APPENDIX A. CONTRASTIVE ROLE REPRESENTATION LEARNING

In this section, we give the proof of Theorem 1 in the text based on introducing a lemma as follows.

Lemma 1. *Given a role from the distribution $M \sim P(M)$, let $e = f_\phi(\sum_t(o^t, a^{t-1}))$ as the agent embedding generated by role M, and $z \sim f_\theta(z|e)$, where $\sum_t(o^t, a^{t-1})$ is the agent's local trajectory following a given policy. Then, we have*

$$\frac{p(M|z)}{p(M)} = \mathbb{E}_e\left[\frac{p(z|e)}{p(z)}\right]. \tag{9}$$

Proof.

$$\begin{aligned}\frac{p(M|z)}{p(M)} &= \frac{p(z|M)}{p(z)} \\ &= \int_e \frac{p(e|M)p(z|e)}{p(z)}\mathrm{d}e \\ &= \mathbb{E}_e\left[\frac{p(z|e)}{p(z)}\right].\end{aligned} \tag{10}$$

The proof is completed. □

Theorem 1. *Let $\mathcal{M}$ denote a set of roles following the role distribution $P(M)$, and $|\mathcal{M}| = K$. $M \in \mathcal{M}$ is a given role. Let $e = f_\phi(\sum_t(o^t, a^{t-1}))$, $z \sim f_\theta(z|e)$, and $h(e, z) = \frac{p(z|e)}{p(z)}$, where $\sum_t(o^t, a^{t-1})$ is the agent's local trajectory following a given policy. For any role $M^* \in \mathcal{M}$, let e^* denote the agent embedding generated by the role M^*, then we have*

$$I(z; M) \geq \log K + \mathbb{E}_{\mathcal{M},z,e}\left[\log \frac{h(e, z)}{\sum_{M^*\in\mathcal{M}} h(e^*, z)}\right]. \tag{2}$$

Proof. Using Lemma A.1 and Jensen's inequality, we have

$$\begin{aligned}\mathbb{E}_{\mathcal{M},z,e}\left[\log \frac{h(e,z)}{\sum_{M^*\in\mathcal{M}} h(e^*,z)}\right] &= \mathbb{E}_{\mathcal{M},z,e}\left[\log \frac{\frac{p(z|e)}{p(z)}}{\frac{p(z|e)}{p(z)} + \sum_{M^*\in\mathcal{M}\setminus M}\frac{p(z|e^*)}{p(z)}}\right] \\ &= \mathbb{E}_{\mathcal{M},z,e}\left[-\log\left(1 + \frac{p(z)}{p(z|e)}\sum_{M^*\in\mathcal{M}\setminus M}\frac{p(z|e^*)}{p(z)}\right)\right] \\ &\approx \mathbb{E}_{\mathcal{M},z,e}\left[-\log\left(1 + \frac{p(z)}{p(z|e)}(K-1)\mathbb{E}_{M^*\in\mathcal{M}\setminus M}\left[\frac{p(z|e^*)}{p(z)}\right]\right)\right] \\ &= \mathbb{E}_{\mathcal{M},z,e}\left[-\log\left(1 + \frac{p(z)}{p(z|e)}(K-1)\right)\right] \\ &= \mathbb{E}_{\mathcal{M},z,e}\left[\log\left(\frac{1}{1 + \frac{p(z)}{p(z|e)}(K-1)}\right)\right] \\ &\leq \mathbb{E}_{\mathcal{M},z,e}\left[\log\left(\frac{1}{\frac{p(z)}{p(z|e)}K}\right)\right] \\ &\leq \mathbb{E}_{\mathcal{M},z}\left[\log \mathbb{E}_e\left[\frac{p(z|e)}{p(z)}\right]\right] - \log K \\ &= \mathbb{E}_{\mathcal{M},z}\left[\log\frac{p(M|z)}{p(M)}\right] - \log K \\ &= I(z; M) - \log K.\end{aligned} \tag{11}$$

Thus, we complete the proof. □

APPENDIX B. IMPLEMENTATION DETAILS AND EXTENDED EXPERIMENTS OF QMIX-BASED ACORM

Based on the implementations in Section 2, we summarize the brief procedure of ACORM based on QMIX in Algorithm 1.

