A Proofs

A.1 Proof of Lemma 1

Lemma 1 (Joint Policy Evaluation). *Consider the modified Bellman backup operator $\Gamma_{\pi_{\text{jt}}}$ (5) and a mapping $Q^0_{\text{jt}} : S \times U \to \mathbb{R}$ with $|U| < \infty$, and define $Q^{k+1}_{\text{jt}} = \Gamma_{\pi_{\text{jt}}} Q^k_{\text{jt}}$. Then, the sequence Q^k_{jt} will converge to the joint Q-function of π_{jt} as $k \to \infty$.*

Proof. First, define the augmented reward[2] as:

$$r_{\pi_{\text{jt}}}(\boldsymbol{s}, \boldsymbol{u}) := r(\boldsymbol{s}, \boldsymbol{u}) - \alpha \, \mathbb{E}_{\boldsymbol{s}'} \left[\mathbb{E}_{\pi_{\text{jt}}} \left[\sum_i \log \frac{\pi_i(u_i|\boldsymbol{s}')}{\pi_i(u_i|s_i'^{+})} \right] \right].$$

Then, rewrite the update rule as:

$$Q_{\text{jt}}(\boldsymbol{s}, \boldsymbol{u}) \leftarrow r_{\pi_{\text{jt}}}(\boldsymbol{s}, \boldsymbol{u}) + \gamma \, \mathbb{E}_{\boldsymbol{s}', \boldsymbol{u}' \sim \pi_{\text{jt}}}[Q_{\text{jt}}(\boldsymbol{s}', \boldsymbol{u}')].$$

Last, apply the standard convergence results for policy evaluation (Sutton and Barto, 2018). □

A.2 Proof of Lemma 2

Lemma 2 (Individual Policy Improvement). *Let π_i^{new} be the optimizer of the maximization problem in (7). Then, we have $Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{new}}}(\boldsymbol{s}, \boldsymbol{u}) \geq Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}, \boldsymbol{u})$ for all $(\boldsymbol{s}, \boldsymbol{u}) \in |S| \times |U|$ with $|U| < \infty$, where $\pi_{\text{jt}}^{\text{old}}(\boldsymbol{u}|\boldsymbol{s}) = \prod_i \pi_i^{\text{old}}(u_i|\boldsymbol{s})$ and $\pi_{\text{jt}}^{\text{new}}(\boldsymbol{u}|\boldsymbol{s}) = \prod_i \pi_i^{\text{new}}(u_i|\boldsymbol{s})$.*

Proof. As π_i^{new} optimizes (7), we can have:

$$\mathbb{E}_{\pi_i^{\text{new}}} \left[Q_i^{\pi_i^{\text{old}}}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i^{\text{new}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] \geq \mathbb{E}_{\pi_i^{\text{old}}} \left[Q_i^{\pi_i^{\text{old}}}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i^{\text{old}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right]. \tag{19}$$

Since we assume that:

$$Q_{\text{jt}}^{\pi_{\text{jt}}}(\boldsymbol{s}, \boldsymbol{u}) = \sum_i w_i(\boldsymbol{s}) * Q_i^{\pi_i}(\boldsymbol{s}, u_i) + b(\boldsymbol{s}),$$

we can have:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{u} \sim \pi_{\text{jt}}^{\text{new}}} \left[Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}, \boldsymbol{u}) - \alpha \sum_i \log \frac{\pi_i^{\text{new}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] \\
&= \mathbb{E}_{\boldsymbol{u} \sim \pi_{\text{jt}}^{\text{new}}} \left[\sum_i w_i(\boldsymbol{s}) * Q_i^{\pi_i^{\text{old}}}(\boldsymbol{s}, u_i) + b(\boldsymbol{s}) - \alpha \sum_i \log \frac{\pi_i^{\text{new}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] \\
&= \sum_i \mathbb{E}_{u_i \sim \pi_i^{\text{new}}} \left[w_i(\boldsymbol{s}) * Q_i^{\pi_i^{\text{old}}}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i^{\text{new}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] + b(\boldsymbol{s}) \\
&\geq \sum_i \mathbb{E}_{u_i \sim \pi_i^{\text{old}}} \left[w_i(\boldsymbol{s}) * Q_i^{\pi_i^{\text{old}}}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i^{\text{old}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] + b(\boldsymbol{s}) \\
&= \mathbb{E}_{\boldsymbol{u} \sim \pi_{\text{jt}}^{\text{old}}} \left[Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}, \boldsymbol{u}) - \alpha \sum_i \log \frac{\pi_i^{\text{old}}(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)} \right] \\
&= V_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}),
\end{aligned} \tag{20}$$

