HiMacMic: Hierarchical Multi-Agent Deep Reinforcement Learning with Dynamic Asynchronous Macro Strategy

Hancheng Zhang
Beijing Inst. of Tech.
Beijing, China
hancheng@bit.edu.cn

Guozheng Li
Beijing Inst. of Tech.
Beijing, China
guozheng.li@bit.edu.cn

Chi Harold Liu
Beijing Inst. of Tech.
Beijing, China
chiliu@bit.edu.cn

Guoren Wang
Beijing Inst. of Tech.
Beijing, China
wanggr@bit.edu.cn

Jian Tang
Midea Group
Beijing, China
tangjian22@midea.com

ABSTRACT

Multi-agent deep reinforcement learning (MADRL) has been widely used in many scenarios such as robotics and game AI. However, existing methods mainly focus on the optimization of agents' micro policies without considering the macro strategy. As a result, they cannot perform well in complex or sparse reward scenarios like the StarCraft Multi-Agent Challenge (SMAC) and Google Research Football (GRF). To this end, we propose a hierarchical MADRL framework called "HiMacMic" with dynamic asynchronous macro strategy. Spatially, HiMacMic determines a critical position by using a positional heat map. Temporally, the macro strategy dynamically decides its deadline and updates it asynchronously among agents. We validate HiMacMic in four widely used benchmarks, namely: Overcooked, GRF, SMAC and SMAC-v2 with nine chosen scenarios. Results show that HiMacMic not only converges faster and achieves higher results than ten existing approaches, but also shows its adaptability to different environment settings.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**.

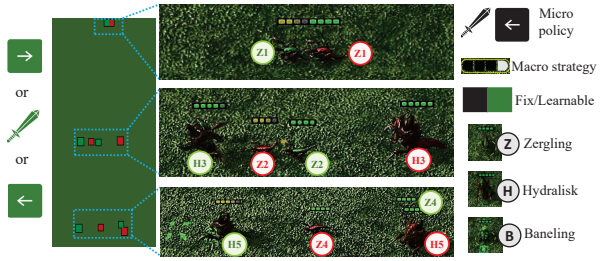
KEYWORDS

Multi-agent deep reinforcement learning, macro strategy

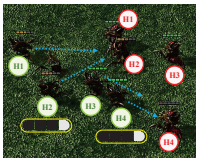
ACM Reference Format:
Hancheng Zhang, Guozheng Li, Chi Harold Liu, Guoren Wang, and Jian Tang. 2023. HiMacMic: Hierarchical Multi-Agent Deep Reinforcement Learning with Dynamic Asynchronous Macro Strategy. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599379

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

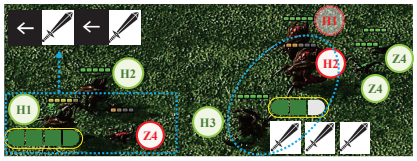
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599379



(a) Micro policy only without any macro strategy.



(b) Macro strategy with synchronous decision making.



(c) Macro strategy with artificial and fixed rules.

Figure 1: Motivation of this paper by examples in SMAC-v2 zerg_5_vs_5 scenario.

1 INTRODUCTION

Multi-agent deep reinforcement learning (MADRL) has shown potentials in various applications such as life and medical sciences [41], robotics [22, 23, 30, 42, 49, 51], and game AI [12, 13, 25, 27, 31]. In order to solve the challenge of optimal cooperations among agents, existing solutions [16, 21, 26, 33, 34, 37, 43] mainly used centralized training decentralized execution (CTDE [18]) pattern by assigning a global reward [28]. However, these approaches primarily focus on micro policies (i.e., actions at each time step), lacking decision-making at the macro strategic level. This lead to possible discoordination among agents or suboptimal policies in challenging scenarios [14, 36, 47], such as 6h_vs_8z in SMAC [36], zerg_5_vs_5 in SMAC-v2 [9], academy_3_vs_1_with_keeper (or simply keeper) or academy_counterattack_hard (or simply hard) in GRF [20].

Take the zerg_5_vs_5 scenario in SMAC-v2 as an example, as shown in Figure 1(a), if agents only consider the micro-operations, then they prefer to fight against their own battle individually, by selecting the first enemy found in their respective field of vision and attack range without integrity. However, a good high-level macro

strategy might be gathering together to fire to destroy enemy units, rather than in three disconnected battle zones. Despite the fact that curiosity-based auxiliary rewards [7] and goal-conditioned hierarchical designs [3, 11] improved micro policy and achieved the task-level understanding in single-agent settings, they cannot be well applied to multiple agents directly. This is because that, on one hand, if multiple agents share the same macro strategy, they need to be trained and executed in a centralized manner, which may not be feasible in practice; on the other hand, if each agent adopts the macro strategy in a decentralized training and decentralized execution (DTDE) pattern without centralized coordination, agents will face the challenge brought by the impact of the unstable environment during agent interactions.

Along the direction of CTDE, existing works relied on expert knowledge in macro strategy design. Methods [17, 45, 47] stipulated that agents share the same manually defined macro strategy duration. In other words, they idealized the macro strategy among multiple agents into a synchronous decision-making process. However, it is impractical by enforcing agents to wait for others to terminate and communicate with each other. As shown in Figure 1(b), Hydralisk (denoted as H) H3 and H4 will soon destroy enemy H4, but H1 and H2 are still in the process of attacking enemy H2. Given the fact that the macro strategies of multiple agents are synchronous, there will be contradictions and may lead to sub-optimal results. In practice, cooperation of multiple agents should be asynchronous. In this case, units H3 and H4 better choose a new strategy to attack enemies H1 and H3 without waiting for their alliance H1 and H2. Furthermore, although [46] allowed agents to select macro strategy asynchronously, it fixed it as an established rule. As shown in Figure 1(c), it designed macro strategies with different time lengths of a sequence of actions. The agent independently and asynchronously selects the original actions and manually defines macro strategies. Inspired by human players, when facing a combat-attacking enemy Zergling (denoted as Z) Z4 and H1, a long-range unit, can use the defined kiting strategy. That is, to attack and move back to pull more space alternatively. However, a better strategy might be to directly destroy the enemy units with a small amount of blood and then focusing on the fire to destroy the remaining units on the right hand side of the enemy, rather than mechanically implementing the established strategy which results in unnecessary movement and time loss. Furthermore, it is not convenient to adapt established strategy to multiple types of agents (such as zergling, hydralisk, baneling with random proportion) and flexibly applied to different scenarios. Therefore, there lacks of a method that can guide micro policies based on asynchronous macro strategy, where the latter can dynamically optimize and update itself.

In this paper, we propose a hierarchical MADRL framework called "HiMacMic", and its contributions are:

- We propose a hierarchical MADRL architecture called HiMacMic, where agents are able to decide when and where to perform the specific micro policy, navigated by the spatiotemporal macro strategy.
- We propose a macro strategy deadline generation approach by controller networks, to achieve dynamic and asynchronous high-level strategy guidance of all agents and is adaptable to different tasks.
- We propose a positional heat map from successful past experiences to train the macro strategy, and an intrinsic reward mechanism by measuring the Manhattan distance between predicted positions from macro strategy and agents' actual positions to guide micro policy movement, to achieve higher sample efficiency and cooperation in complex and sparse reward scenarios.
- We evaluate HiMacMic in four common benchmarks, including Overcooked, GRF, SMAC and SMAC-v2. Empirical results demonstrate that HiMacMic achieves both faster convergence and better final results over 10 state-of-the-art MADRL baselines. A complete set of ablation studies as well as trajectory visualization are also given.

2 RELATED WORK

2.1 MARL with Only Micro Policy

Multi-agent reinforcement learning (MARL) consists of cooperative [12, 48], competitive [4], and mixed settings [24, 32]. The main challenge lies in assigning credit between the entire team and individual agents to learn micro policies. Early attempts [35] at value function factorization required expert knowledge for suitable per-agent team reward decomposition. Furthermore, some methods [14, 42] adopted DTDE pattern and directly regarded other agents as part of the environment that adopts a single agent algorithm to train all agents together. Due to the lack of global information, they cannot effectively address the environmental instability issue caused by agent-environment interaction, leading to subpar performance.