Algorithm 1: ACORM based on QMIX

Input: ϕ: agent's trajectory encoder
θ: role encoder
K: number of clusters
T_{cl}: time interval for updating contrastive loss
n: number of agents
$\mathcal{B}$: replay buffer
T: time horizon of a learning episode

Initialize all network parameters
Initialize the replay buffer $\mathcal{B}$ for storing agent trajectories
for *episode* $= 1, 2, ...$ **do**
  Initialize history agent embedding e_i^0, and action vector a_i^0 for each agent
  for $t = 1, 2, ..., T$ **do**
    Obtain each agent's partial observation the $\{o_i^t\}_{i=1}^n$ and global state $\boldsymbol{s}^t$
    for *agent* $i = 1, 2, ..., n$ **do**
      Calculate the agent embedding $e_i^t = f_\phi(o_i^t, a_i^{t-1}, e_i^{t-1})$
      Calculate the role representation $z_i^t = f_\theta(e_i^t)$
      Select the local action a_i^t according to individual Q-function $Q_i(e_i, a_i^t)$
    end
    Execute joint action $\boldsymbol{a}^t = [a_1^t, a_2^t, ..., a_n^t]^\top$, and obtain global reward r^t
  end
  Store the trajectory to $\mathcal{B}$
  Sample a batch of trajectories from $\mathcal{B}$
  if *episode* $\bmod T_{cl} == 0$ **then**
    Partition agent embeddings $\{e_i^t\}_{i=1}^n$ into K clusters $\{C_j\}_{j=1}^K$ using K-means
    for *agent* $i = 1, 2, ..., n$ **do**
      Construct positive keys $\{z_{i'}\}_{i' \in C_j}$ and negative keys $\{z_{i^*}\}_{i^* \notin C_j}$ for query $z_i, i \in C_j$
    end
    Update contrastive learning loss according to Eq. (4)
    Update momentum role encoder according to Eq. (5)
  Calculate attention output $\boldsymbol{\tau}_{\text{mha}}$ via prompting the global state to attend to role representations $\{z_i\}_{i=1}^n$ in Eqs. (6)-(8)
  Concatenate $\boldsymbol{\tau}_{\text{mha}}$ with state embedding $\boldsymbol{\tau}$ to form the input to the mixing network
  Update the parameters of individual Q-network and the mixing network
end

In this paper, we use simple network structures for the trajectory encoder, the role encoder, and the attention mechanism. Specifically, the trajectory encoder contains a fully-connected multi-layer perceptron (MLP) and a GRU network with ReLU as the activation function, and encodes agent's trajectory into a 128-dimensional embedding vector. The role encoder is a fully-connected MLP that transforms the 128-dimensional agent embedding into a 64-dimensional role representation. The setting of the mixing network is kept as the same as that of QMIX (Rashid et al., 2020), where the architecture contains two 32-dimensional hidden layers with ReLU activation. Table 1 shows the details of network structures.

For all tested algorithms, we use the Adam optimizer with a learning rate of 6e-4. For exploration, we use the ϵ-greedy strategy with ϵ annealed linearly from 1.0 to 0.02 over $80k$ time steps and kept constant for the rest of the training. Every time an entire episode from online interaction is collected and stored in the buffer, the Q-networks are updated using a batch of 32 episodes sampled from the replay buffer with a capacity of 5000 state transitions. The target Q-network is updated using a soft update strategy with momentum coefficient 0.005. The contrastive learning loss is jointly trained every 100 steps of the Q-network updates. The decentralized policy is evaluated every $5k$ update steps with 32 episodes generated. For all domains, the number of clusters in ACORM is set to $K = 3$. Appendix D provides an analysis on this hyperparameter, and the experiments show that the performance of ACORM is not significantly affected by the value of K. The details of hyperparameters can be found in Table 2.

Table 1: The network configurations used for ACORM based on QMIX.

