiagent settings (see Equations (2) and (3)).

4) **Meta-optimization gradient:** Compared to Al-Shedivat et al. (2018), our meta-optimization gradient inherently includes the additional term of the peer learning gradient that considers how an agent can directly influence peer's learning.

5) **Importance sampling:** Compared to Al-Shedivat et al. (2018), we avoid using the importance sampling during meta-testing by modifying the meta-value function. Specifically, the framework uses a meta-value function on a pair consecutive joint policies, denoted $V^i_{\boldsymbol{\phi}_{\boldsymbol{\ell:\ell+1}}}(s_0, \phi_0^i)$, which assumes initializing every ϕ_ℓ^i from ϕ_0^i. However, as noted in Al-Shedivat et al. (2018), this assumption requires interacting with the same peers multiple times and is often impossible during meta-testing.

To address this issue, the framework uses the importance sampling correction during meta-testing. However, the correction generally suffers from high variance (Wang et al., 2016b). As such, we effectively avoid using the correction by initializing from ϕ_0^i only once at the beginning of Markov chains for both meta-training and meta-testing.

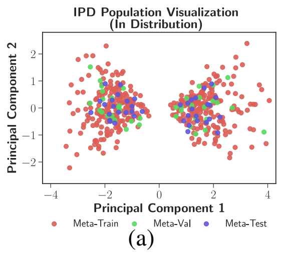
F.2. LOLA-DiCE

We used an open-source PyTorch implementation for LOLA-DiCE: https://github.com/alexis-jacq/LOLA_DiCE. We make minor changes to the code, such as adding the LSTM policy and value function.

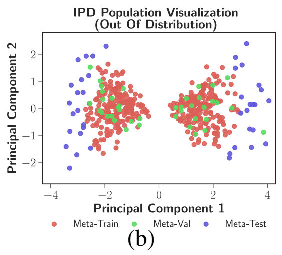
G. Additional Experiment Details

G.1. IPD

We choose to represent the peer agent j's policy as a tabular representation to effectively construct the population of initial personas $p(\boldsymbol{\phi}_0^{-i})$ for the meta-learning setup. Specifically, the tabular policy has a dimension of 5 that corresponds to the number of states in IPD. Then, we randomly sample a probability between 0.5 and 1.0 and a probability between 0 and 0.5 at each state to construct the cooperating and defecting population, respectively. As such, the tabular representation enables us to sample as many as personas but also controllable distribution $p(\boldsymbol{\phi}_0^{-i})$ by merely adjusting the probability range. We sample a total of 480 initial personas, including cooperating personas and defecting personas, and split them into 400 for meta-training, 40 for meta-validation, and 40 for meta-testing. Figure 7a and Figure 7b visualize in and out of distribution, respectively, where we used the principal component analysis (PCA) with two components.



(a)



(b)

Figure 7. (**a**) and (**b**) Visualization of j's initial policy for in distribution and out of distribution meta-testing, respectively, where the out of distribution split has a smaller overlap between the policies used for meta-training/validation and those used for meta-testing.

G.2. RPS

In RPS, we follow the same meta-learning setup as in IPD, except we sample a total of 720 initial opponent personas, including rock, paper, and scissors personas, and split them into 600 for meta-training, 60 for meta-validation, and 60 for meta-testing. Additionally, because RPS has three possible actions, we sample a rock preference probability between $1/3$ and 1 for building the rock persona population, where the rock probability is larger than the other two action probabilities. We follow the same procedure for constructing the paper and scissors persona population.

G.3. 2-Agent HalfCheetah

We used an open source implementation for multiagent-MuJoCo benchmark: https://github.com/schroederdewitt/multiagent_mujoco. Agents in our experiments receive state observations that include information about all the joints. For the meta-learning setup, we pre-train a teammate j with an LSTM policy that has varying expertise in moving to the left direction. Specifically, we train the teammate up to 500 train iterations and save a checkpoint at each iteration. Intuitively, as the number of train iteration increases, the teammate gains more expertise. We then use the checkpoints from 0 to 300 iterations as the meta-train/val (randomly split them into 275 for meta-training and 25 for meta-validation) and from 475 and 500 iterations as the meta-test distribution (see Figure 8). We construct the distribution with the gap to ensure that the meta-testing distribution has a sufficient difference to the meta-train/val so that we can test the generalization of our approach. As in IPD and RPS, the teammate j updates its policy based on the policy gradient with the linear feature baseline.

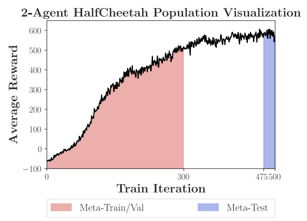


Figure 8. Visualization of a teammate j's initial expertise in the 2-Agent HalfCheetah domain, where the meta-test distribution has a sufficient difference to meta-train/val.