Following the CTDE paradigm [28], MADRL has made remarkable progress [36] recently. For example, VDN [38] decomposed the joint Q-value function into a sum of local utility functions and used it to select action greedily. QMIX [34] used general monotonic functions to decompose the joint Q-value function into a sum of local value functions. QTRAN [37] proposed IGM condition and mapped the joint value function to a new function that is decomposed to each agent to reduce the strong constraint of QMIX on monotonicity. Weighted-QMIX [33] put forward two weighted mixing methods, centrally-weighted and optimistically-weighted, which achieved better results in fitting IGM conditions, especially in the grid world environment. QPLEX [43] used duplex dueling network to avoid the inaccurate fitting of monotonic network at the peak reward. FACMAC [30] used a centralized policy gradient estimator without monotonicity constraint, to improve the micro policy in continuous action space. However, these methods focused on micro policy actions in each time step to seek cooperations with dense rewards, and lacked consideration of cooperative decisions from the perspective of macro strategy.

2.2 Hierarchical MARL with Macro Strategy

A number of approaches to single-agent hierarchical reinforcement learning have been suggested, including goal-reach approach (e.g., H-DQN [19]), multi-level control (e.g., Feudal RL [8]), options framework [39], skill based method [10], etc. These approaches are conducive to the decomposition of complex problems from a macro strategy perspective, guiding the micro policy to complete actions according to certain strategies.

Migrated directly from single agent approaches, FMH [2] applied Feudal RL [8] to multi-agent environments. HSD [47] introduced a single agent skill-based method [10] through supervised learning and thereby promoting skill discovery. RODE [45] used a clustering method to generate a fixed number of invariant action space groups as roles assigned to agents, and then selected actions according to roles. MASER [17] generated subgoals from experience replay buffer based on Q-function estimate, and used subgoals to guide agents complete tasks hierarchically. However, these methods required that the intervals to make macro strategies between all agents are synchronized and also fixed based on human experiences. That is, agents need to wait for others to complete their macro strategies, and interval to make the next macro strategy is fixed that cannot be modified dynamically during training process.

On the other hand, Mac-CAC [46] also emphasized that synchronizing decisions across multiple agents in realistic settings is problematic. Ideally in real life, agents should train and execute asynchronously. In this way, macro strategy temporally extends micro policy that can take different amounts of time based on the environment situation. Unfortunately, they required the macro strategy as a fixed path planning algorithm that cannot be modified, for the reason of current policy gradient methods are not applicable in asynchronous settings. In summary, existing macro strategies cannot be flexibly and automatically tuned according to the dynamic change of the underlying environment's spatiotemporal information, thus resulting in unsatisfactory overall performance.

3 PRELIMINARIES

We consider a fully cooperative task with N agents denoted as $\mathcal{N} \triangleq \{1, 2, \ldots, n, \ldots, N\}$ under a Dec-POMDP setting [28], as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{Z}, r, P, O, N, \gamma >$. $\mathcal{S}$ is the finite set of global states, $\mathcal{A}$ is the set of joint action space, $\mathcal{Z}$ is the set of joint observations, r is the joint reward function shared among all agents, P is the transition function specifying the state transition probabilities, O is the observation function, and γ is the discount factor.

Since the joint action and observation space grows exponentially as the number of agents increases, it is challenging to use joint action-value function to train an MADRL method directly. Recent studies [37] tried to find the factorizable cooperation task according to IGM condition, which showed promising results in complex environment with many agents and large state-action space. As shown in Eqn. (1), each agent selects a greedy action according to their individual action-value functions in a decentralized fashion, making it possible to use the unique global reward to optimize the multi-agent cooperation problem.

$$\arg\max_{\boldsymbol{a}_t} Q^{\text{ttl}}(\boldsymbol{\tau}_t, \boldsymbol{a}_t) = \begin{pmatrix} \arg\max_{a_t^1} Q^1\left(\tau_t^1, a_t^1\right) \\ \vdots \\ \arg\max_{a_t^N} Q^N\left(\tau_t^N, a_t^N\right) \end{pmatrix}, \tag{1}$$

where t is the time step in an episode $i \triangleq \{1, \ldots, t, \ldots, T\}$ and $\boldsymbol{\tau}_t$ is joint action observation histories, and $\boldsymbol{a}_t$ is the joint action for all agents at time step t.

In the CTDE regime, the mixing network is introduced to merge all individual Q-values into Q^{ttl} as:

$$Q^{\text{ttl}}(\boldsymbol{\tau}_t, \boldsymbol{a}_t, s_t) = f^{\text{mix}}\left(Q^n\left(\tau_t^n, a_t^n\right)|_{n=1}^N, s_t\right), \tag{2}$$

where f^{mix} represents different mixing functions. QMIX [34] proposed a monotonicity function as one of the most widely used:

$$\frac{\partial Q^{\text{ttl}}(\boldsymbol{\tau}_t, \boldsymbol{a}_t, s_t)}{\partial Q^n\left(\tau_t^n, a_t^n, s_t\right)} \geq 0, \quad \forall n \in \mathcal{N}, t. \tag{3}$$

Then the micro policy network of multiple agents is trained with TD loss:

$$\mathcal{L}_{\text{TD}} = \sum\nolimits_b \left[\left(y_t^{\text{ttl},b} - Q^{\text{ttl}}(\boldsymbol{\tau}_t^b, \boldsymbol{a}_t^b, s_t^b)\right)^2\right], \tag{4}$$

where $b \in \mathcal{B}$ is the batch index and B is the batch size of transitions sampled from a replay buffer. y_t^{ttl} is the target value denoted as $y_t^{\text{ttl}} = r_t + \gamma \max_{\boldsymbol{a}_{t+1}} Q^{\text{ttl}}(\boldsymbol{\tau}_{t+1}, \boldsymbol{a}_{t+1}, s_{t+1}; \theta^-)$, and θ^- is the parameter of the target network.

4 PROPOSED METHOD: HIMACMIC

We propose HiMacMic, consisted of four modules: macro strategy controller (MaSC), micro policy controller (MiPC), high-level mixing network trained with additional heap map loss, and low-level mixing network trained with intrinsic reward, as shown in Figure 2. First, we use MaSC to generate the macro strategy $a_t^{\text{Ma},n}$:

$$a_t^{\text{Ma},n} = (g_t^n, d_t^n) = \text{MaSC}(o_t^n, a_{t-1}^{\text{Ma},n}), \quad \forall t, n, \tag{5}$$

where $a_t^{\text{Ma},n}$ contains spatiotemporal information of a position $g_t^n \in \mathcal{G}$ and deadline $d_t^n \in \mathcal{D}$. d_t^n indicates the number of time steps that $a_t^{\text{Ma},n}$ will last, during which the position will remain the same. MaSC takes local observations o_t^n and $a_{t-1}^{\text{Ma},n}$ as inputs, combined with historical information h_{t-1}^n provided by GRU cell [6] to produce an abstract representation. The latter, as a common head, is then mapped to make macro strategies $a_t^{\text{Ma},n}$ by MLP layers M. The MaSC can be formally expressed as:

$$h_t^n = \text{GRU}(h_{t-1}^n, M(o_t^n, a_{t-1}^{\text{Ma},n})), \quad g_t^n, d_t^n = M(h_t^n). \tag{6}$$

