HLSMAC: A New StarCraft Multi-Agent Challenge for High-Level Strategic Decision-Making

Xingxing Hong
Peking University
Beijing, China
hongxingxing@pku.edu.cn

Yungong Wang
University of California, Santa Barbara
Santa Barbara, USA
yungong@ucsb.edu

Dexin Jin
Illinois Institute of Technology
Chicago, USA
jindexin6@gmail.com

Ye Yuan
University of Electronic Science and Technology of China
Shenzhen, China
2020030701007@std.uestc.edu.cn

Ximing Huang
Peking University
Beijing, China
hee@stu.pku.edu.cn

Zijian Wu
Peking University
Beijing, China
gwzj@pku.edu.cn

Yirui Rao
Nanchang University
Nanchang, China
rao-yi-rui@email.ncu.edu.cn

Wenxin Li
Peking University
Beijing, China
lwx@pku.edu.cn

ABSTRACT

Benchmarks are crucial for assessing multi-agent reinforcement learning (MARL) algorithms. While StarCraft II-related environments have driven significant advances in MARL, existing benchmarks like SMAC focus primarily on micromanagement, limiting comprehensive evaluation of high-level strategic intelligence. To address this, we introduce HLSMAC, a new cooperative MARL benchmark with 12 carefully designed StarCraft II scenarios based on classical stratagems from the *Thirty-Six Stratagems*. Each scenario corresponds to a specific stratagem and is designed to challenge agents with diverse strategic elements, including tactical maneuvering, timing coordination, and deception, thereby opening up avenues for evaluating high-level strategic decision-making capabilities. We also propose novel metrics across multiple dimensions beyond conventional win rate, such as ability utilization and advancement efficiency, to assess agents' overall performance within the HLSMAC environment. We conduct a large-scale evaluation of 21 state-of-the-art MARL algorithms and LLM-based agents, with additional multi-seed analysis for relatively better-performing methods. The results demonstrate that HLSMAC serves as a robust testbed for advancing multi-agent strategic decision-making.

KEYWORDS

StarCraft II; Multi-Agent Reinforcement Learning; Benchmark

ACM Reference Format:
Xingxing Hong, Yungong Wang, Dexin Jin, Ye Yuan, Ximing Huang, Zijian Wu, Yirui Rao, and Wenxin Li. 2026. HLSMAC: A New StarCraft Multi-Agent Challenge for High-Level Strategic Decision-Making. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages. https://doi.org/10.65109/ELYZ1330

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/ELYZ1330

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has achieved significant advancements, driven by diverse benchmarks across cooperative, competitive, and mixed settings. Environments like the StarCraft II Learning Environment (SC2LE [41]) and its popular derivative, the StarCraft Multi-Agent Challenge (SMAC [29]) series, alongside other prominent testbeds, have driven sophisticated algorithmic development.

However, current MARL benchmarks present several critical limitations that hinder progress in the field. Most existing benchmarks, including SMAC, emphasize micromanagement over strategic decision-making. While full-game environments like AlphaStar [39] capture strategic complexity, their massive computational demands make them infeasible for broad evaluation. Additionally, current benchmarks inadequately leverage established human strategic wisdom, relying primarily on emergent learning from environmental interactions rather than assessing agents' capacity to integrate human knowledge. This scarcity is further exacerbated by the time-consuming nature of developing new robust benchmarks.

Beyond traditional MARL approaches, the emergence of large language models (LLMs) provides a promising alternative paradigm for multi-agent decision-making. LLMs offer unique advantages through their advanced reasoning capabilities, interpretability, and inherent knowledge of human strategic principles. Recent research demonstrates their potential across diverse strategic environments, from simulating complex social interactions in Werewolf [10] to strategic planning in games such as Diplomacy [4] and Chess [5]. However, LLM-based approaches face significant challenges: inherent issues such as hallucination and coordination complexities in multi-agent settings pose obstacles to developing robust strategic intelligence.

To address these limitations, we introduce **HLSMAC** (**S**tarCraft **M**ulti-**A**gent **C**hallenge for **H**igh-**L**evel Strategic Decision-Making), a novel benchmark that translates classical Chinese Thirty-Six Stratagems into AI evaluation scenarios. We select 12 representative stratagems, each chosen for its clear principles and practical feasibility within game environments. Each stratagem is embodied in a dedicated StarCraft II map that is named after it. These maps are designed to challenge agents with diverse strategic elements, such as tactical maneuvering, timing coordination, and deception. The key innovations of HLSMAC include: (1) the systematic integration of established human strategic wisdom; (2) its emphasis on high-level strategic decision-making over micromanagement; and (3) compatibility with popular frameworks like PyMARL [29] and LLM-PySC2 [15].

Our contributions are as follows. We introduce HLSMAC, which, to our knowledge, is the first benchmark to systematically integrate the Thirty-Six Stratagems into multi-agent AI evaluation, shifting focus from micromanagement to high-level strategic decision-making. Second, we provide an extensive evaluation of 21 state-of-the-art MARL algorithms and the LLM-PySC2 framework, revealing current limitations and establishing a robust testbed. Third, we propose novel evaluation metrics beyond win rate to comprehensively assess strategic intelligence.

To ensure terminological clarity, we distinguish between three key concepts. According to [29], *macromanagement* refers to high-level strategic considerations such as economy and resource management, as distinguished from fine-grained unit control or micromanagement. In our work, a *stratagem* is one of the classical Chinese Thirty-Six Stratagems, each representing a carefully planned way of achieving a goal, often involving clever tactics or deception. Building on this, we define *high-level strategic decision-making* as the complex process of formulating and executing such deliberate plans. Our benchmark HLSMAC thus emphasizes both the high-level cognitive demands required for the tasks and the human-like strategic wisdom embodied in the Thirty-Six Stratagems.

2 RELATED WORK

2.1 Multi-Agent Reinforcement Learning Benchmarks

2.1.1 StarCraft II-based Environments. StarCraft II, as a real-time strategy game, provides an ideal testbed for multi-agent reinforcement learning with unique challenges including imperfect information, vast combinatorial action spaces, and long-term temporal credit assignment. Early works like TorchCraft [35] and SC2LE [41] enable training on StarCraft games. Subsequently, SMAC [29] has become one of the most popular benchmarks for cooperative MARL, focusing on decentralized micromanagement challenges. SMACv2 [3] introduces procedurally generated scenarios and extended partial observability challenges to ensure agents must learn genuine closed-loop policies that condition on observations. SMAC-Hard [2] addresses the critical limitation of existing benchmarks where algorithms exploit specific weaknesses in static opponents rather than learning robust strategies, introducing opponent strategy editing, randomized opponent selection, and black-box testing frameworks. AlphaStar [40] and TStarBot-X [7] both achieve competitive performance in StarCraft II.