where the inequality is from plugging in (19).

[2] We assume $\pi_i(u_i|\boldsymbol{s})$ and $\pi_i(u_i|s_i^+)$ to be ϵ-soft policy (Sutton and Barto, 2018) to avoid the log term being undefined.

Last, considering the modified Bellman equation, the following holds:

$$
\begin{aligned}
Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}, \boldsymbol{u}) &= r(\boldsymbol{s}, \boldsymbol{u}) + \gamma\, \mathbb{E}_{\boldsymbol{s}'}\left[V_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}')\right] \\
&\leq r(\boldsymbol{s}, \boldsymbol{u}) + \gamma\, \mathbb{E}_{\boldsymbol{s}'}\left[\mathbb{E}_{\boldsymbol{u}' \sim \pi_{\text{jt}}^{\text{new}}}\left[Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}(\boldsymbol{s}', \boldsymbol{u}') - \alpha \sum_i \log \frac{\pi_i^{\text{new}}(u_i'|\boldsymbol{s}')}{\pi_i(u_i'|s_i'^{+})}\right]\right] \\
&\vdots \\
&\leq Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{new}}}(\boldsymbol{s}, \boldsymbol{u}),
\end{aligned}
$$

where we have repeatedly expanded $Q_{\text{jt}}^{\pi_{\text{jt}}^{\text{old}}}$ on the RHS by applying the modified Bellman equation and the inequality in (20). □

A.3 Proof of Theorem 1

Theorem 1 (Multi-Agent Policy Iteration with a Fixed Marginal Distribution). *For any joint policy π_{jt}, if we repeatedly apply joint policy evaluation and individual policy improvement. Then the joint policy $\pi_{\text{jt}}(\boldsymbol{u}|\boldsymbol{s}) = \prod_{i=1}^{n} \pi_i(u_i|\boldsymbol{s})$ will eventually converge to π_{jt}^*, such that $Q_{\text{jt}}^{\pi_{\text{jt}}^*}(\boldsymbol{s}, \boldsymbol{u}) \geq Q_{\text{jt}}^{\pi_{\text{jt}}}(\boldsymbol{s}, \boldsymbol{u})$ for all π_{jt}, assuming $|U| < \infty$.*

Proof. First, by Lemma 2, the sequence $\{\pi_{\text{jt}}^k\}$ monotonically improves with $Q_{\text{jt}}^{\pi_{\text{jt}}^{k+1}} \geq Q_{\text{jt}}^{\pi_{\text{jt}}^k}$. Since the augmented reward is bounded, then $Q_{\text{jt}}^{\pi_{\text{jt}}^k}$ is bounded. Thus, this sequence must converge to some π_{jt}^*. Then, at convergence, we have the following inequality:

$$
\mathbb{E}_{\pi_i^*}\left[Q_i^{\pi_i^*}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i^*(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)}\right] \geq \mathbb{E}_{\pi_i}\left[Q_i^{\pi_i^*}(\boldsymbol{s}, u_i) - \alpha \log \frac{\pi_i(u_i|\boldsymbol{s})}{\pi_i(u_i|s_i^+)}\right], \forall \pi_i \neq \pi_i^*.
$$