Network Configurations	**Value**	**Network Configurations**	**Value**
role representation dim	64	hypernetwork hidden dim	32
agent embedding dim	128	hypernetwork layers num	2
state embedding dim	64	type of optimizer	Adam
attention output dim	64	activation function	ReLU
attention head num	4	add last action	True
attention embedding dim	128		

Table 2: Hyperparameters used for ACORM based on QMIX.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
buffer size	5000	start epsilon ϵ_s	1.0
batch size	32	finish epsilon ϵ_f	0.02
learning rate	6×10^{-4}	ϵ decay steps	80000
use learning rate decay	True	evaluate interval	5000
contrastive learning rate	8×10^{-4}	evaluate times	32
momentum coefficient β	0.005	target update interval	200
update contrastive loss interval T_{cl}	100	discount factor γ	0.99
cluster num	3		

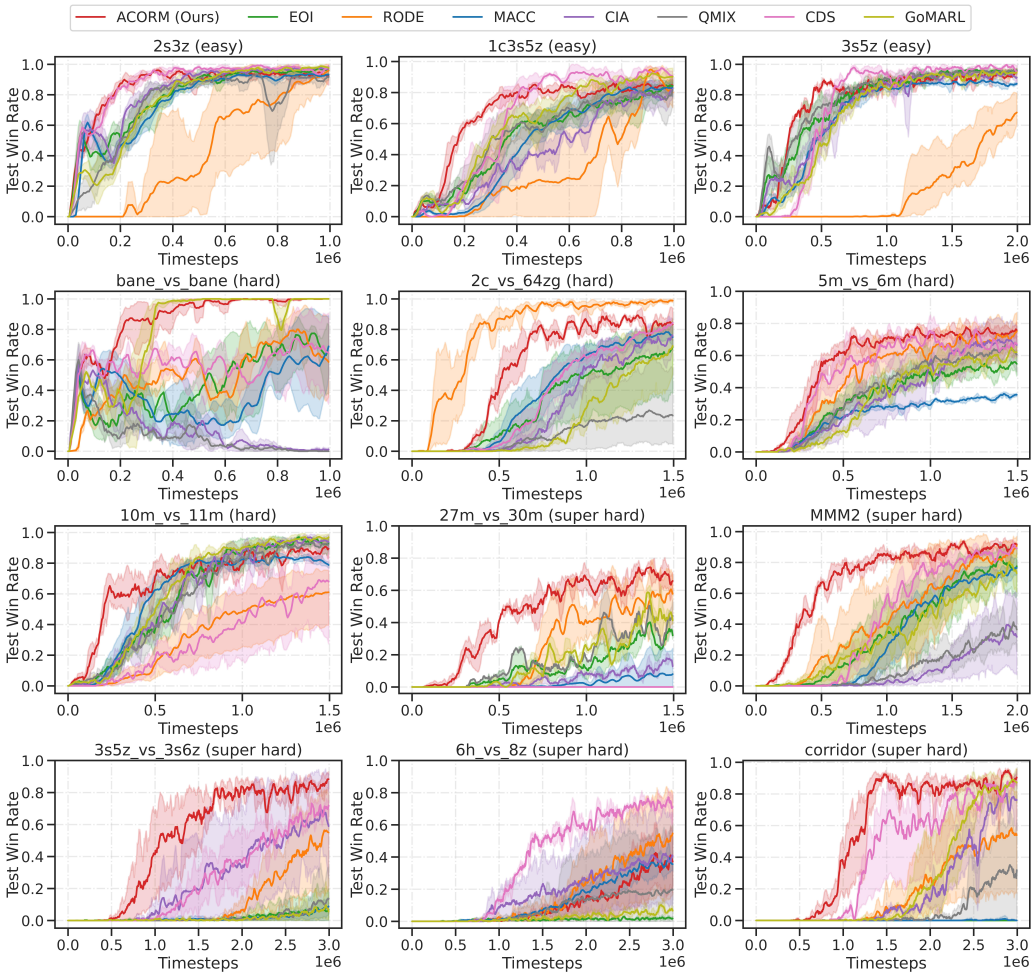

Figure 6: Extended performance comparison between ACORM and baselines on 12 SMAC maps.

Fig. 6 presents the extended performance of ACORM on 12 SMAC maps. Obviously, the observations and conclusions from the extended performance are kept consistent with those in Sec. 3.1 of the main paper.

APPENDIX C. EXTENDED EXPERIMENTS ON GOOGLE RESEARCH FOOTBALL

In addition to SMAC environments, we also benchmark our approach on three challenging Google research football (GRF) offensive scenarios as

- `academy_3_vs_1_with_keeper`: Three of our players try to score from the edge of the box, one on each side, and the other at the center. Initially, the player at the center has the ball and is facing the defender. There is an opponent keeper.
- `academy_counterattack_hard`: 4 versus 2 counter-attack with keeper; all the remaining players of both teams run back towards the ball.
- `academy_run_to_score_with_keeper`: Our player starts in the middle of the field with the ball, and needs to score against a keeper. Five opponent players chase ours from behind.