2.1.2 Other Prominent Testbeds. Beyond early grid-world environments [14, 19], MARL research has increasingly adopted more comprehensive platforms. OpenSpiel [13] offers a collection of environments and algorithms for MARL, as well as search and planning in games. For human-AI coordination, Overcooked-AI [1] provides a dedicated collaborative cooking environment. FACMAC [23] adapts continuous control environments like MuJoCo for multi-agent settings, while PettingZoo [37] offers a broad library covering competitive, cooperative, and mixed-sum games, including classic Atari and board games. The Melting Pot Contest [38] provides a testbed for measuring cooperative intelligence. For physics-based simulations, Google Research Football [12] provides a 3D soccer simulator with various benchmark tasks. More recently, Honor of Kings Arena [50] presents a competitive RL environment based on Honor of Kings, introducing new generalization challenges. Collectively, these platforms underscore the field's rapid progress.

2.2 Methods for Solving MARL Benchmarks

2.2.1 MARL Algorithms. Multi-agent reinforcement learning algorithms can be broadly categorized into two main approaches: value-based and policy-based. For value-based methods, IQL [36] treats other agents as part of the environment. To address the non-stationarity problem, VDN [33], QMIX [28], QTRAN [32], QPLEX [43], and others propose various methods to decompose the global Q-function into individual agent Q-functions. Qatten [51] employs multi-head attention structures for more accurate value decomposition. WQMIX [27] introduces weighting mechanisms to overcome the monotonicity constraints of standard mixing methods. Additional methods include ROMA [44] and RESQ [25], which further enhance coordination capabilities through various approaches, such as role-based decomposition mechanisms and improved network architectures. For policy-based methods, MADDPG [20] extends DDPG to multi-agent settings, BicNet [24] introduces bidirectional communication, and COMA [6] employs counterfactual reasoning. LICA [54] presents a multi-agent actor-critic method that formulates the centralized critic as a hypernetwork and employs adaptive entropy regularization. Additionally, trust region methods such as HATRPO and HAPPO [11] provide theoretical guarantees for cooperative settings.

2.2.2 LLM-based Approaches. Large language models have shown potential in solving MARL problems. LLM-PySC2 [16] introduces a novel environment that enables LLMs to interact with StarCraft II. ChessGPT [5] demonstrates the LLM's potential to integrate strategic learning with language understanding in complex decision-making tasks. ONUW [10] emphasizes the importance of strategic discussion tactics in shaping player beliefs and game outcomes in Werewolf. GITM [55] and Voyager [42] leverage the MineDojo environment to demonstrate the capabilities of LLM agents in navigation and task execution within Minecraft.

3 HLSMAC

In this section, we first examine the key design features considered in constructing HLSMAC scenarios to support high-level strategic decision-making, and then illustrate how stratagems are incorporated into the scenarios via two examples.

3.1 Design Features of Scenarios

HLSMAC fully leverages the inherent characteristics of StarCraft II to enable the evaluation of high-level strategic decision-making over micromanagement. Specifically, we utilize the game's rich strategic complexity, flexible map editor, and diverse official map pool to create tailored HLSMAC scenarios. The following details the core design features of the scenarios.

3.1.1 Larger Map Sizes and Richer Terrain Elements. To enhance strategic complexity, each map features almost 80×80 grids, compared to SMAC's standard 32×32 layouts, providing longer movement distances, more route options, and additional combat zones. These expanded dimensions also make it feasible to place more units and structures, as well as design more diverse terrain, including high grounds, choke points, and open fields. Although creating such terrains is typically challenging, we streamline the process by cropping from official StarCraft II ladder maps, reproducing authentic battlefield dynamics. We expect agents to exploit spatial and terrain features for strategic decisions, rather than focusing solely on micromanagement.

3.1.2 Expanded Unit and Structure Abilities. We introduce more game-inherent abilities for units and structures to expand the action space. In contrast with SMAC, where unit abilities are restricted to basic functions like `move`, `stop`, and `attack` that primarily support micromanagement, HLSMAC's diverse abilities are highly relevant to the specific scenario objectives and can enhance task performance when utilized properly. For units, Zerglings can use `Burrow` to enable ambushes or retreats, while Sentries may cast `Hallucination` to create fake allies, deceiving opponents about the true force composition. For structures, operational abilities are selectively extended, such as the `Load` and `Unload` abilities for Nydus Worms, which facilitate rapid unit transportation, and the `WarpIn` ability for Warp Gates, enabling instant unit deployment. Considering the PySC2 library's compatibility for action space extension, we use built-in StarCraft II units (and structures) instead of SMAC's RL units. The proper usage of such abilities indicates strategic thinking that involves deception, misdirection, and long-term planning, thereby shifting the action space from reactive to strategic.

3.1.3 Diverse Opponent Policies. We define diverse opponent policies using the StarCraft II Editor's trigger system, which generates Galaxy scripts for dynamic behavior control. These policies exhibit various patterns, including sustained defensive or offensive postures and dynamic transitions between these stances. Transitions are implemented through coordinated changes in ability usage, combat modes, and movement paths, creating adaptive opponent behaviors. To illustrate, the *dhls* ("Lure the Tiger Down the Mountain") scenario employs multiple trigger zones to dynamically reposition enemy forces in response to allied actions. The implementation leverages the built-in library functions(e.g., *UnitGroupIssueOrder*), numerical computations (e.g., *SetVariable*), and conditional logic (e.g., *Comparison*) to create realistic opponent responses.

3.1.4 Redefined Game Termination Conditions. We redefine game termination conditions for HLSMAC scenarios. Rather than using the elimination of all enemy units as the sole victory condition, we implement more diverse success criteria, such as destroying critical structures or maintaining unit survival for specified periods. For instance, as described in the Thirty-Six Stratagems, targeting key assets is often more effective than engaging in exhaustive confrontation. This shift from elimination-based victory conditions to strategic objectives encourages higher-level strategic planning rather than micro-level optimization.

We comprehensively integrate the above design features to carefully craft scenarios that require high-level strategic reasoning to succeed. We expect that most existing methods will struggle to succeed unless they follow the stratagem.