Using the same iterative argument as in the proof of Lemma 2, we get $Q_{\text{jt}}^{\pi_{\text{jt}}^*}(\boldsymbol{s}, \boldsymbol{u}) \geq Q_{\text{jt}}^{\pi_{\text{jt}}}(\boldsymbol{s}, \boldsymbol{u})$ for all $(\boldsymbol{s}, \boldsymbol{u}) \in |S| \times |U|$. That is, the state-action value of any other policy π_{jt} is lower than or equal to that of the converged policy π_{jt}^*. Therefore, π_{jt}^* is the optimal joint policy. □

A.4 Proof of Theorem 2

Theorem 2 (Convergence of Constrained Individual Policy Improvement). *The optimization problem in (10) can be solved by iterating in an alternate fashion through the following two equations:*

$$
\pi_i^m(u_i|s_i^+) = \sum_{s_i^*} \hat{\rho}(s_i^*|s_i^+)\pi_i^m(u_i|s_i^+, s_i^*)
$$

$$
\pi_i^{m+1}(u_i|s_i^+, s_i^-) = \frac{\pi_i^m(u_i|s_i^+)\exp(Q_i(\boldsymbol{s}, u_i)/\alpha)}{\sum_{u_i} \pi_i^m(u_i|s_i^+)\exp(Q_i(\boldsymbol{s}, u_i)/\alpha)},
$$

where m refers to the iteration index. Denoting the total number of iterations as M, the presented scheme converges at a rate of $O(1/M)$ to an optimal policy π_i^* for any given bounded utility function Q_i and any initial policy π_i^0.

Proof. First, we notice that for a fixed $\pi_i(u_i|s_i^+, s_i^*)$, we can have its optimal marginal as constrained in (11):

$$
\pi_i(u_i|s_i^+) = \sum_{s_i^*} \hat{\rho}(s_i^*|s_i^+)\pi_i(u_i|s_i^+, s_i^*).
$$

Then, for a fixed marginal $\pi_i(u_i|s_i^+)$, we can have the optimal $\pi_i(u_i|s_i^+, s_i^*)$ by solving (10) via standard variational calculus:

$$
\pi(u_i|s_i^+, s_i^-) = \frac{\pi(u_i|s_i^+)\exp(Q_i(\boldsymbol{s}, u_i)/\alpha)}{\sum_{u_i} \pi(u_i|s_i^+)\exp(Q_i(\boldsymbol{s}, u_i)/\alpha)}.
$$

Lastly, with the above two equations, we can apply Theorem 1 in Leibfried and Grau-Moya (2020) to finish our proof. □

B Experiment Settings and Implementation Details

B.1 Matrix Game

In the matrix game, we use a learning rate of 3×10^{-4} for all algorithms. For the algorithm that uses mutual information as the augmented reward, we set the number of Blahut–Arimoto iterations to 1. For algorithms that use mutual information and entropy as the augmented reward, we fix α as 0.5. The batch size used in the experiment is 64. Critics and polices used in the experiments consist of one hidden layer of 64 units with ELU non-linearity. For the mixer network, we use a hypernetwork similar to QMIX (Rashid et al., 2018), except no non-linearity is used. The environment and model are implemented in Python. All models are built by PyTorch and are trained via 1 Nvidia RTX 1060 GPU to conduct all the experiments. Each experiment takes roughly 1 hour.