In GRF tasks, agents need to coordinate timing and positions for organizing offense to seize fleeting opportunities, and only scoring leads to rewards. In our experiments, we control left-side players except the goalkeeper. The right-side players are rule-based bots controlled by the game engine. Agents have a discrete action space of 19, including moving in eight directions, sliding, shooting, and passing. The observation contains the positions and moving directions of the ego-agent, other agents, and the ball. The z-coordinate of the ball is also included. Environmental reward only occurs at the end of the game. They will get $+100$ if they win, else get -1.

Fig. 7 presents the performance comparison between ACORM and baselines on three challenging GRF scenarios. It can be observed that QMIX obtains poor performance, since the tasks in GRF are more challenging than in the SMAC benchmark. In contrast, ACORM gains a significantly improved increase in the test win rate, especially in the first 1M training timesteps, which successfully demonstrates the effectiveness of our method evaluated on GRF benchmarks. Together with evaluations on SMAC domains, the same conclusion can still be drawn that ACORM outperforms baseline methods by a larger margin on harder tasks that demand a significantly higher degree of behavior diversity and coordination. In summary, experimental results on extended GRF environments are generally consistent with those on the SMAC benchmark. .

Due to the very limited time for rebuttal revision, we only compare ACORM to QMIX and CDS, as other baselines (RODE, EOI, MACC, CIA) are not evaluated on GRF in their original papers. Also, we just show the performance within 2M training steps. We hope that the extended experimental evaluation could demonstrate adequate persuasiveness of our method. We are rushing to conduct the evaluation of baseline methods on GRF scenarios with more training timesteps, and trying to update them before the rebuttal deadline.

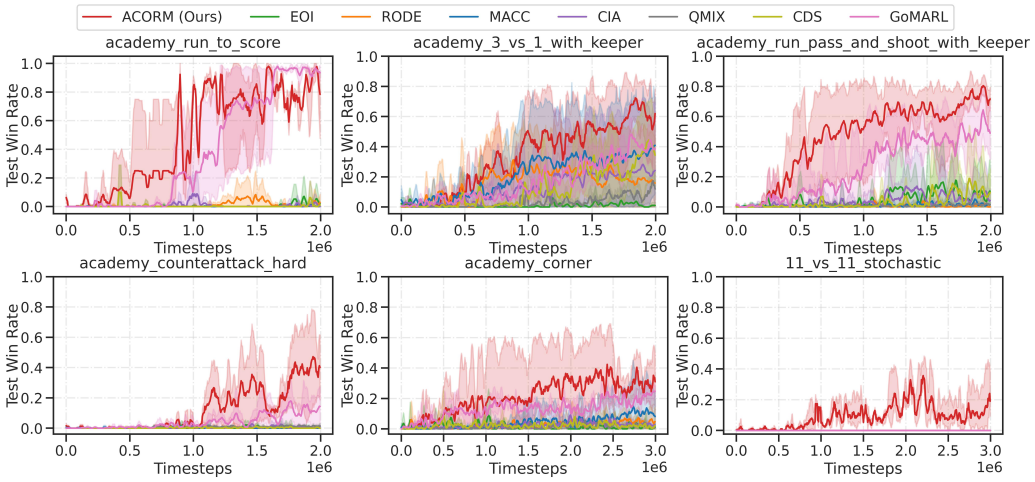

Figure 7: Performance comparison between ACORM and baselines on three GRF scenarios.

APPENDIX D. ACORM BASED ON MAPPO

In addition, we realize ACORM on top of the MAPPO algorithm, as show in Fig. 8. Most of the network structure is kept the same as QMIX-based ACORM, including the agent embedding, the role encoder, and the attention mechanism. To align with MAPPO that uses an actor-critic architecture, we input both the agent's observation and the augmented global state $\tilde{s}$ (obtained from the attention mechanism in Fig. 8(e)) into the critic, as shown in Fig. 8(a). Tables 3 and 4 present the detailed network structure and experimental hyperparameters, respectively.