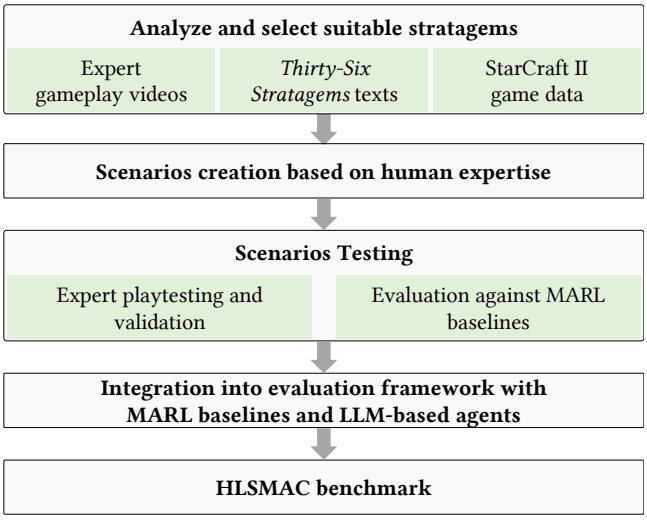


Figure 1: Benchmark Construction Pipeline

3.2 Scenarios

The Thirty-Six Stratagems is a unique and well-known collection of ancient Chinese proverbs that describe some of the most cunning and subtle tactics ever devised. This classical work represents a sophisticated codification of human strategic thinking. To our knowledge, HLSMAC is the first benchmark to systematically integrate the stratagems into the research on multi-agent intelligence.

We adopt the benchmark construction pipeline as shown in Figure 1. First, we comprehensively gather and study the Thirty-Six Stratagems texts, gameplay videos of human players applying these stratagems, and relevant StarCraft II game data. Following this, we analyze the core strategic concepts of these stratagems and select those most suitable for the StarCraft II environment. Next, we leverage human expertise to design scenario storylines, then implement them by configuring terrain, game mechanics, and unit abilities through the StarCraft II Editor. Each scenario is iteratively refined through testing to ensure that it accurately captures the essence of the stratagem.

We eventually select 12 representative stratagems, each chosen for its clear strategic principles and practical feasibility within game environments. The complete list of 12 HLSMAC scenarios is presented in Table 1, which provides a brief overview of the unit compositions for each scenario. Maps are named using the initials of the stratagems' Chinese names. The stratagems and scenarios

are detailed in the appendix of the full paper (arXiv: https://arxiv.org/abs/2509.12927).

We now present two representative examples to illustrate the integration of human strategic wisdom into HLSMAC scenarios. For each example, we examine the original stratagem, describe the scenario mechanics, and propose the expected solution.

3.2.1 Example 1: Besiege Wei to Rescue Zhao (wwjz). This stratagem derives its name from a famous ancient incident that occurred in 354 B.C. When Wei forces besieged Zhao's capital, to rescue Zhao from its predicament, the strategists chose to attack Wei's capital instead of the besieging army, forcing Wei to abandon the siege and rush back to defend its homeland. The exhausted Wei troops were then ambushed and defeated during their retreat. Its strategic logic is articulated as follows: When the enemy is too strong to attack directly, strike at something he holds dear.

Based on this stratagem, we design the following scenario (see Figure 2(a)). When the game starts, our Nexus (upper-left) is under frantic assault by 8 Hellions and 6 Marines, with no local defenses. Meanwhile, our seven Zealots are positioned in the upper-right region of the map. Our objective is clear: either destroy the enemy Command Center or eliminate all enemy units, all while ensuring our Nexus survives. To simulate the core stratagem, a crucial trigger is in place: the moment our Zealots approach the enemy Command Center, all enemy forces will immediately disengage from our Nexus and retreat to defend their own base, mirroring the dynamics of *Besiege Wei to Rescue Zhao*. Through extensive testing, we have calibrated the scenario to ensure that our Zealots cannot reach the Nexus in time for a direct defense. Even if the Zealots retreat and engage the enemy directly, we are likely to lose.

Correspondingly, the expected solution is as follows. Rather than engaging the superior enemy force in direct combat, the Zealots should encircle the enemy Command Center, embodying the *Besiege Wei* principle. This forces the Hellions and Marines to abandon their assault on the Nexus and retreat to defend their base, thereby achieving the *Rescue Zhao* effect. Furthermore, due to the speed difference between Hellions and Marines, the enemy forces return in a scattered formation, allowing the Zealots to defeat them in successive waves as they arrive piecemeal at their base.

3.2.2 Example 2: Lure Your Enemy onto the Roof, Then Take Away the Ladder (swct). This stratagem derives its core principle from Sun Tzu's brilliant deception at the Battle of Maling in 341 B.C. When facing a powerful enemy's pursuit, the stratagem avoids direct engagement, but instead designs an elaborate trap to lure the enemies deeper, subsequently blocking their line of advance and destroying the entire force in a narrow valley.

Inspired by this stratagem, we design the following scenario (see Figure 2(b)). When the game starts, our 1 Warp Prism (with `Load` and `Unload` abilities) and 4 Sentries (capable of casting `Force Field`) face a complex challenge. The enemy Hatchery is located on nearby high ground, accessible only through a narrow choke, and along the path to this choke, 10 enemy Zerglings are stationed as defenders. Our goal is either to destroy the enemy Hatchery or eliminate all the Zerglings. The trigger is configured as follows: once our forces approach the Hatchery, all enemy Zerglings will immediately retreat to defend their own base. It must be emphasized that due to the disparity in unit numbers, our forces cannot achieve victory through direct engagement with the Zerglings.

Similarly, the expected solution requires human-like strategic thinking. The Warp Prism should bypass the enemy Zerglings' frontline and deploy Sentries onto the high ground. Once the enemy Hatchery is under a surprise attack, the Zerglings will retreat, allowing the Sentries to maintain Force Field barriers at the choke point. This either delays reinforcements to destroy the Hatchery or isolates and eliminates returning Zerglings, thus mirroring and executing this classic stratagem.

4 EVALUATION FRAMEWORKS

From the outset, the HLSMAC benchmark aims to interface with both Multi-Agent Reinforcement Learning (MARL) and Large Language Model (LLM) frameworks, providing a unified testbed for exploration of how different AI paradigms tackle high-level strategic decision-making tasks. To achieve this, our implementation specifically integrates PyMARL for MARL approaches and LLM-PySC2 for large language model applications.

Both frameworks share a common foundation that enables this integrative capability: the StarCraft II interface ecosystem, developed by Blizzard, DeepMind, and the research community, comprising StarCraft II binaries (for Windows and Linux), the StarCraft II API, and PySC2. Taking PySC2 as a key example, this open-source Python library provides general capabilities that extend beyond its initial RL optimization, enabling diverse agents to extract observations, execute actions, and access game state information for strategic intelligence development.