Then macro strategy $a_t^{\text{Ma},n}$ is integrated into the input of MiPC, which decides primitive action $a_t^{\text{Mi},n}$ for each agent to interact with the environment, as:

$$a_t^{\text{Mi},n} = \text{MiPC}(o_t^n, d_t^n, g_t^n, a_{t-1}^{\text{Mi},n}), \quad \forall t, n, \tag{7}$$

where MiPC uses the same structure as in [34]. Next, following [37], we define the joint value function, based on the individual value functions of each agent at the macro strategic level as:

$$\begin{aligned} &\arg\max_{\boldsymbol{a}_t^{\text{Ma}}} Q^{\text{Ma}}(\boldsymbol{\tau}_t, \boldsymbol{a}_t^{\text{Ma}}; \boldsymbol{\Theta}) \\ &= \begin{pmatrix} \arg\max_{a_t^{\text{Ma},1}} Q^1\left(\tau_t^1, a_t^{\text{Ma},1}; \Theta^1\right) \\ \vdots \\ \arg\max_{a_t^{\text{Ma},N}} Q^N\left(\tau_t^N, a_t^{\text{Ma},N}; \Theta^N\right) \end{pmatrix}, \end{aligned} \tag{8}$$

where Θ is the parameters of MaSC. Similarly, the micro policy level task factorization is given by:

$$\begin{aligned} &\arg\max_{\boldsymbol{a}_t^{\text{Mi}}} Q^{\text{Mi}}(\boldsymbol{\tau}_t, \boldsymbol{g}_t, \boldsymbol{d}_t, \boldsymbol{a}_t^{\text{Mi}}; \boldsymbol{\theta}) \\ &= \begin{pmatrix} \arg\max_{a_t^{\text{Mi},1}} Q^1\left(\tau_t^1, g_t^1, d_t^1, a_t^{\text{Mi},1}; \theta_t^1\right) \\ \vdots \\ \arg\max_{a_t^{\text{Mi},N}} Q^N\left(\tau_t^N, g_t^N, d_t^N, a_t^{\text{Mi},N}; \theta_t^N\right) \end{pmatrix}, \end{aligned} \tag{9}$$

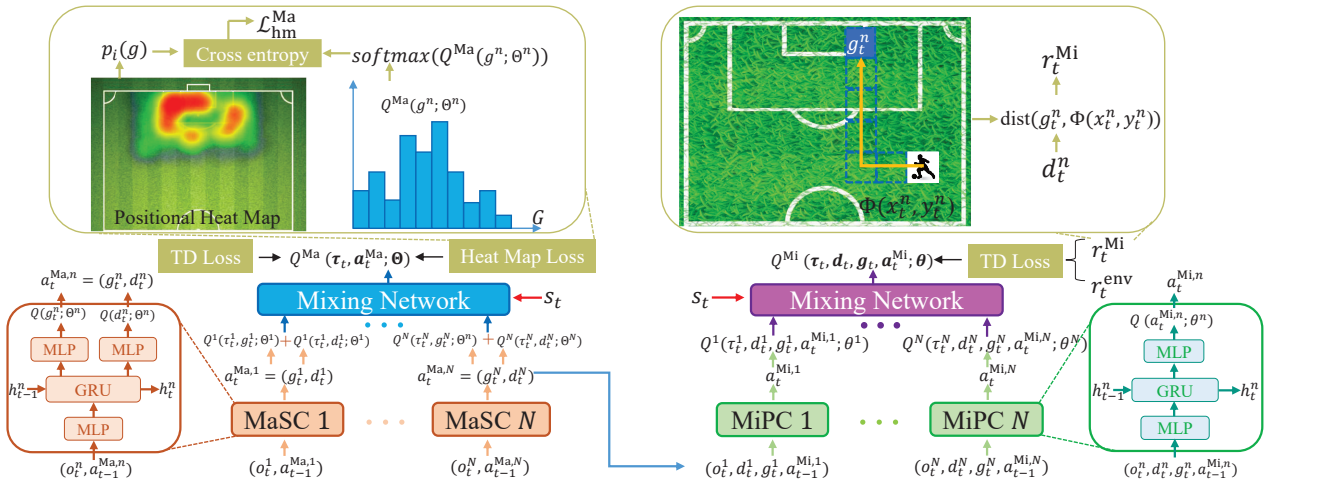


Figure 2: Overall structure of HiMacMic. Bottom is the decision-making process when high-level macro strategy inputs into low-level micro policy. Upper part shows the process of optimizing network parameters in CTDE by using two mixing networks, with the presence of heat map based self imitating learning loss and the intrinsic reward.

where θ is the parameters of MiPC.

Finally, we optimize HiMacMic in CTDE mode by using two mixing networks with the help of a novel heat map based self imitating learning loss $\mathcal{L}_{\text{hm}}^{Ma}$ and an intrinsic reward r_t^{Mi}, under the guidance of the macro strategy. Details will be given in the following sections. HiMacMic has the following advantage, that on the basis of micro policy, it models the agent cooperation by time-varying macro strategies, to aid spatial exploration and temporal decision-making, which might be quite conducive to better understanding and completing tasks in difficult or sparse reward scenarios.

4.1 Asynchronous Deadline-Driven Dynamic Macro Strategy Generation

Existing works HiTS [15] and TempoRL [5] used temporal abstraction from a macro perspective to seek a higher-level strategy. Inspired by these works, we propose an asynchronous deadline (Asy-DDL) driven macro strategy generation process, where durations of macro strategies are not predetermined based on human experiences and will be dynamically changed.

For each agent, when the deadline $d_t^n > 1$, we keep its current position unchanged; otherwise we reselect a macro strategy according to Eqn. (5), as shown in Eqn. (10):

$$a_{t+1}^{\text{Ma},n} = \begin{cases} \left(g_t^n, d_t^n - 1\right), & d_t^n > 1 \\ \left(g_{t+1}^n, d_{t+1}^n\right), & d_t^n = 0. \end{cases} \tag{10}$$

To optimize $a_{t+1}^{\text{Ma},n}$, the reward between the beginning and ending states of a macro strategy within d_t^n time steps is calculated by:

$$r_{t:t+d_t^n-1}^{\text{Ma}}\left(\tau_{t:t+d_t^n-1}, a_{t:t+d_t^n-1}^{\text{Ma}}\right) = \sum_{\Delta t=0}^{d_t^n-1} \gamma^{\Delta t} r_{t+\Delta t}^{\text{env}}, \tag{11}$$

where r_t^{env} is the extrinsic reward from environment. Agent updates its own network parameters according to its own experiences and cumulative rewards by Eqn. (11).

Different from previous studies that artificially defined a fixed number of time steps to skip, the deadline of a macro strategy in HiMacMic is changing dynamically. This not only ensures that the macro strategy has a certain continuity targeting certain spatial position when $d_t^n > 1$, but also expresses the urgency (if d_t^n is relatively small) and degree of task completion (highly likely so if d_t^n is big so that the same macro strategy remain unchanged for a long period). Finally, our method allows different deadline durations of all agents to asynchronously guide the underlying micro policies in a dynamic way.

4.2 Spatial Navigation by Positional Heat Map and Intrinsic Reward

Inspired by recent work in cognitive neuroscience [29] that mice build a spatial map through the neurons in the hippocampal and entorhinal cortex, we propose a two-dimensional heat map $\Phi\left(x_t^n, y_t^n\right) \rightarrow g_t^n \in \mathcal{G}$ to navigate an agent's macro strategic movement. The resolution is a hyperparameter that can be tuned. Specifically, if the agents receive a successful signal at the end of an episode (e.g., receiving battle winning signal in SMAC or maximum reward in one time step in Overcooked and GRF), we calculate the number of visits of all agents to a position g and update the heat map $C_i(g)$ from episode i to $i+1$ as:

$$C_{i+1}(g) = C_i(g) + \sum_n \sum_t F(\Phi(x_t^n, y_t^n), g), \forall g, i, \tag{12}$$

where F is a discriminant function that outputs 1 if and only if the location of agent equal to the position updated in this round. In this way, we obtain a statistical distribution p_i related to successful experiences:

$$p_i(g) = \frac{C_i(g)}{\sum_{g'=0}^{G-1} C_i(g')}, \tag{13}$$

where G is the dimension of position space $\mathcal{G}$. To obtain the self imitation learning loss function based on the heat map, we calculate the cross entropy distance between distribution p and Q as:

$$\mathcal{L}_{\text{hm}}^{\text{Ma}}(\Theta^n) = -\sum_{g=0}^{G-1} p_i(g) \log \frac{\exp(Q(g;\Theta^n))}{\sum_{g'=0}^{G-1} \exp(Q(g';\Theta^n))}, \tag{14}$$

Algorithm 1 HiMacMic

Init: reset environment and initialize $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$.
Output: MaSC and MiPC.