B.2 SMAC

In StarCraft II, we use a learning rate of 5×10^{-4} for all algorithms. The structure of the critic network and the mixer network of MIPI are the same as REFIL (Iqbal et al., 2021) except no non-linearity is used in the mixer of MIPI. The number of Blahut–Arimoto iterations is set to 4 for MIPI in this experiment. The policy network of MIPI shares all layers with the critic network except the last layer of the policy network being a different fully-connected layer. The target networks will be updated once every 200 training episodes for all algorithms. The temperature parameters α and α_i are fixed as 0.03 in SZ and CSZ and fixed as 0.1 in MMM for MIPI and Entropy. For REFIL, AQMIX, and CollaQ, we use their default settings. For CollaQ, as the original implementation is based on a different SMAC environment where the entity-level observation is not available, we re-implement CollaQ with minimum changes to adapt the entity-level observation based on the framework provided in REFIL to ensure fairness of comparison. For MAPPO, as there is no published version of MAPPO for dynamic team compositions, we choose to implement MAPPO following Papoudakis et al. (2021), with additional attention modules used in the policy and the critic to handle dynamic team compositions. The environment and model are implemented in Python. All models are built by PyTorch and are trained via a mixture of 4 Nvidia A100, 4 RTX 3090, and 1 RTX 2080 TI GPUs to conduct all the experiments. Each experiment takes 6 to 32 hours depending on the algorithms and scenarios. Our implementation of MIPI is based on REFIL (Iqbal et al., 2021) with MIT license. It is worth noting that, although we assume full observability for the rigorousness of proof, the trajectory of each agent is used to replace state s for each agent as input to settle the partial observability in all SMAC experiments.

B.3 Resource Collection

In Resource Collection, we use a learning rate of 5×10^{-4} for all algorithms. The structure of the critic network and the mixer network of MIPI are the same as REFIL (Iqbal et al., 2021) except no non-linearity is used in the mixer of MIPI. The number of Blahut–Arimoto iterations is set to 1 for MIPI in this experiment. The policy network of MIPI shares all layers with the critic network except the last layer of the policy network being a different fully-connected layer. The target networks will be updated once every 200 training episodes for all algorithms. The temperature parameters α and α_i are fixed as 0.05 in Resource Collection for MIPI. For REFIL, AQMIX, and CollaQ, we use their default settings. For CollaQ, as the original implementation is based on a different SMAC environment where the entity-level observation is not available, we re-implement CollaQ with minimum changes to adapt the entity-level observation based on the framework provided in REFIL to ensure fairness of comparison. For MAPPO, as there is no published version of MAPPO for dynamic team compositions, we choose to implement MAPPO following Papoudakis et al. (2021), with additional attention modules used in the policy and the critic to handle dynamic team compositions. The environment and model are implemented in Python. All models are built by PyTorch and are trained via 4 Nvidia RTX 3090 GPUs to conduct all the experiments. Each experiment takes roughly 20 hours. Our implementation of MIPI is based on REFIL (Iqbal et al., 2021) with MIT license. It is worth noting that, although we assume full observability for the rigorousness of proof, the trajectory of each agent is used to replace state s for each agent as input to settle the partial observability in all SMAC experiments. As suggested by previous research (Liu et al., 2021; Shao et al., 2022), random sub-group partitioning does not work well in Resource Collection, therefore we choose not to use it for MIPI in this experiment.

C Training performance on SMAC

In this section, we additionally provide the learning curves of all algorithms used in Section 5.2. As we can see from Figure 3, these algorithms achieve similar training performance, except Entropy.

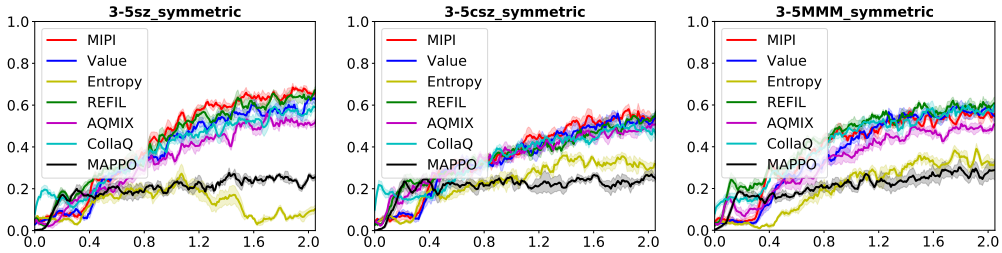

Figure 3: Learning curves of all the methods in SMAC, where the unit of x-axis is 1M timesteps and y-axis represents the win rate of each map.