Despite sharing a common foundation, the two frameworks serve distinct purposes. PyMARL focuses on training reinforcement learning models and supports multiple environments, whereas LLM-PySC2 specializes in large language model inference for StarCraft II scenarios. Below, we present the integration details for both frameworks.

4.1 PyMARL Framework

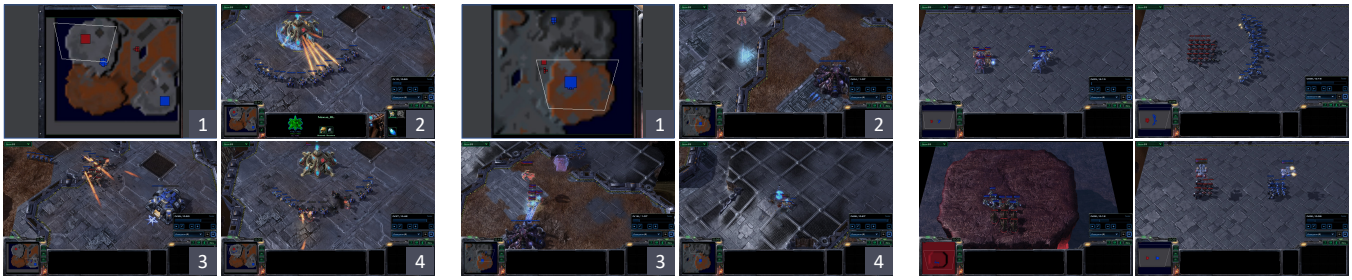
PyMARL[29], initially built for developing MARL algorithms for SMAC, has become a widely adopted, open-source framework across diverse multi-agent environments. Its modular architecture facilitates the integration of state-of-the-art algorithms. Numerous fork implementations with various baselines have emerged, providing an ideal evaluation platform for HLSMAC.

It is worth noting that PyMARL serves solely as a training framework, decoupled from environment components. To integrate HLSMAC with PyMARL's baseline algorithms, our core effort focuses on developing a dedicated environment codebase for its game maps.

4.1.1 HLSMAC Environment Implementation. The HLSMAC environment uses the factory pattern for modular design. A shared BaseEnv class handles common logic, while each of the 12 scenarios inherits from it to implement specific action spaces, unit state updates, and game termination conditions. This design achieves logical separation of scenarios, promotes code reusability, and facilitates extensibility for new scenarios. Additionally, by mirroring SMAC's environment wrapper design, HLSMAC enables backward compatibility with existing SMAC maps.

Table 1: HLSMAC Scenarios. Name: Chinese stratagem initials. Z: Zerg, T: Terran, P: Protoss.

Name	Ally Units and Structures	Enemy Units and Structures	Race Matchup
adcc	16 Zerglings, 1 Hatchery	4 Hellbats, 1 Command Center	ZvT
dhls	9 Zerglings, 4 Roaches, 1 Hatchery, 1 Nydus Network, 1 Nydus Worm	8 Marines, 2 Siege Tanks, 1 Command Center	ZvT
fkwz	2 Warp Gates, 2 Pylons, 1 Warp Prism	1 Stalker, 1 Gateway, 2 Pylons, 3 Photon Cannons	PvP
gmzz	5 Marines, 3 Supply Depots	8 Zerglings, 3 Spine Crawlers	TvZ
jctq	4 Roaches	2 Sentries, 6 Stalkers, 1 Observer	ZvP
jdsr	4 Roaches, 1 Infestor	3 Stalkers, 1 Colossus	ZvP
sdjx	14 Marines, 4 Medivacs	5 Zealots, 4 Stalkers, 3 Colossi, 2 Nexuses, 2 Assimilators, 1 Pylon	TvP
swct	4 Sentries, 1 Warp Prism	10 Zerglings, 1 Hatchery	PvZ
tlhz	1 Drone, 3 Larvas	1 Photon Cannon, 1 Pylon	ZvP
wwjz	7 Zealots, 1 Nexus	8 Hellions, 6 Marines, 1 Command Center	PvT
wzsy	2 Stalkers, 4 Sentries	3 Immortals, 5 Zealots, 1 Nexus, 1 Pylon	PvP
yqgz	24 Zerglings	6 Marines, 2 Siege Tanks	ZvT



(a) HLSMAC scenario wwjz (b) HLSMAC scenario swct (c) Four SMAC scenarios

Figure 2: In subfigures (a) and (b), scene 1 shows the initial battlefield (red: our forces, blue: enemy). Scenes 2-4 depict the early stage, the victory stage (with stratagem), and the defeat stage (without stratagem), corresponding to each scenario. Subfigure (c) presents four classical SMAC scenarios featuring direct combat at spawn. HLSMAC's battlefields are more complex.

Considering that certain abilities in HLSMAC scenarios, such as `Hallucination` and `WarpIn` create new units mid-game, our environment implements support for dynamic unit spawning, a feature unavailable in SMAC. To handle this dynamism, the environment pre-allocates unit slots at the beginning of each episode, initially set to None for potential spawning. These empty slots are padded with zero-filled observation and state vectors, along with `no-op` actions. When units spawn, the corresponding None slots are replaced with actual unit data, enabling flexible scalability for complex multi-agent scenarios.

HLSMAC scenarios feature diverse victory conditions, such as destroying critical structures or maintaining unit survival for specified periods. Given this diversity, HLSMAC implements tailored termination logic for each scenario. Additionally, the environment enhances the reward-tracking mechanism by independently logging reward components, including delta_ally, delta_enemy, and delta_death, to help analyze agent training processes.

HLSMAC adopts the same data structures for state and observation as SMAC, primarily for maintaining compatibility with existing baselines.

4.1.2 Integration with PyMARL. Integrating HLSMAC with PyMARL frameworks requires minimal modification, typically just registering the new environment and updating the environment selector logic by adding a few lines of code. This simplicity stems from HLSMAC's close alignment with the SMAC environment's underlying implementation, enabling straightforward algorithm integration and evaluation within PyMARL's framework.

4.2 LLM-PySC2 Framework

4.2.1 Extend LLM Agents Configuration. The LLM-PySC2 framework employs a hierarchical agent system comprising a MainAgent and SubAgents. MainAgent coordinates the SubAgents, while each SubAgent interfaces with the LLM to handle specific tasks like building construction and unit production.