1: **while** episode $i = 1, 2, \ldots$ **do**
2: **for** $t = 1, 2, \ldots, T$ **do**
3: **for** $n = 1, 2, \ldots, N$ **do**
4: Get observation o_t^n from environment;
5: Use Eqn. (5) and (10) to acquire $a_t^{\mathrm{Ma},n}$;
6: Use Eqn. (7) to acquire $a_t^{\mathrm{Mi},n}$;
7: **end for**
8: Take joint primitive action $\boldsymbol{a}_t^{\mathrm{Mi}}$ and acquire reward r_t^{env}, s_t;
9: **end for**
10: **if** update MiPC **then**
11: Sample $(s_t^{1:B}, \boldsymbol{a}_t^{\mathrm{Ma},1:B}, \boldsymbol{a}_t^{\mathrm{Mi},1:B}, r_t^{\mathrm{env},1:B})$ batch data from buffer and calculate r_t^{Mi} by Eqn. (15);
12: Use Eqn.(16) to update MiPC;
13: **end if**
14: **if** update MaSC **then**
15: **for** $n = 1, 2, \ldots, N$ **do**
16: Sample $(s_t, g_t^n, d_t^n, s_{t+d_t^n}^n, r_{t:t+d_t^n}^{\mathrm{env}})$ from buffer ;
17: Calculate $r_{t:t+d_t^n-1}^{\mathrm{Ma}}$ and $\mathcal{L}_{\mathrm{TD}}^{\mathrm{Ma}}$ by Eqn. (11), (17);
18: Calculate heatmap loss $\mathcal{L}_{\mathrm{hm}}^{\mathrm{Ma}}$ by Eqn. (14);
19: Update MaSC by Eqn. (18);
20: **end for**
21: **end if**
22: **end while**

where $Q(g;\Theta^n)$ refers to the macro strategy value of agent n at time step t under a particular position g. The proposed positional heat map has the following advantages:

- It enhances the agent exploration of environment. In Eqn. (13), when no successful experience is collected at the beginning, positional heat map presents a spatially uniform distribution, which is conducive to guide multiple agents to explore task environment comprehensively, and avoid possible local optima.
- It improves sampling efficiency. After successful experiences are gained, certain positions on the heat map become focusing areas as macro strategy, which guides agents to cooperate to complete the task. This will in turn optimize the heat map distributions again, and thus sampling efficiency is improved.

Finally, we use Manhattan distance $\mathrm{dist}(\cdot)$ to measure the sum of absolute difference between an agent's mapped position by macro strategy $\Phi(x_t^n, y_t^n)$ and its predicted position g_t^n from past trajectories. Then, we assign an intrinsic reward r_t^{Mi} from the micro policy to guide the agent to reach the designated position quickly and accurately as:

$$r_t^{\mathrm{Mi}} = \frac{1}{N}\sum\nolimits_n \left(\frac{\sqrt{d_t^n}}{\mathrm{dist}\left(g_t^n, \Phi(x_t^n, y_t^n)\right)} - c\right), \tag{15}$$

where c is a constant penalty.

4.3 Training Process and Complexity Analysis

As shown in Algorithm 1, in each training episode, we first use a macro strategy and micro policy controller to acquire action for each agent (Line 3-7). Next, we use joint primitive action to interact with the environment (Line 8). When an episode is finished, we save useful data (i.e., observation, location of agents, state, reward, game status info from environment and the output of MaSC and MiPC) into a replay buffer (Line 2-9) and reset environment. When updating MiPC, we calculate the intrinsic reward r_t^{Mi} (Line 11) and use batch training to update all agents' MiPC parameter through loss function together:

$$\begin{aligned}\mathcal{L}_{\mathrm{TD}}^{\mathrm{Mi}}(\boldsymbol{\theta}) = \sum\nolimits_b \Bigg[\gamma \max_{\boldsymbol{a}_{t+1}^{\mathrm{Mi},b}} Q^{\mathrm{Mi}}\left(s_{t+1}^b, \boldsymbol{a}_{t+1}^{\mathrm{Ma},b}, \boldsymbol{a}_{t+1}^{\mathrm{Mi},b}; \boldsymbol{\theta}^-\right) \\ - Q^{\mathrm{Mi}}\left(s_t^b, \boldsymbol{a}_t^{\mathrm{Ma},b}, \boldsymbol{a}_t^{\mathrm{Mi},b}; \boldsymbol{\theta}\right) + r_t^{\mathrm{env},b} + \beta r_t^{\mathrm{Mi},b}\Bigg]^2,\end{aligned} \tag{16}$$

where β is the hyperparameter to weight intrinsic reward in micro policy loss function (Line 12). When updating MaSC, we re-organize the macro state sequence between the beginning and ending states of a macro strategy within d_t^n time steps for each agent and calculate TD loss by:

$$\begin{aligned}\mathcal{L}_{\mathrm{TD}}^{\mathrm{Ma}}\left(\Theta^n\right) = \Big[r_t^{\mathrm{Ma},n} - Q^{\mathrm{Ma}}\left(s_t, a_t^{\mathrm{Ma},n}; \Theta^n\right) \\ + \gamma \max\nolimits_{a_{t+d_t^n}^{\mathrm{Ma},n}} Q^{\mathrm{Ma}}\left(s_{t+d_t^n}, a_{t+d_t^n}^{\mathrm{Ma},n}; \Theta^{-,n}\right)\Big]^2,\end{aligned} \tag{17}$$

where $r_t^{\mathrm{Ma},n}$ is multi-step return in Eqn. (11) (Line 16-18). Finally, we use loss (18) to update MaSC (Line 19):

$$\mathcal{L}^{\mathrm{Ma}}(\Theta^n) = \mathcal{L}_{\mathrm{TD}}^{\mathrm{Ma}}(\Theta^n) + \lambda \mathcal{L}_{\mathrm{hm}}^{\mathrm{Ma}}(\Theta^n), \tag{18}$$

where λ is hyperparameter to weight macro loss function.

Correspondingly, the network inference complexity of HiMacMic during training can be expressed as $O\left(\frac{\chi \cdot D^{\mathrm{GRU}} \cdot \sum_\omega D_\omega^{\mathrm{in}} D_\omega^{\mathrm{out}}}{\Lambda}\right)$, where χ, Λ denote the time of environment interaction and parallel runners, respectively, $D^{\mathrm{in}}, D^{\mathrm{out}}$ are the dimensions of input vector and output vector of the ω-th FC layers, respectively, and D^{GRU} is the dimension of GRU-cell in controllers [6].

5 EXPERIMENTAL RESULTS

5.1 Setup

We select four games for performance benchmarking where nine scenarios are chosen to represent either known difficult or sparse reward cases.

- Overcooked [44]: Two chefs (agents) collaborate to accomplish cooking tasks in a gridded kitchen. The task involves subtasks like vegetable selection, cutting, plating, and delivery. Successfully completing the overall task yields an environmental reward of 200, while completing each subtask earns a reward of 10. However, delivering the wrong dish incurs a penalty of -5, and a time step penalty of -0.1 is applied. The final score is used for evaluation purposes.
- GRF [20]: Agents cooperate to score goals. When a goal is scored or the maximum step limit is reached, the environment ends and resets. A reward of 100 is assigned for scoring a goal. We consider two sparse reward scenarios: "keeper"

Table 1: Impact of loss coefficients λ, β.