H. Analysis on Joint Policy Dynamics

H.1. IPD

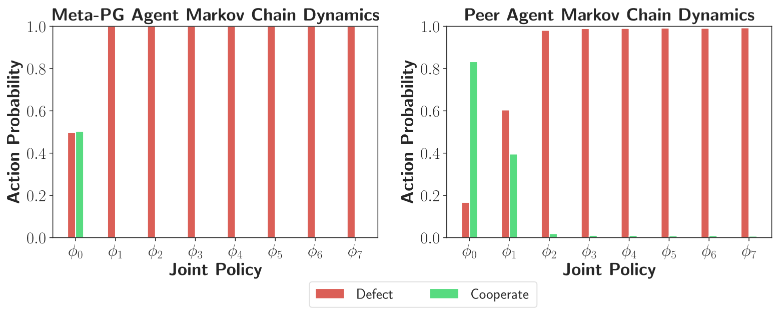


Figure 9. Action probability dynamics with Meta-PG in IPD with a cooperating persona peer

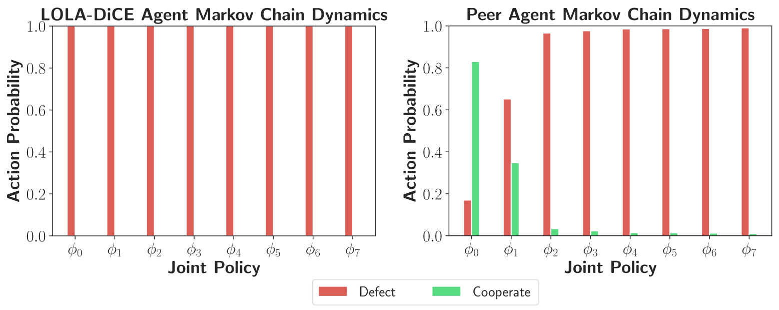


Figure 10. Action probability dynamics with LOLA-DiCE in IPD with a cooperating persona peer

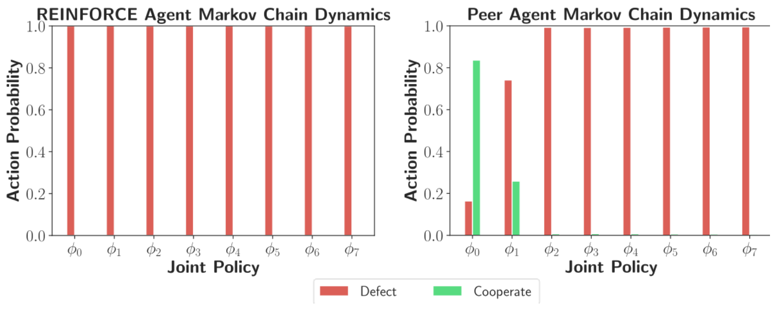


Figure 11. Action probability dynamics with REINFORCE in IPD with a cooperating persona peer

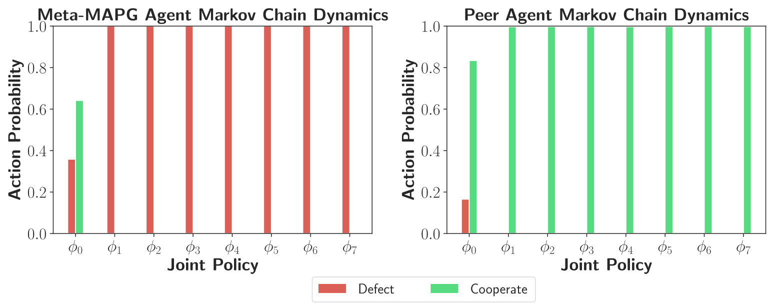


Figure 12. Action probability dynamics with Meta-MAPG in IPD with a cooperating persona peer

H.2. RPS

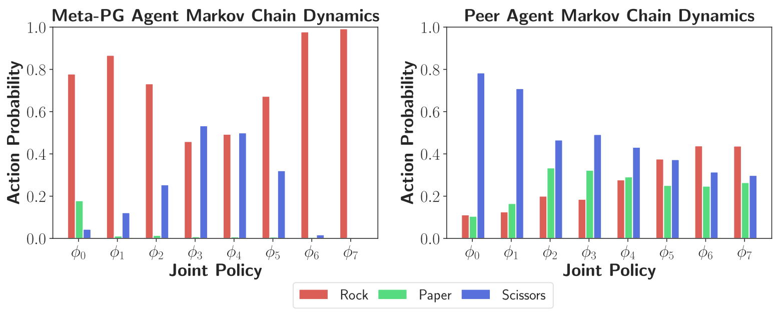


Figure 13. Action Probability Dynamics with Meta-PG in RPS with a scissors persona opponent

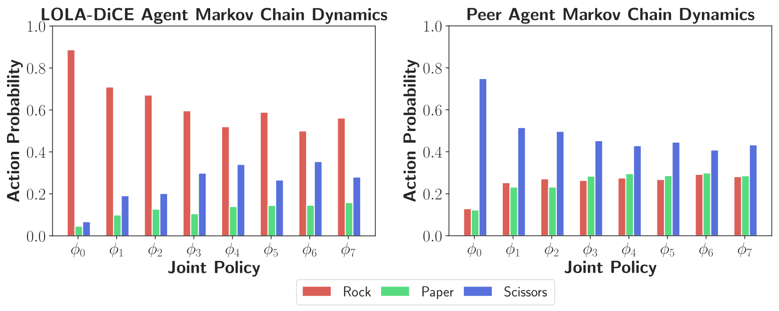


Figure 14. Action Probability Dynamics with LOLA-DiCE in RPS with a scissors persona opponent