We extend the framework from Protoss-only to all three races by configuring additional agents for HLSMAC scenarios. The extension process follows two core principles. The first is designing agents based on specific scenario requirements. For example, in the *gmzz* (Shut the Door to Catch the Thief) scenario, we develop a dedicated Supply Depot agent to handle the complexity of depot management. The second approach is to group the agents by unit attributes or functional similarity. In the *sdjx* ("Clamour in the East, Attack in the West") scenario, Medivacs join the Air Force group while Marines join the Ground Force group. This grouping mirrors human gameplay logic while maximizing framework modularity.

4.2.2 Interface with LLM Agents. We develop an interface supporting 3 levels of HLSMAC task descriptions: level 0 primarily includes stratagem names and explanations; level 1 additionally provides game mechanics; and level 2 further incorporates expected solutions. For example, a level 0 prompt might be "Follow the [stratagem name] to defeat the enemy," which is then combined with real-time game information to form the complete prompt.

5 EVALUATION METRICS AND RESULTS

StarCraft II provides performance indicators such as average unspent resources, time supply capped, workers created, and APM (Actions Per Minute). These metrics target comprehensive, match-long performance based on the full map and are not suitable for HLSMAC scenarios. Therefore, we carefully examine the existing framework-related metrics and introduce HLSMAC's scenario-specific metrics.

5.1 Existing Framework-Related Metrics

MARL algorithms typically offer two types of evaluation metrics. Training performance metrics track learning dynamics and convergence, encompassing optimization indicators such as loss, td_error_abs, and target_mean, reward statistics like reward_mean and reward_std, and combat effectiveness measures like battle_won _mean, dead_allies_mean, and dead_enemies_mean. On the other hand, test performance metrics utilize corresponding test versions of training metrics, prefixed with "test_". The test_battle_won_mean serves as the primary win-rate indicator. This dual-metric framework enables a comprehensive assessment of both algorithmic learning capabilities and practical combat effectiveness.

Similar to MARL methods, LLM-PySC2 evaluates the capabilities of large models based on winning_rates, kill_rates, and death_rates.

5.2 HLSMAC Scenario-Specific Metrics

We propose a set of metrics broadly applicable to diverse baselines in HLSMAC scenarios. These metrics assess performance from several additional dimensions, including critical target advancement, ability utilization frequency, target damage, and unit survival rate. To calculate these metrics, we extract data from the game replay files generated by the evaluated algorithms.

Each metric is defined as follows, where N is the number of game episodes for each scenario (map).

`Critical_Target_Advancement` measures how effectively allied units move toward enemy critical targets. These targets typically refer to main structures such as Command Center and Hatchery, strategic coordinates, and key enemy units, depending on scenario. This metric can be computed using two methods:

- Target Proximity Frequency (TPF) reflects the average frequency of allied units entering within a specified distance range around enemy critical targets per game episode.

$$\text{TPF} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{1}(d_{ij} \leq L) \tag{1}$$

  Where M is the number of allied combat units in the scenario, d_{ij} is the distance between the j-th unit and the enemy critical targets at any game loop in episode i, L is a threshold defining the distance range around the target (e.g., $L = 6$), and $\mathbf{1}(\cdot)$ is the indicator function.

- Target Directional Alignment (TDA) computes the projection ratio of actual displacement vectors onto shortcut vectors.

$$\text{TDA} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{\vec{v}_{ij} \cdot \vec{t}_{ij}}{||\vec{t}_{ij}||^2} \tag{2}$$

  Where $\vec{v}_{ij}$ is the actual displacement vector of the j-th unit in episode i (final position - initial position), $\vec{t}_{ij}$ is the shortcut vector from the initial position to the enemy critical targets.

`Ability_Utilization_Frequency` measures how frequently allied units cast special abilities during gameplay. Such abilities include `Burrow`/`Unburrow`, `Load`/`Unload`, and other actions beyond basic movement and attack.

$$\text{AUF} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} A_{ik} \tag{3}$$

Where K is the number of distinct ability types tracked, and A_{ik} is the count of type k ability cast in episode i.

`Critical_Target_Damage` measures the total damage dealt by allied units to enemy critical targets during gameplay.

$$\text{CTD} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{J} \frac{D_{ij}}{D_{max,j}} \tag{4}$$

Where J is the number of enemy critical target types, D_{ij} is the total damage dealt to type j enemy critical targets in episode i, and $D_{max,j}$ is the total health points of type j enemy critical targets.

`Unit_Survival_Rate` is the average ratio of surviving units to initial unit count. It measures the AI's capability to preserve units during gameplay.

$$\text{USR} = \frac{1}{N} \sum_{i=1}^{N} \frac{U_{remaining,i}}{U_{initial,i}} \tag{5}$$

Where $U_{remaining,i}$ is the number of allied units surviving at the end of episode i, and $U_{initial,i}$ is the initial number of allied units at the beginning of episode i.

5.3 Results

We conduct a broad evaluation of 21 MARL algorithms and various LLM-based agents, then perform multi-seed evaluation on the top-performing MARL methods. Key findings include:

First, HLSMAC poses significant challenges for both MARL and LLM-based methods. As shown in Table 2, nearly 80% of MARL algorithm-scenario combinations achieved zero win rates. For LLM-PyMARL, we conducted 432 trials (4 LLMs × 12 maps × 3 levels × 3 trials). Only 3 trials succeeded: DeepSeek on *wwjz* (level 1) and *sdjx* (level 2), and Qwen on *gmzz* (level 1).

Second, win-rate metrics alone inadequately capture agents' human-like strategic decision-making capabilities in HLSMAC. For example, RIIT achieves a 93% win rate in *adcc*, but replay analysis reveals that it does not actually follow the intended stratagem. Similarly, DOP achieves only a 19% win rate in *swct*, yet exhibits repeated loading and unloading of Sentries with Warp Prism, a behavior rarely seen in normal human play.

Third, our new metrics, together with win rate, offer richer dimensions for evaluating the performance of various methods. Through R^2 analysis of applicable metric-scenario combinations,

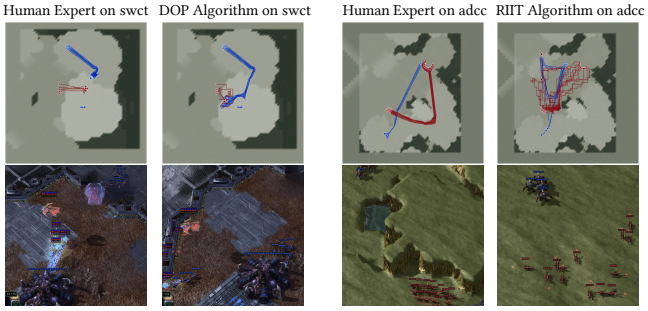


Figure 3: Top row: Trajectory comparison between human expert and MARL algorithms. The red and blue traces represent allied and enemy movements in one episode, respectively. Bottom row: Corresponding screenshots of the replays.