Overcooked						GRF						SMAC						SMAC-v2					
λ	β					λ	β					λ	β					λ	β				
	0.01	**0.03**	0.05	0.1	0.2		0.01	0.05	**0.1**	0.2	0.3		0.001	0.005	**0.01**	0.015	0.03		0.01	0.015	**0.02**	0.025	0.03
0.01	118.1	118.2	69.47	97.47	83.65	0.01	0.434	0.451	0.454	0.479	0.473	0.01	0.756	0.781	0.839	0.734	0.619	0.1	0.676	0.711	0.685	0.614	0.625
0.03	151.2	210.6	191.3	72.67	37.60	0.1	0.341	0.481	0.606	0.588	0.580	0.05	0.813	0.887	0.903	0.788	0.694	0.15	0.631	0.702	0.723	0.715	0.716
0.05	195.8	**234.6**	201.5	90.41	87.05	**0.3**	0.645	0.722	**0.832**	0.785	0.605	**0.1**	0.845	0.907	**0.935**	0.861	0.745	**0.2**	0.682	0.701	**0.776**	0.712	0.634
0.1	162.7	174.4	186.6	153.5	113.7	0.5	0.705	0.738	0.741	0.783	0.618	0.2	0.726	0.880	0.883	0.847	0.752	0.25	0.681	0.704	0.753	0.721	0.699
0.2	102.6	103.8	87.20	89.31	53.20	0.8	0.402	0.311	0.336	0.276	0.165	0.5	0.707	0.726	0.735	0.781	0.613	0.3	0.608	0.641	0.635	0.635	0.612

Table 2: Impact of heatmap resolution and macro strategy deadline.

GRF						SMAC						SMAC-v2					
resolution	deadline					resolution	deadline					resolution	deadline				
	2	3	**4**	5	6		2	3	**4**	5	6		3	4	**5**	6	7
9	0.646	0.608	0.676	0.597	0.471	25	0.864	0.906	0.863	0.826	0.878	36	0.671	0.689	0.646	0.627	0.631
16	0.627	0.731	0.801	0.725	0.583	36	0.881	0.901	0.894	0.898	0.839	49	0.698	0.710	0.695	0.663	0.652
25	0.734	0.715	**0.832**	0.798	0.617	**49**	0.876	0.897	**0.935**	0.911	0.874	**64**	0.714	0.751	**0.776**	0.705	0.678
36	0.654	0.645	0.735	0.704	0.649	64	0.847	0.866	0.915	0.904	0.881	81	0.674	0.678	0.715	0.697	0.654
49	0.613	0.684	0.603	0.699	0.592	81	0.850	0.843	0.861	0.839	0.817	100	0.688	0.691	0.712	0.681	0.623

involves a confrontation between a few agents in front of the goal, and "hard" represents a more challenging task. The evaluation metric used is the winning rate over 100 rounds.

- SMAC [36]: We choose three cases 6h_vs_8z, 2c_vs_64zg and MMM2 as well-known super hard scenarios [36] that need sufficient exploration and cooperation, where existing method can not solve task well. Battle winning rate over 100 rounds is used for evaluation.
- SMAC-v2 [9]: Known as much more challenging than SMAC, agents on both sides have two types of random initial positions: opposite and surrounded. It adds a type of random units. Taking Zerg as an example, there are 45%, 45% and 10% probabilities to initially generate Zergling, Hydralisk and Baneing, respectively. In the experiment, we selected the maps of Zerg, Protoss and Terran to test. Battle winning rate over 100 rounds is used for evaluation.

In all the experiments, we use Pytorch 1.13.0 to implement HiMacMic, and all the codes are run on a Ubuntu 18.04.4 LTS server with 8 GeForce RTX 3090 graphic cards. By default, we take eight million time steps to train all algorithms. If the scene is difficult and cannot completely converge, we will extend the training time steps to 15 million.

5.2 Hyperparameter Tuning

5.2.1 Impact of macro strategy/micro policy loss function coefficients λ, β: We first show the impact of hyperparameters in the reward and loss function calculation related to training macro strategy/micro policy. We set constant penalty $c = 0.2$, while other parameters follow [33, 34]. In GRF, SMAC and SMAC-v2 environment, we take the average result of all selected scenarios. From Table 1, we observe that appropriate selection of λ, β achieves peak winning rate. This is because the additional guidance from macro strategy and intrinsic reward is insufficient when λ, β is small. On the other hand, large coefficients may bring too much influence on agents to cause training instability, which may result in poor performance.

5.2.2 Impact of Heat map resolution and macro strategy deadline: Next, we show the impact of heat map resolution and macro strategy deadline, which is related to how to analyze and use environment information in both temporal and spatial dimensions. Since Overcooked is a grid game which cannot tune the heat map resolution, we keep it as 25, while changing the resolution in GRF, SMAC and SMAC-v2. We take the average result of all selected scenarios. From Table 2, we observe that resolution in $5 \times 5 = 25$, $7 \times 7 = 49$ and $8 \times 8 = 64$ with 4, 4, 5 maximum macro strategy deadline yields the best winning rate. This is because the extracted spatiotemporal information of macro strategy is insufficient when heat map resolution and maximum macro strategy deadline are small, and too much fine-grained representation of the state and too long time span may bring information redundancy, resulted in poor overall performance.

5.3 Ablation Study

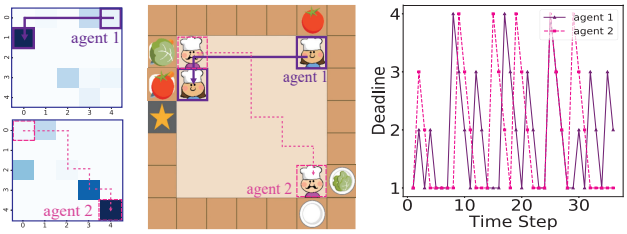
We conduct ablation experiments, by removing key modules proposed in this paper:

- HiMacMic with synchronous deadline (SynDDL): Different from HiMacMic using AsynDDL, we update the macro strategies of all agents synchronously.
- HiMacMic with simple heatmap (w. s-hm): Instead of optimizing macro strategy by the hindsight heatmap according to the success signal from environment, we remove discriminant func F in Eqn. (12) and sample trajectories of all agents uniformly to build a simple heatmap.
- HiMacMic w/o hm: We set $\lambda = 0$ in Eqn. (18), to reflect removing self imitating learning loss function based on heat map.
- HiMacMic w/o r_t^{Mi}: We only use heat map loss without intrinsic reward by setting $\beta = 0$ in Eqn. (16).

From Table 3, we see that HiMacMic with SynDDL obtains lower result, which confirms that synchronous macro strategy is not suitable for all agents. On the contrary, our approach with AsyDDL allows agents to flexibly adjust their macro strategies based on their own observations to achieve better cooperation. When removing self imitating learning loss function based heat map and intrinsic reward, or using simple heatmap without hindsight, the performance drops more than 5%, which confirms their benefits of bringing spatiotemporal information to update macro strategy

Table 3: Ablation Study.

	HiMacMic	w. SynDDL	w. s-hm	w/o hm	w/o r_t^{Mi}
Overcooked	234.6	218.7	219.8	213.5	198.4
GRF: keeper	0.869	0.793	0.702	0.670	0.634
GRF: hard	0.798	0.691	0.477	0.504	0.389
SMAC: 6h_vs_8z	0.851	0.754	0.719	0.679	0.683
SMAC: 2c_vs_64zg	0.969	0.891	0.857	0.861	0.886
SMAC: MMM2	0.985	0.947	0.925	0.901	0.897
SMAC-v2: zerg_5_vs_5	0.793	0.734	0.731	0.702	0.695
SMAC-v2: protoss_5_vs_5	0.763	0.691	0.706	0.688	0.679
SMAC-v2: terran_5_vs_5	0.771	0.722	0.721	0.704	0.718



(a) Heat map (b) Agent movement (c) Asynchronous deadline

Figure 3: Visualization of Overcooked with HiMacMic.