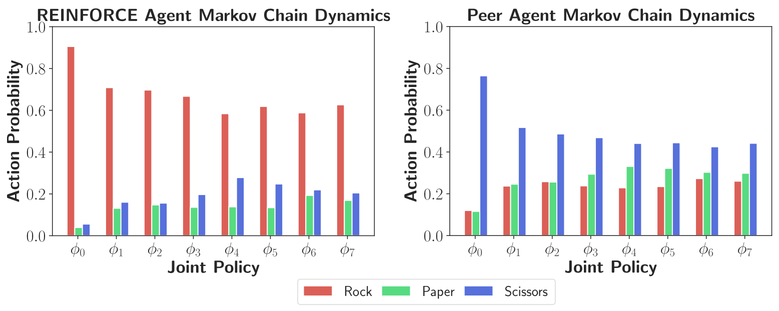


Figure 15. Action Probability Dynamics with REINFORCE in RPS with a scissors persona opponent

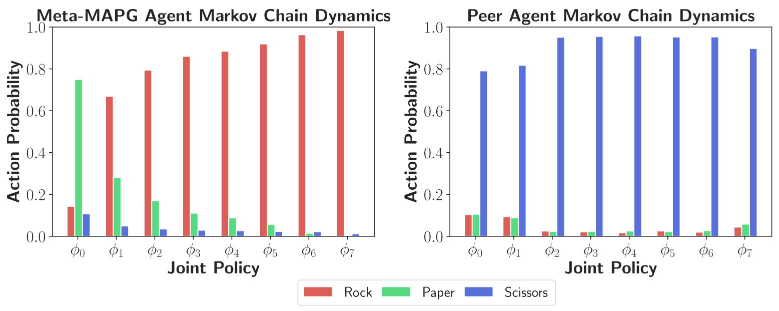


Figure 16. Action Probability Dynamics with Meta-MAPG in RPS with a scissors persona opponent

I. Hyperparameter Details

We report our hyperparameter values that we used for each of the methods in our experiments:

I.1. Meta-MAPG and Meta-PG

Hyperparameter	Value
Trajectory batch size K	4, 8, 16, 32, 64
Number of parallel threads	5
Actor learning rate (inner)	1.0, 0.1
Actor learning rate (outer)	1e-4
Critic learning rate (outer)	1.5e-4
Episode horizon H	150
Max chain length L	7
GAE λ	0.95
Discount factor γ	0.96

Table 1. IPD

Hyperparameter	Value
Trajectory batch size K	64
Number of parallel threads	5
Actor learning rate (inner)	0.01
Actor learning rate (outer)	1e-5
Critic learning rate (outer)	1.5e-5
Episode horizon H	150
Max chain length L	7
GAE λ	0.95
Discount factor γ	0.90

Table 2. RPS

Hyperparameter	Value
Trajectory batch size K	64
Number of parallel threads	5
Actor learning rate (inner)	0.005
Actor learning rate (outer)	5e-5
Critic learning rate (outer)	5.5e-5
Episode horizon H	200
Max chain length L	2
GAE λ	0.95
Discount factor γ	0.95

Table 3. 2-Agent HalfCheetah

I.2. LOLA-DiCE

Hyperparameter	Value
Trajectory batch size K	4, 8, 16, 32, 64
Actor learning rate	1.0, 0.1
Critic learning rate	1.5e-3
Episode horizon H	150
Max chain length L	7
Number of Look-Ahead	1, 3, 5
Discount factor γ	0.96

Table 4. IPD

Hyperparameter	Value
Trajectory batch size K	64
Actor learning rate	0.01
Critic learning rate	1.5e-5
Episode horizon H	150
Max chain length L	7
Number of Look-Ahead	1
Discount factor γ	0.90

Table 5. RPS

Hyperparameter	Value
Trajectory batch size K	64
Actor learning rate	0.005
Critic learning rate	5.5e-5
Episode horizon H	200
Max chain length L	2
Number of Look-Ahead	1
Discount factor γ	0.95

Table 6. 2-Agent HalfCheetah

I.3. REINFORCE

Hyperparameter	Value
Trajectory batch size K	4, 8, 16, 32, 64
Actor learning rate	1.0, 0.1
Episode horizon H	150
Max chain length L	7
Discount factor γ	0.96

Table 7. IPD

Hyperparameter	Value
Trajectory batch size K	64
Actor learning rate	0.01
Episode horizon H	150
Max chain length L	7
Discount factor γ	0.90

Table 8. RPS

Hyperparameter	Value
Trajectory batch size K	64
Actor learning rate	0.005
Episode horizon H	200
Max chain length L	2
Discount factor γ	0.95

Table 9. 2-Agent HalfCheetah