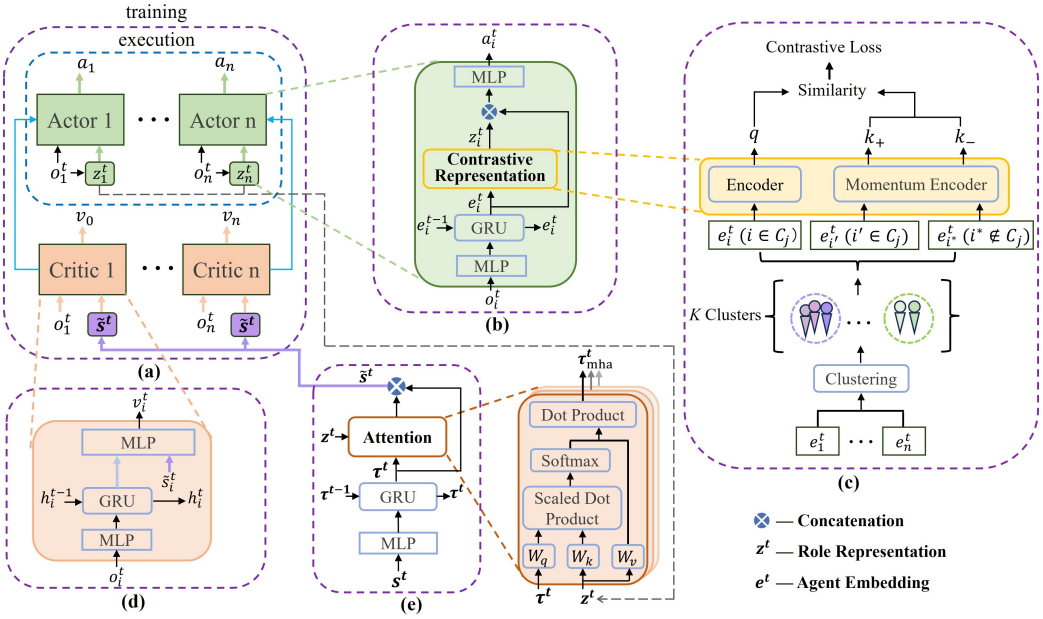

Figure 8: The ACORM framework based on MAPPO. (a) The overall architecture. (b) The shared actor network structure for each agent, where the role representation is extracted from agent's trajectory. (c) The detail of learning role representations via contrasting learning. (d) The shared critic network structure for each agent. (e) The attention module that incorporates learned role representations into value decomposition.

Table 3: The network configurations used for ACORM based on MAPPO.

Network Configurations	Value	Network Configurations	Value
role representation dim	64	attention output dim	64
agent embedding dim	128	attention head num	4
state embedding dim	64	attention embedding dim	128
critic RNN hidden dim	64	type of optimizer	Adam
add agent ID	False	activation	ReLU

Table 4: Hyperparameters used for ACORM based on MAPPO.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
batch size	32	entropy coefficient	0.02
mini batch size	32	cluster num	3
actor learning rate	6×10^{-4}	discount factor γ	0.99
critic learning rate	8×10^{-4}	momentum coefficient β	0.005
contrastive learning rate	8×10^{-4}	evaluate interval	5000
update contrastive loss interval T_{cl}	32	evaluate times	32
clip	0.2	use advantage normalization	True
GAE lambda λ	0.95	use learning rate decay	False
K epochs	5		