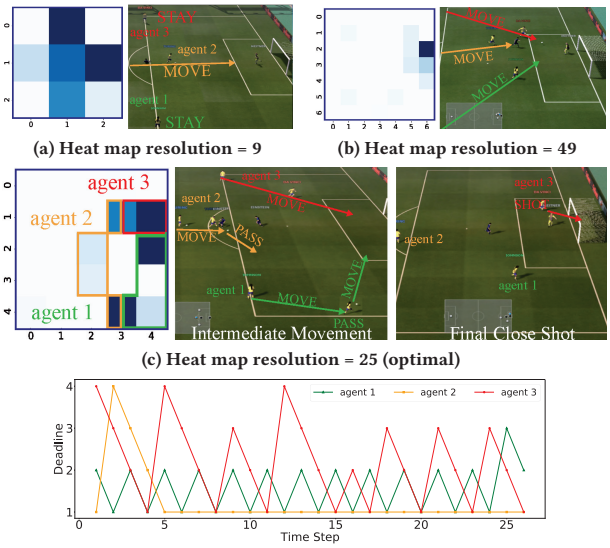
dynamically and using it to guide micro policy. When removing all above modules, the rest becomes vanilla QMIX and results are given in Section 5.5.

5.4 Macro Strategy Visualizations by HiMacMic

We show the macro strategy in four game scenarios. The heatmap shows the spatial distribution of macro strategy position of agents selected in the test phase. The darker the color, the higher the number of choices.

Overcooked: As shown in Figure 3, agent 1 takes a tomato then brings it to the knife for cutting, thus selecting the purple box position as macro strategy. Meanwhile, agent 2 finishes cutting vegetables and then takes the macro strategy advice to load it on a plate. Figure 3(c) further shows the time-varying deadlines of two agents, which reflects when/where to move from the vegetable cutting area to the dish loading area, or from the vegetable picking area to the cutting area, where the macro strategy may last for some time before changing. Furthermore, we see that two agents' macro strategy deadlines may not always coincide, since their tasks are not synchronized as well as their macro strategies.

GRF-keeper: As shown in Figure 4(a), when the heat map resolution is too low, agent 1 and 3 prefer to stay, letting agent 2 to dribble into the penalty area alone. As another extremity, when the heat map resolution is too high, the macro strategies of all agents all point to a small scoring area right in front of the keeper (see Figure 4(b)), so that the only defender can better position himself to stop ball from the final shot. Figure 4(c) shows the optimal heat map resolution 25, where we can clearly see a strategy is generated, that agent 2 moves in front of the penalty area and completes a pass (by selecting the orange box as macro strategy); then agent 1 first moves to the right side to attract the opponent in flank, and passes the ball to the goal and outflank to the keeper (by selecting the green box as macro strategy); finally, agent 3 moves to the goal directly and complete an easy shot (by selecting the red box as macro strategy). This is further confirmed by reading their time-varying

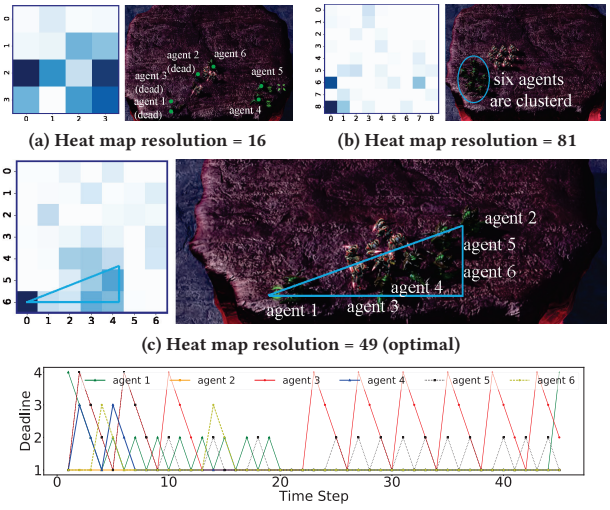


(a) Heat map resolution = 9 (b) Heat map resolution = 49

(c) Heat map resolution = 25 (optimal)

(d) Asynchronous deadline of 3 agents.

Figure 4: Visualization of GRF: keeper with HiMacMic.



(a) Heat map resolution = 16 (b) Heat map resolution = 81

(c) Heat map resolution = 49 (optimal)

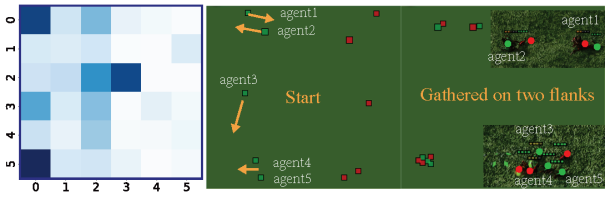
(d) Asynchronous deadline of 6 agents.

Figure 5: Visualization of SMAC: 6h_vs_8z with HiMacMic.

deadlines in Figure 4(c). Agent 2 hosts an increasing deadline duration (referring to ball carrying and passing) at the beginning, and then keeps moving in the front of restricted area (i.e., d_t^n=1). Similar phenomenon is observed for agent 1 and 3, which show the frequent macro strategy update.

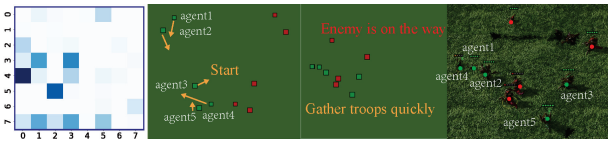
SMAC-6h_vs_8z: As shown in Figure 5(a) and (b), due to inappropriate heat map resolution, agents are too dispersed or clustered, which is difficult to concentrate fire or easily being surrounded. Figure 5(c) shows the optimal resolution 49 where agents form a triangular formation (blue box as macro strategy). Figure 5(d) further shows the time-varying deadlines of six agents at the beginning of 45 time steps, which reflects when/where to move in order to better attack the enemy.

SMAC-v2-zerg_5_vs_5: As shown in Figure 6(a), due to inappropriate heat map resolution, agents are too dispersed where they

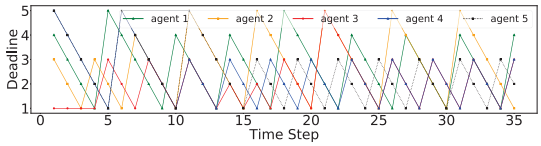


(a) Heat map resolution = 36

(b) Heat map resolution = 100



(c) Heat map resolution = 64 (optimal)



(d) Asynchronous deadline of 5 agents.

Figure 6: Visualization of SMAC-v2: zerg_5_vs_5 with HiMacMic.

fight individually. Agent 1 and 2 fight at the top of the map, while agent 3 moves down to battle with agent 4 and 5. However, a good strategy should build a more powerful group rather than forming multiple battle zones. On the contrary, when the resolution is too high, agents are inclined to form a group right after the game start so that they are surrounded or annihilated by the enemy (e.g., agents 2, 3 and 4 gather in the middle thus quite vulnerable to range damage caused by enemy Baneling units through self-explosion; see Figure 6(b)). Figure 6(c) shows the optimal resolution we found where the agent 1 and 2 move down quickly and gather with others in the middle, to form a local advantage before the enemy's upper right unit arrives. Meanwhile, agents fight against the enemy in the battle zone at the bottom left of the map, but not too close as in Figure 6(b). Instead, they keep relatively scattered in the small battle zone and retain the possibility of attracting the enemy by moving towards the border of the map. Figure 6(d) further shows the time-varying deadlines of five agents at the beginning of 35 time steps, which confirms when/where to move in order to better attack the enemy.

5.5 Comparing with 10 Baselines

We compare HiMacMic with 10 state-of-the-art solutions:

- MADRL for only micro policy, including value-based method VDN [38], QMIX [34], QTRAN [37], OW-QMIX [33], CW-QMIX [33] and QPLEX [43], as well as policy-based method MAPPO [50] and FACMAC [30].

Table 4: Computational complexity of all methods.