Figure 4: Heatmap of R^2 values showing correlations between win rates and evaluation metrics across scenarios.

we find that these metrics demonstrate strong explanatory power ($R^2 \geq 0.6$) in 40% of the combinations (Figure 4). Metrics such as TPF, TDA, and CTD serve as effective indicators of high win-rate performance in the *wzsy*, *wwjz*, and *sdjx* scenarios. This reflects that success requires approaching and neutralizing critical targets, as exemplified by the *wwjz* and *sdjx* scenarios. Furthermore, although AUF cannot directly reflect the appropriateness of ability usage, it can distinguish between human players and current methods, as humans demonstrate purposeful utilization of critical abilities, whereas current methods do not. For instance, in the *gmzz* scenario, the human expert consistently maintains AUF of 2, while MARL methods range from 0 to 600 and LLM methods range from 18 to 78, showing substantial deviation from human performance. Moreover, complementing these quantitative metrics derived from offline replays, visualization of replay trajectories (Figure 3) also facilitates intuitive observation of execution processes.

5.4 Analysis

Experimental results reveal a fundamental understanding-execution gap in both MARL and LLM methods. For MARL methods, success stems from maximizing cumulative reward rather than understanding stratagem principles. For example, in *jdsr* ("Kill with a Borrowed Sword") scenario, high win rates do not necessarily indicate understanding of the "borrowed sword" concept behind `Neural Parasite`.

Conversely, for LLM methods, articulating stratagems does not ensure reliable execution. In *sdjx*, Gemini 2.5 Pro exhibited concurrent rather than sequential execution of the "clamour in the East" and "attack in the West" sub-tasks. In *gmzz*, GPT-5 frequently raised the supply depot immediately after lowering it, failing to trap Zerglings on the high ground. Three factors might contribute to the understanding-execution gap: (1) HLSMAC complexity: multi-step execution with implicit sub-tasks challenges coordination; (2) LLM-PySC2 constraints: the multi-turn pipeline introduces cumulative information loss across observation-to-text-to-action transformations; (3) LLMs' inherent limitations: hallucination and textual ambiguity lead to failures in translating stratagem intent into precise actions. Differentiating whether failures stem from stratagem intent or execution breakdowns remains an open challenge.

(a) Human expert "Shut and Open the Door" only once by casting SupplyDepotLower and SupplyDepotRaise to lure the Zerglings onto highlands and kill them.

(b) GPT-5 repeatedly "Shut and Open the Door" by casting multiple SupplyDepotLower and SupplyDepotRaise abilities without consistent intent.

Figure 5: Comparison of stratagem execution between human expert and GPT-5 on HLSMAC scenario *gmzz*.

Our proposed metrics serve as behavioral indicators of strategic reasoning rather than exhaustive measures. Beyond them, assessing ability usage efficiency and deception remains challenging, requiring analysis of timing, coordination quality, and complex behavioral patterns. A promising solution is a trigger-based approach using Galaxy SC2Bank APIs to export variables when key events occur, such as the enemy_lured flag for deception detection.

6 CONCLUSION

In this paper, we propose HLSMAC, the first StarCraft multi-agent challenge that systematically integrates the Chinese Thirty-Six Stratagems into high-level strategic decision-making scenarios. HLSMAC focuses on cooperative multi-agent decision-making, supports frameworks like PyMARL and LLM-PySC2, and introduces richer metrics beyond win rates to assess strategic performance. Our comprehensive experiments demonstrate that HLSMAC serves as a robust testbed for advancing multi-agent research.

Table 2: Win-rates across MARL baselines on all scenarios via broad evaluation. Inspired by the exploration-exploitation principle in reinforcement learning, we first conduct single-seed evaluation across all algorithms, then perform three-seed evaluation on the 9 top-performing methods (upper section, shown as mean ± std) identified from the initial round, with the remaining 12 algorithms (lower section) showing single-seed results.

Algorithm	adcc	dhls	fkwz	gmzz	jctq	jdsr	sdjx	swct	tlhz	wwjz	wzsy	yqgz
QTRAN[32][1]	**0.01±0.01**	0.00±0.00	0.00±0.00	**0.07±0.05**	0.00±0.00	**0.62±0.28**	**0.23±0.16**	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
CWQMIX[27][1]	0.00±0.00	0.00±0.00	0.00±0.00	**0.05±0.07**	0.00±0.00	**0.56±0.10**	**0.43±0.34**	0.00±0.00	0.00±0.00	**0.02±0.02**	**0.74±0.28**	0.00±0.00
DOP[48][6]	0.00±0.00	0.00±0.00	0.00±0.00	**0.12±0.11**	0.00±0.00	**0.99±0.01**	0.00±0.00	**0.13±0.07**	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
LICA[54][2]	0.00±0.00	0.00±0.00	0.00±0.00	**0.26±0.20**	0.00±0.00	**0.65±0.08**	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	**0.01±0.01**	0.00±0.00
Qatten[51][1]	0.00±0.00	**0.29±0.41**	0.00±0.00	**0.14±0.06**	0.00±0.00	**0.69±0.29**	**0.88±0.01**	**0.01±0.01**	0.00±0.00	0.00±0.00	**0.57±0.38**	0.00±0.00
QPLEX[43][1]	0.00±0.00	0.00±0.00	0.00±0.00	**0.06±0.08**	0.00±0.00	**0.68±0.21**	**0.29±0.40**	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.01	0.00±0.00
RIIT[9][2]	**0.88±0.06**	0.00±0.00	0.00±0.00	**0.20±0.15**	0.00±0.00	**0.90±0.06**	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
RESZ[30][8]	0.00±0.00	0.00±0.00	0.00±0.00	**0.39±0.10**	0.00±0.00	**0.77±0.07**	**0.84±0.08**	0.00±0.00	0.00±0.00	**0.53±0.38**	**0.02±0.03**	0.00±0.00
dTAPE[17][9]	**0.16±0.23**	0.00±0.00	0.00±0.00	**0.82±0.01**	0.00±0.00	**0.90±0.03**	**0.94±0.04**	**0.31±0.44**	0.00±0.00	**0.97±0.02**	**0.97±0.04**	0.00±0.00
IQL[36][1]	**0.30**	0.00	0.00	**0.04**	0.00	**0.40**	0.00	**0.03**	0.00	0.00	**0.34**	0.00
COMA[6][1]	0.00	0.00	0.00	0.00	0.00	**0.02**	0.00	0.00	0.00	0.00	0.00	0.00
VDN[33][1]	0.00	0.00	0.00	**0.12**	0.00	**0.60**	**0.14**	0.00	0.00	0.00	**0.09**	0.00
QMIX[28][1]	0.00	0.00	0.00	**0.01**	0.00	**0.88**	0.00	0.00	0.00	0.00	**0.85**	0.00
VMIX[34][2]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MAVEN[21][7]	0.00	0.00	0.00	0.00	0.00	**0.74**	0.00	0.00	0.00	0.00	0.00	0.00
OWQMIX[27][1]	0.00	0.00	0.00	0.00	0.00	**0.85**	0.00	0.00	0.00	0.00	0.00	0.00
FOP[52][5]	0.00	0.00	0.00	0.00	**0.03**	**0.73**	0.00	0.00	0.00	0.00	0.00	0.00
RODE[46][4]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ROMA[44][3]	0.00	0.00	0.00	0.00	0.00	**0.77**	0.00	0.00	0.00	0.00	0.00	0.00
RESQ[30][8]	0.00	0.00	0.00	**0.41**	0.00	**0.91**	0.00	0.00	0.00	0.00	0.00	0.00
sTAPE[17][9]	0.00	0.00	0.00	0.00	0.00	**0.98**	0.00	0.00	0.00	0.00	0.00	0.00