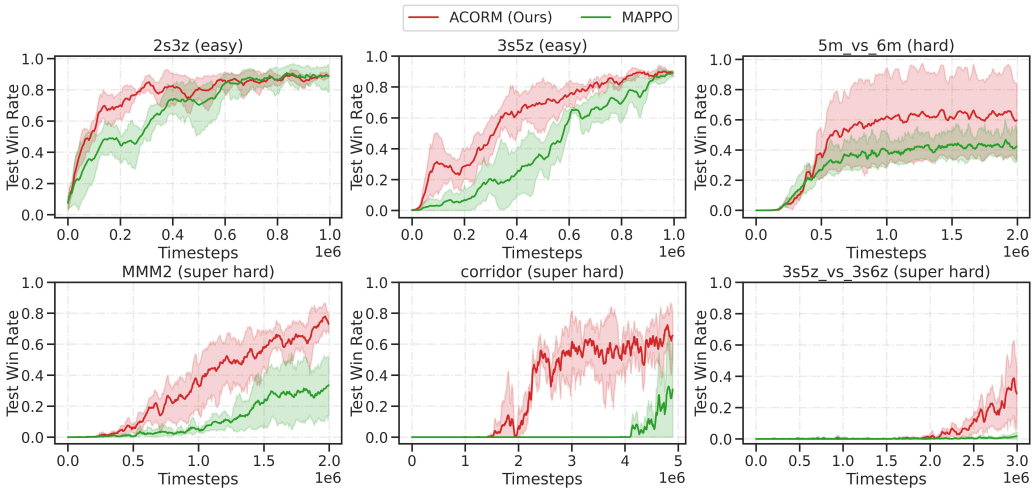

Figure 9: Performance comparison between MAPPO-based ACORM and the MAPPO baseline on representative maps, including two easy levels (`2s3z`, `3s5z`), one hard level (`5m_vs_6m`), and three super hard levels (`MMM2`, `corridor`, `3s5z_vs_3s6z`).

Figure 9 presents the performance of MAPPO-based ACORM on six representative tasks: `2s3z`, `3s5z`, `5m_vs_6m`, `MMM2`, `corridor`, and `3s5z_vs_3s6z`. It can be observed that ACORM achieves a significant performance improvement over MAPPO. Akin to the QMIX-based version, ACORM outperforms MAPPO by the largest margin on super hard maps that demand a significantly higher degree of behavior diversity and coordination. Again, experimental results demonstrate ACORM's superiority of learning efficiency in complex multi-agent domains.

APPENDIX E. HYPERPARAMETER ANALYSIS ON THE NUMBER OF CLUSTERS

We test the influence of the number of clusters K on ACROM's performance. Fig. 10 shows the performance of ACORM with varying values of $K = 2, 3, 4, 5$. Generally speaking, ACORM obtains similar performance across different values of K. It demonstrates that ACORM achieves good learning stability and robustness as the performance is insensitive to the pre-defined number of role clusters. An outlier case is observed in the `5m_vs_6m` map where the ACORM's performance drops a little when $K = 5$. This is likely because there are only 5 agents in `5m_vs_6m`. When $K = 5$, each agent represents a distinct role cluster. It forces the strategies of each agent to diverge, which might not be conductive to realize effective team composition across agents.

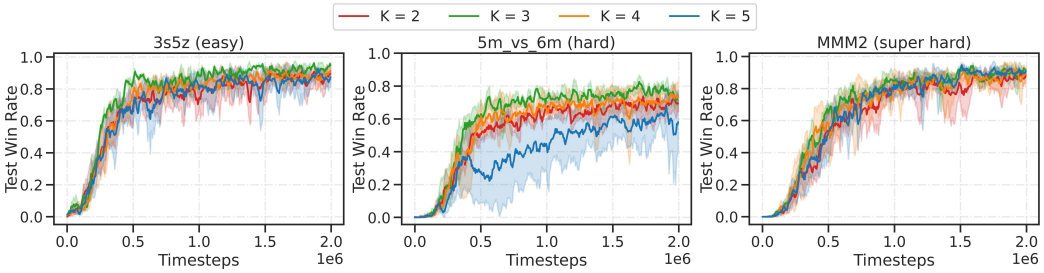

Figure 10: Hyperparameter analysis on the number of clusters K in negative pairs generation.

To illustrate why ACORM performs well across different values of K, we show visualizations from different learned ACORM policies with $K = 3, 4, 5$ in Fig. 11. When the clustering granularity is coarse as $K = 3$ in Fig. 11(a), even within the same cluster, ACORM can still learn meaningful role representations with distinguishable patterns. Agent embeddings of Marines $\{2, 3, 4, 5, 6, 7, 8\}$ are crowded together with limited discrimination. Via contrastive learning, the obtained role representations exhibit an interesting self-organized structure where Marines $\{2, 5, 6\}$ and $\{3, 4, 7, 8\}$ implicitly form two distinctive sub-groups. It can be observed from the rendering scene that Marines $\{2, 5, 6\}$ and $\{3, 4, 7, 8\}$ are all at an attacking stage, while the former sub-group is in lower health than the latter. On the other hand, with a fine granularity of $K = 5$ in Fig. 11(c), the contrastive learning module transforms clustered agent embeddings into more discriminative role representations. For example, while Marines $2, 3, 5, 6, 8$, and $4, 7$ form three clusters, their role representations are still closer to each other and farther from Marauders $\{0, 1\}$ and Medivac $\{9\}$, since they are the same type of agents with similar behavior patterns. In summary, it again demonstrates that ACORM learns meaningful role representations and achieves effective and robust team composition.