Method	Time Cost (ms)	Graphic Card Mem. Usage (GB)
HiMacMic	1.328	1.981
VDN [38]	**1.176**	1.383
QMIX [34]	1.267	1.503
QTRAN [37]	1.437	2.679
OW-QMIX [33]	1.389	2.541
CW-QMIX [33]	1.405	2.673
QPLEX [43]	1.512	1.897
MAPPO [50]	1.226	**1.259**
FACMAC [30]	1.364	1.321
RODE [45]	1.306	1.921
MASER [17]	1.337	1.329

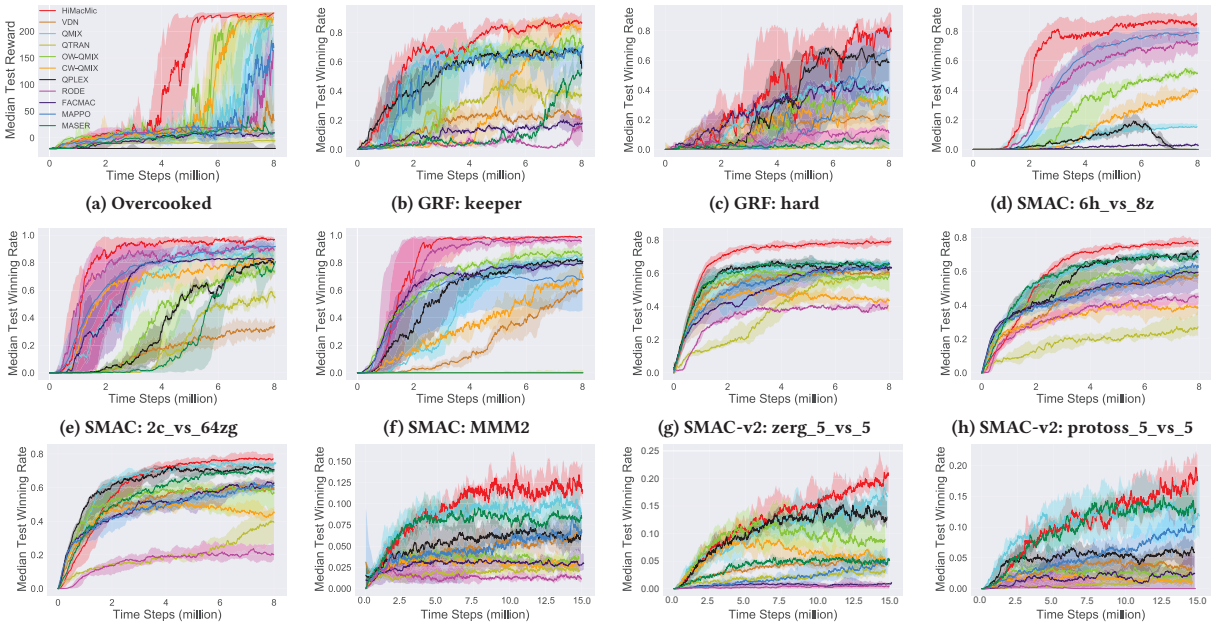
- MADRL for micro policy with hierarchical control, including RODE [45], and MASER [17].

Figure 7 shows the training curves with 8/15 million time steps. We see that HiMacMic converges much faster and gets higher final results. Especially in the most difficult SMAC scenario (see Figure 7 (d)) and SMAC-v2 scenario (see Figure 7 (g)), HiMacMic improves the best performance by around 10%. In GRF scenarios (see Figure 7(b) and (c)), achieves a lot better performance compared with QPLEX. This result reflects the benefits of bringing macro strategy and micro policy together in a hierarchical MADRL framework. Also, by introducing heat map based self imitation learning, the macro strategy updates rapidly and efficiently, to allow agents to fully explore environment with better group cooperations.

HiMacMic also shows a good degree of algorithm stability and adaptability to different environments of the same game. RODE pre-trained different roles to make hierarchical decisions at the beginning and did not change them afterwards, thus performing bad in GRF and SMAC-v2 (see Figure 7(b-c) and Figure 7(g-i)). MASER generated subgoals only from replay buffer, which are insufficient spatiotemporally, thus showing weak performance in Overcookded (see Figure 7(a)) and GRF: hard (see Figure 7(c)) and fail to solve SMAC: 6h_vs_8z (see Figure 7(d)) and 2c_vs_64zg (see Figure 7(e)). This confirms the benefits brought by HiMacMic to make macro strategic decisions dynamically and asynchronously, as well as environment exploration by intrinsic reward.

Figure 7(j)-(l) show the training curves with 15 million time steps in the most challenging SMAC-v2 environment with conical field of view. As said earlier in SMAC-v2, there are three major changes: using random start positions, restricting the agent field of view and shooting range to a cone (where the agent can no longer have a circular view, but needs to select the direction of observation through 12 actions and obtain local observation). It can be seen that our method outperforms all baselines by more than 3%. This confirms the benefits brought by HiMacMic to make macro strategic decisions dynamically and asynchronously, as well as environment exploration by intrinsic reward.

Finally, computational complexity (both time cost and graphic card memory usage) is given in Table 4. We observe that the running time to produce actions in a time step by HiMacMic is similar to that of other baselines, and within the same order of magnitude. The graphic card memory usage of HiMacMic is only slightly higher than others but lower than QTRAN and Weighted QMIX, given the benefits it brings.



(a) Overcooked (b) GRF: keeper (c) GRF: hard (d) SMAC: 6h_vs_8z

(e) SMAC: 2c_vs_64zg (f) SMAC: MMM2 (g) SMAC-v2: zerg_5_vs_5 (h) SMAC-v2: protoss_5_vs_5

(i) SMAC-v2: terran_5_vs_5 (j) SMAC-v2_conic_fov: zerg_5_vs_5 (k) SMAC-v2_conic_fov: protoss_5_vs_5 (l) SMAC-v2_conic_fov: terran_5_vs_5

Figure 7: Experiment results on Overcooked, GRF, SMAC and SMAC-v2.

Table 5: Large-scale and practical experimental results.

Method	Vehicle Dispatch (reward)	SMAC-v2: zerg_20_vs_20
HiMacMic	**109.17**	**0.6931**
VDN [38]	10.54	0.3565
QMIX [34]	101.35	0.6091
QTRAN [37]	-26.13	0.2627
OW-QMIX [33]	102.64	0.5143
CW-QMIX [33]	97.09	0.5625
QPLEX [43]	-14.92	0.3242
MAPPO [50]	103.57	0.5018
FACMAC [30]	-20.14	0.4921
RODE [45]	-29.71	0.1803
MASER [17]	14.19	0.6127

5.6 Large-scale and Practical Experiments

We designed a vehicle dispatching simulator based on real-world ride-hailing datasets [1]. Similar to [40], we rasterized the latitude and longitude range covered the data into a 10*10 square grid. Then, we obtained an initial distribution of order distribution and available taxis based on the average value of the selected data in a time period between 18:00 and 19:00. Based on the initial order distribution and taxi distribution, we initialized 120 orders and 110 taxis in the environment in proportion, where each taxi is an independent agent unit. Our objective is to minimize passenger waiting time. When a passenger and a vehicle occupy the same grid, we consider them to be matched, and the agent completes the task. The episode ends when all agents have completed their matches. Agents have six action options: move (up, down, left, right), stay, and finish. An agent's observation is a local view that includes the number of cars and orders in nearby cells. Each time step, an agent receives a step penalty of -0.01 multiplied by the remaining passenger count n. When an agent successfully completes a subtask (picks up a passenger), it receives a subtask finished reward of 0.2. When all agents complete the matching process, they receive an all finish reward of 100. The multi-agent system shares a global reward, and there is no separate reward function designed for each agent in the environment. The results demonstrate that HiMacMic achieves higher reward values, effectively minimizing the overall waiting time for passengers.

When increasing number of agents in the SMAC-v2-zerg environment, with a 20vs20 matchup, from Table 5, we see that even though the scenario became more complex, HiMacMic still maintained its advantage.