D More Experiments

D.1 Resource Collection

In this section, We further evaluate MIPI on Resource Collection, which is a more challenging scenario in terms of the level of collaboration used by COPA (Liu et al., 2021). During training, the map randomly initializes 3-5 agents, and during the evaluation, we will have 6-8 agents. We plot the curve of training and evaluation performance in Figure 4. As we can see, MIPI outperforms the baselines by a large margin, which indicates that MIPI can also perform well in scenarios requiring strong collaboration.

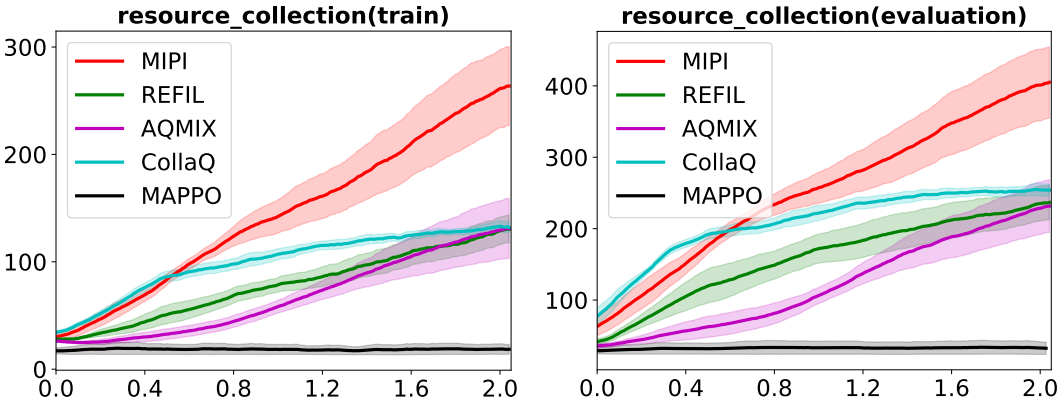

Figure 4: Learning curves of all the methods in Resource Collection, where the unit of x-axis is 1M timesteps and y-axis represents the return.

D.2 Ablation on Alpha

In this section, We further include the ablation study on the impact of alpha. We train MIPI with different alpha on 3-5 agents scenarios and evaluate their performance on 6-8 agents scenarios. We use alpha=∞ to represent the case where team-related information is completely removed (it is worth noting how this is different from actually set alpha=∞ in MIPI). Results are summarized in Table 3. As we can see, unless alpha is set unreasonably, MIPI can always achieve better generalization ability without sacrificing the training performance. It's worth noting that alpha=∞ outperforms Value here, which further indicates that reducing the dependency on team-related information promotes generalization, even when the team-related information is completely removed. However, this strategy is not widely effective and sacrifices the training performance too much in some cases (see MMM), which further leads to a decay in both training and evaluation. In contrast, our method uses alpha to control the degree of dependency on team-related information, which provides more flexibility.

Table 3: Final performance on all SMAC maps. MIPI is compared with the ablation baseline. We bold the best mean performance for each map.

Tasks \ #Agent	Alpha	Training	Evaluation		
		3-5	6	7	8
SZ	0 (Value)	0.621±0.042	0.336±0.075	0.275±0.114	0.154±0.052
	0.01	**0.672**±0.02	0.394±0.065	0.365±0.068	0.261±0.092
	0.03 (MIPI)	0.659±0.02	**0.453**±0.08	0.404±0.062	0.276±0.076
	0.05	0.643±0.02	0.447±0.062	**0.408**±0.054	**0.313**±0.069
	0.1	0.475±0.073	0.277±0.125	0.26±0.093	0.146±0.075
	0.5	0.175±0.053	0.056±0.015	0.129±0.026	0.043±0.021
	∞	0.546±0.069	0.429±0.036	0.389±0.038	0.221±0.004
CSZ	0 (Value)	0.542±0.059	0.368±0.083	0.207±0.076	0.172±0.112
	0.01	**0.592**±0.02	0.378±0.033	**0.364**±0.073	**0.304**±0.056
	0.03 (MIPI)	0.548±0.032	**0.42**±0.102	0.297±0.112	0.261±0.09
	0.05	0.506±0.046	0.417±0.092	0.223±0.094	0.192±0.091
	0.1	0.344±0.05	0.218±0.16	0.113±0.079	0.098±0.086
	∞	0.506±0.076	0.368±0.064	0.309±0.056	0.27±0.07
MMM	0 (Value)	0.545±0.048	0.505±0.058	0.391±0.083	0.319±0.105
	0.05	**0.59**±0.008	**0.59**±0.053	**0.526**±0.055	0.426±0.152
	0.1 (MIPI)	0.548±0.023	0.495±0.054	0.447±0.041	**0.467**±0.067
	0.5	0.277±0.042	0.158±0.094	0.18±0.105	0.139±0.056
	∞	0.396±0.07	0.432±0.018	0.383±0.041	0.315±0.061