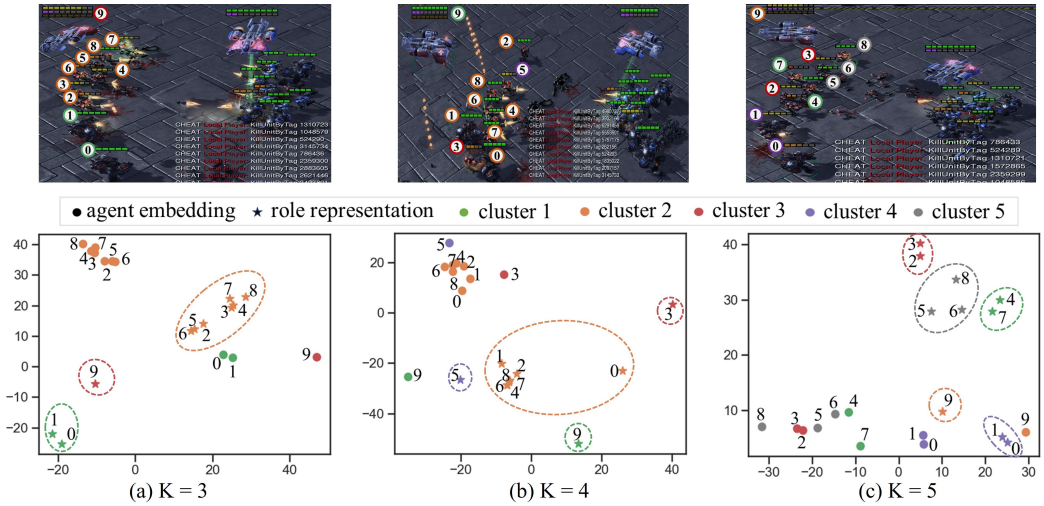

Figure 11: Visualization with different number of clusters K in negative pairs generation.

APPENDIX F. MORE ABLATION RESULTS

Figure 12 presents the full ablation results of ACORM on all six representative maps. It again demonstrates the significance of both the contrastive learning module and the attention mechanism for ACORM's performance. A noteworthy point is that both ACORM_w/o_CL and ACORM_w/o_MHA gain remarkable performance improvement by the largest margin on super hard maps, which further validate ACORM's advantages of promoting diversified behaviors and skillful coordination in complex multi-agent tasks. Another interesting observation is that when omitting the contrastive learning module, there is a notable increase in the variance of learning curves. It can be interpreted as an evidence that contrastive learning helps training more robust role representations and enhances learning stability.

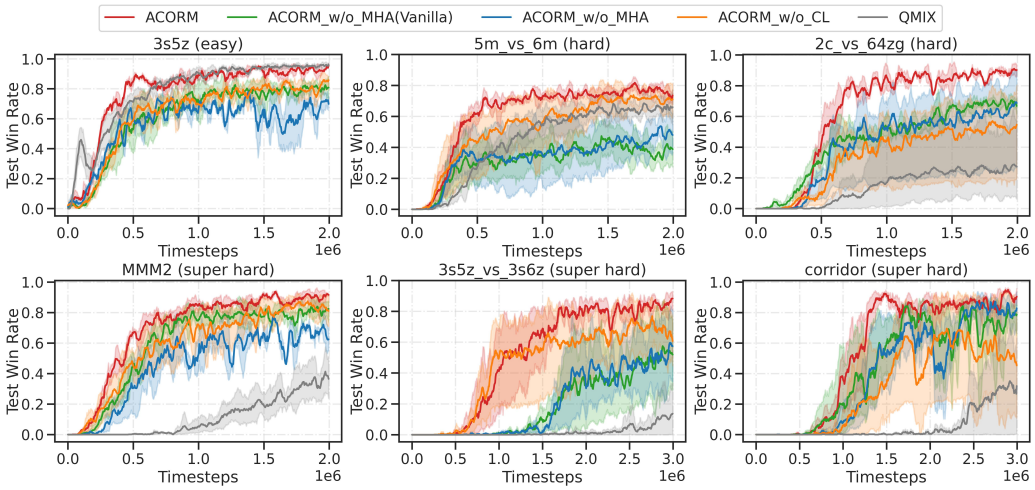

Figure 12: Full ablation results on ACORM. ACORM_w/o_CL means removing contrastive learning, ACORM_w/o_MHA represents excluding attention, ACORM_w/o_MHA (Vanilla) represents excluding attention and state encoding, and QMIX corresponds to removing all components.