Baseline implementations adapted from these GitHub repos: [1]wqmix [26]. [2]pymarl2 [8]. [3]ROMA [45]. [4]RODE [47]. [5]FOP [53]. [6]DOP [49]. [7]MAVEN [22]. [8]RESQ [31]. [9]TAPE [18].

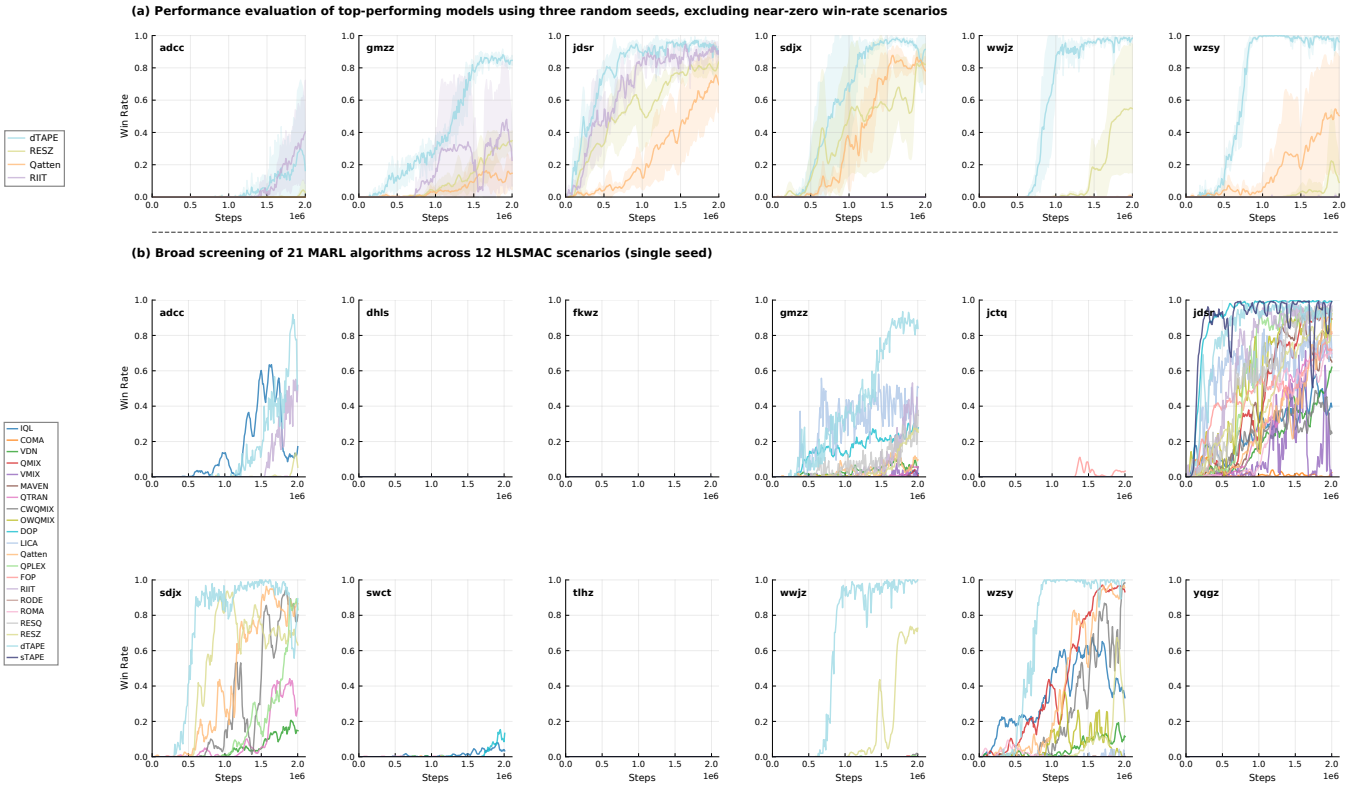


Figure 6: Win-rates across 21 MARL baselines on the 12 HLSMAC scenarios. Some figures show none or fewer curves because many algorithms achieve consistently poor performance (zero or near-zero win rates) during training.

REFERENCES

[1] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems* 32 (2019).

[2] Yue Deng, Yan Yu, Weiyu Ma, Zirui Wang, Wenhui Zhu, Jian Zhao, and Yin Zhang. 2024. SMAC-Hard: Enabling Mixed Opponent Strategy Script and Self-play on SMAC. arXiv:2412.17707 [cs.AI] https://arxiv.org/abs/2412.17707

[3] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. 2023. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=5OjLGiJW3u

[4] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science* 378, 6624 (2022), 1067–1074.

[5] Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2023. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems* 36 (2023), 7216–7262.

[6] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[7] Lei Han, Jiechao Xiong, Peng Sun, Xinghai Sun, Meng Fang, Qingwei Guo, Qiaobo Chen, Tengfei Shi, Hongsheng Yu, Xipeng Wu, and Zhengyou Zhang. 2021. TStarBot-X: An Open-Sourced and Comprehensive Study for Efficient League Training in StarCraft II Full Game. arXiv:2011.13729 [cs.AI] https://arxiv.org/abs/2011.13729

[8] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih wei Liao. 2021. Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning. https://github.com/hijkzzz/pymarl2.