6 CONCLUSION

In this paper, we propose HiMacMic, a hierarchical MADRL framework with both macro strategy and micro policy. Temporally, HiMacMic introduced an asynchronous deadline driven dynamic decision-making process to update macro strategies of all agents flexibly, and spatially, time-varying macro strategies are improved by the positional heat map. Then, low-level micro policies are generated using an intrinsic reward to better complete tasks as a cooperation. Extensive results on Overcooked, GRF, SMAC and SMAC-v2 games show the effectiveness and adaptability of HiMacMic when compared with 10 baselines. We also find the best hyperparameters, and visualize the agent trajectories to better understand the macro strategy and micro policy generated by HiMacMic.

ACKNOWLEDGMENTS

This work has been supported by National Natural Science Foundation of China (No. U21A20519 and 61772072). Corresponding author: Guozheng Li.

REFERENCES

[1] 2015. ECML/PKDD 15: Taxi Trajectory. https://www.kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i/data. Accessed on 6 April 2023.

[2] Sanjeevan Ahilan and Peter Dayan. 2019. Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning. *CoRR* abs/1901.08492 (2019).

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *NeurIPS'17*, Vol. 30. 5048–5058.

[4] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent Complexity via Multi-Agent Competition. In *ICLR'18*.

[5] André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. 2021. TempoRL: Learning When to Act. In *ICML'21*, Vol. 139. 914–924.

[6] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS'14 Deep Learning and Representation Learning Workshop*.

[7] Cédric Colas, Pierre-Yves Oudeyer, Olivier Sigaud, Pierre Fournier, and Mohamed Chetouani. 2019. CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In *ICML'19*, Vol. 97. 1331–1340.

[8] Peter Dayan and Geoffrey E. Hinton. 1992. Feudal Reinforcement Learning. In *NIPS'92*, Vol. 5. 271–278.

[9] Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. 2022. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. *CoRR* abs/2212.07489 (2022).

[10] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *ICLR'19*.

[11] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic Goal Generation for Reinforcement Learning Agents. In *ICML'18*, Vol. 80. 1514–1523.

[12] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *AAAI'18*. 2974–2982.

[13] Yiming Gao, Bei Shi, Xueying Du, Liang Wang, Guangwei Chen, Zhenjie Lian, Fuhao Qiu, GUOAN HAN, Weixuan Wang, Deheng Ye, Qiang Fu, Wei Yang, and Lanxiao Huang. 2021. Learning Diverse Policies in MOBA Games via Macro-Goals. In *NeurIPS'21*, Vol. 34. 16171–16182.

[14] Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artif. Intell. Rev.* 55, 2 (2022), 895–943.

[15] Nico Gürtler, Dieter Büchler, and Georg Martius. 2021. Hierarchical Reinforcement Learning with Timed Subgoals. In *NeurIPS'21*, Vol. 34. 21732–21743.

[16] Xiaotian Hao, Weixun Wang, Hangyu Mao, Yaodong Yang, Dong Li, Yan Zheng, Zhen Wang, and Jianye Hao. 2022. API: Boosting Multi-Agent Reinforcement Learning via Agent-Permutation-Invariant Networks. *CoRR* abs/2203.05285 (2022).

[17] Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. 2022. MASER: Multi-Agent Reinforcement Learning with Subgoals Generated from Experience Replay Buffer. In *ICML'22*, Vol. 162. 10041–10052.

[18] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[19] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *NIPS'16*. 3675–3683.

[20] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2020. Google Research Football: A Novel Reinforcement Learning Environment. In *AAAI'20*. 4501–4510.

[21] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. 2021. Shapley Counterfactual Credits for Multi-Agent Reinforcement Learning. In *KDD'21*. 934–942.

[22] Chi Harold Liu, Zheyu Chen, Jian Tang, Jie Xu, and Chengzhe Piao. 2018. Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach. *IEEE J. Sel. Areas Commun.* 36, 9 (2018), 2059–2070.

[23] Chi Harold Liu, Xiaoxin Ma, Xudong Gao, and Jian Tang. 2020. Distributed Energy-Efficient Multi-UAV Navigation for Long-Term Communication Coverage by Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* 19, 6 (2020), 1274–1285.

[24] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NeurIPS'17*, Vol. 30. 6379–6390.

[25] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NeurIPS'17*, Vol. 30. 6379–6390.

[26] Shuang Luo, Yinchuan Li, Jiahui Li, Kun Kuang, Furui Liu, Yunfeng Shao, and Chao Wu. 2022. S2RL: Do We Really Need to Perceive All States in Deep Multi-Agent Reinforcement Learning?. In *KDD'22*. 1183–1191.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.* 518, 7540 (2015), 529–533.

[28] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

[29] Seongmin A Park, Douglas S Miller, and Erie D Boorman. 2021. Inferences on a multidimensional social hierarchy use a grid-like code. *Nature neuroscience* 24, 9 (2021), 1292–1301.

[30] Bei Peng, Tabish Rashid, Christian Schröder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. In *NeurIPS'21*, Vol. 34. 12208–12221.

[31] Xinghua Qu, Yew Soon Ong, Abhishek Gupta, Pengfei Wei, Zhu Sun, and Zejun Ma. 2022. Importance Prioritized Policy Distillation. In *KDD'22*. 1420–1429.

[32] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. In *ICML'18*, Vol. 80. 4254–4263.

[33] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *NeurIPS'20*, Vol. 33. 10199–10210.

[34] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *ICML'18*, Vol. 80. 4292–4301.

[35] Stuart Russell and Andrew Zimdars. 2003. Q-Decomposition for Reinforcement Learning Agents. In *ICML'03*. 656–663.

[36] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS'19*. 2186–2188.

[37] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *ICML'19*, Vol. 97. 5887–5896.

[38] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *AAMAS'18*. 2085–2087.

[39] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211.

[40] Xiaocheng Tang, Fan Zhang, Zhiwei Qin, Yansheng Wang, Dingyuan Shi, Bingchen Song, Yongxin Tong, Hongtu Zhu, and Jieping Ye. 2021. Value Function is All You Need: A Unified Learning Framework for Ride Hailing Platforms. In *KDD '21*. 3605–3615. https://doi.org/10.1145/3447548.3467096

[41] Runzhe Wan, Xinyu Zhang, and Rui Song. 2021. Multi-Objective Model-Based Reinforcement Learning for Infectious Disease Control. In *KDD'21*. 1634–1644.

[42] Hao Wang, Chi Harold Liu, Zipeng Dai, Jian Tang, and Guoren Wang. 2021. Energy-Efficient 3D Vehicular Crowdsourcing for Disaster Response by Distributed Deep Reinforcement Learning. In *KDD'21*. 3679–3687.

[43] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *ICLR'21*.

[44] Rose E. Wang, Sarah A. Wu, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner. 2020. Too Many Cooks: Coordinating Multi-agent Collaboration Through Inverse Planning. In *AAMAS'20*. 2032–2034.

[45] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2021. RODE: Learning Roles to Decompose Multi-Agent Tasks. In *ICLR'21*.

[46] Yuchen Xiao, Weihao Tan, and Christopher Amato. 2022. Asynchronous Actor-Critic for Multi-Agent Reinforcement Learning. In *NeurIPS'22*.

[47] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. 2020. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *AAMAS'20*. 1566–1574.

[48] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. 2020. CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. In *ICLR'20*.

[49] Rui Yang, Jie Wang, Zijie Geng, Mingxuan Ye, Shuiwang Ji, Bin Li, and Feng Wu. 2022. Learning Task-Relevant Representations for Generalization via Characteristic Functions of Reward Sequence Distributions. In *KDD'22*. 2242–2252.

[50] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2021. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *CoRR* abs/2103.01955 (2021).

[51] Bo Zhang, Chi Harold Liu, Jian Tang, Zhiyuan Xu, Jian Ma, and Wendong Wang. 2018. Learning-Based Energy-Efficient Data Collection by Unmanned Vehicles in Smart Cities. *IEEE Trans. Ind. Informatics* 14, 4 (2018), 1666–1676.