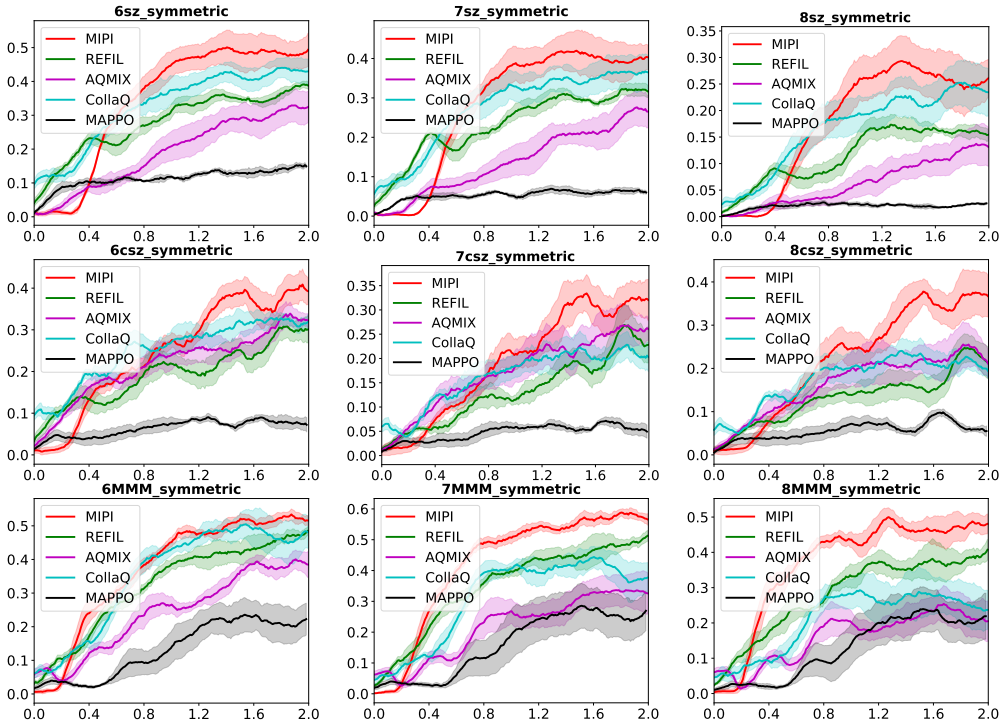

Figure 5: Learning curves of all the methods in SMAC, where the unit of x-axis is 1M timesteps and y-axis represents the win rate of each map.

D.3 Performance on Higher Level Driver

When working on the follow-up project of this paper, we noticed that REFIL can achieve better generalization results in MMM with a higher-level NVIDIA driver without any code-level change.

The results are shown in Figure 5, where all algorithms are trained in a single platform that REFIL achieves better results. As we can see, although REFIL achieves better generalization results in some cases, MIPI can still outperform these baselines in terms of both speed and final performance by properly setting α and α_i (0.01 for sz, 0.015 for csz and 0.05 for MMM).