APPENDIX G. MORE INSIGHTS ON CONTRASTIVE REPRESENTATION LEARNING

Based on Section 3.2, here we provide more insights on the effectiveness of learned role representations. Indeed, one reason for ACORM's significant improvement on learning efficiency comes from its capability of facilitating implicit knowledge transfer across similar agents throughout the entire learning process. For example, Marines $\{2, 3, 4, 5, 6, 7\}$ form a group at $t = 1$, Marauders $\{0, 1\}$ and Marines $\{2, 4, 6, 7, 8\}$ form a group at $t = 12$, and Marines $\{2, 3, 4, 6, 7\}$ form a group at $t = 40$. It can be observed from the rendering scenes that these three groups are all responsible for attacking enemies. At different time steps, agents in the attacking group can share similar role representations to promote knowledge transfer, even if they belong to heterogeneous agent types. This implicit transfer across agents and across timesteps can significantly increase the exploration efficiency of agents.

Another highlight of ACORM is the promotion of behavior heterogeneity, even if agents have the same innate characteristics. For example, while Marines $\{2, 3, 4, 5, 6, 7, 8\}$ belong to the same agent type, they are distributed to different groups with heterogeneous roles as: 1) at $t = 12$, Marines $\{2, 4, 6, 7, 8\}$ with role of attacking, and Marines $\{3\}$ and $\{3\}$ with role of the wounded; and 2) at $t = 40$, Marines $\{2, 3, 4, 6, 7\}$ with role of attacking, and Marines $\{5\}$ and $\{8\}$ with role of the dead. In general, even though some information is lost during dimension reduction using t-SNE, it is evident that our role representation still manages to exhibit such remarkable results.

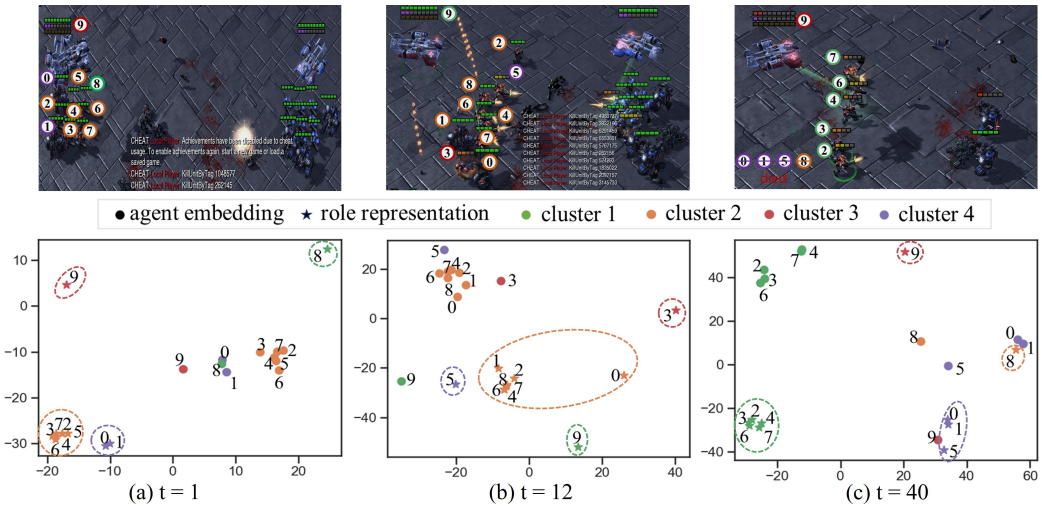

Figure 13: Example rendering scenes at three time steps in an evaluation trajectory generated by the trained ACORM policy on MMM2. The upper row shows screenshots of combat scenarios that contain the information of positions, health points, shield points, states of ally and enemy units, etc. The lower row visualizes the corresponding agent embeddings (denoted with bullets '●') and role representations (denoted with stars '★') by projecting these vectors into 2D space via t-SNE for qualitative analysis, where agents within the same cluster are depicted using the same color.