[9] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih wei Liao. 2023. Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning. arXiv:2102.03479 [cs.LG] https://arxiv.org/abs/2102.03479

[10] Xuanfa Jin, Ziyan Wang, Yali Du, Meng Fang, Haifeng Zhang, and Jun Wang. 2024. Learning to discuss strategically: A case study on one night ultimate werewolf. *Advances in Neural Information Processing Systems* 37 (2024), 77060–77097.

[11] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2021. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251* (2021).

[12] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2019. Google Research Football: A Novel Reinforcement Learning Environment. *ArXiv* abs/1907.11180 (2019). https://api.semanticscholar.org/CorpusID:198899862

[13] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. 2019. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453* (2019).

[14] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037* (2017).

[15] Zongyuan Li, Yanan Ni, Runnan Qi, Lumin Jiang, Chang Lu, Xiaojie Xu, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, et al. 2024. Llm-pysc2: Starcraft ii learning environment for large language models. *arXiv preprint arXiv:2411.05348* (2024).

[16] Zongyuan Li, Yanan Ni, Runnan Qi, Lumin Jiang, Chang Lu, Xiaojie Xu, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, Huanyu Li, Hui Wu, Xian Guo, Kuihua Huang, and Xuebo Zhang. 2025. LLM-PySC2: Starcraft II learning environment for Large Language Models. arXiv:2411.05348 [cs.AI] https://arxiv.org/abs/2411.05348

[17] Xingzhou Lou, Junge Zhang, Timothy J Norman, Kaiqi Huang, and Yali Du. 2023. TAPE: Leveraging Agent Topology for Cooperative Multi-Agent Policy Gradient. *arXiv preprint arXiv:2312.15667* (2023).

[18] Xingzhou Lou, Junge Zhang, Timothy J Norman, Kaiqi Huang, and Yali Du. 2023. TAPE: Leveraging Agent Topology for Cooperative Multi-Agent Policy Gradient. https://github.com/LxzGordon/TAPE. *arXiv preprint arXiv:2312.15667* (2023).

[19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275 [cs.LG] https://arxiv.org/abs/1706.02275

[21] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. Maven: Multi-agent variational exploration. *Advances in neural information processing systems* 32 (2019).

[22] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. https://github.com/AnujMahajanOxf/MAVEN. , 7611–7622 pages.

[23] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. In *Advances in Neural Information Processing Systems*.

[24] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. arXiv:1703.10069 [cs.AI] https://arxiv.org/abs/1703.10069

[25] Rafael Pina, Varuna De Silva, Joosep Hook, and Ahmet Kondoz. 2022. Residual Q-Networks for Value Function Factorizing in Multi-Agent Reinforcement Learning. arXiv:2205.15245 [cs.LG] https://arxiv.org/abs/2205.15245

[26] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation. https://github.com/oxwhirl/wqmix.

[27] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:2006.10800 [cs.LG] https://arxiv.org/abs/2006.10800

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* 21, 178 (2020), 1–51.

[29] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[30] Siqi Shen, Mengwei Qiu, Jun Liu, Weiquan Liu, Yongquan Fu, Xinwang Liu, and Cheng Wang. 2022. ResQ: A Residual Q Function-based Approach for Multi-Agent Reinforcement Learning Value Factorization. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 5471–5483. https://proceedings.neurips.cc/paper_files/paper/2022/file/2456a42386e445ba884511aa17ca4a30-Paper-Conference.pdf

[31] Siqi Shen, Mengwei Qiu, Jun Liu, Weiquan Liu, Yongquan Fu, Xinwang Liu, and Cheng Wang. 2022. ResQ: A Residual Q Function-based Approach for Multi-Agent Reinforcement Learning Value Factorization. https://github.com/xmu-rl-3dv/ResQ.

[32] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5887–5896.

[33] Jianyu Su, Stephen Adams, and Peter A Beling. 2020. Value-Decomposition Multi-Agent Actor-Critics. *arXiv preprint arXiv:2007.12306* (2020).

[34] Jianyu Su, Stephen Adams, and Peter A. Beling. 2020. Value-Decomposition Multi-Agent Actor-Critics. arXiv:2007.12306 [cs.AI] https://arxiv.org/abs/2007.12306

[35] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. 2016. TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games. *arXiv preprint arXiv:1611.00625* (2016).

[36] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[37] Jordan Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 15032–15043.

[38] Rakshit Trivedi, Akbir Khan, Jesse Clifton, Lewis Hammond, Edgar A. Duéñez-Guzmán, Dipam Chakraborty, John P Agapiou, Jayd Matyas, Sasha Vezhnevets, Barna Pásztor, Yunke Ao, Omar G. Younis, Jiawei Huang, Benjamin Swain, Haoyuan Qin, Mian Deng, Ziwei Deng, Utku Erdoğanaras, Yue Zhao, Marko Tesic, Natasha Jaques, Jakob Nicolaus Foerster, Vincent Conitzer, Jose Hernandez-Orallo, Dylan Hadfield-Menell, and Joel Z Leibo. 2024. Melting Pot Contest: Charting the Future of Generalized Cooperative Intelligence. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=slqbOc67W8

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* 575, 7782 (2019), 350–354.

[40] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature* 575, 7782 (2019), 350–354.

[41] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. StarCraft II: A New Challenge for Reinforcement Learning. arXiv:1708.04782 [cs.LG] https://arxiv.org/abs/1708.04782

[42] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[43] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062* (2020).

[44] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. arXiv:2003.08039 [cs.MA] https://arxiv.org/abs/2003.08039

[45] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. https://github.com/TonghanWang/ROMA.

[46] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2020. RODE: Learning Roles to Decompose Multi-Agent Tasks. arXiv:2010.01523 [cs.LG] https://arxiv.org/abs/2010.01523

[47] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2020. RODE: Learning Roles to Decompose Multi-Agent Tasks. https://github.com/TonghanWang/RODE.

[48] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Off-Policy Multi-Agent Decomposed Policy Gradients. arXiv:2007.12322 [cs.LG] https://arxiv.org/abs/2007.12322

[49] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Off-Policy Multi-Agent Decomposed Policy Gradients. https://github.com/TonghanWang/DOP.

[50] Hua Wei, Jingxiao Chen, Xiyang Ji, Hongyang Qin, Minwen Deng, Siqin Li, Liang Wang, Weinan Zhang, Yong Yu, Lin Liu, Lanxiao Huang, Deheng Ye, Qiang Fu, and Wei Yang. 2022. Honor of Kings Arena: an Environment for Generalization in Competitive Reinforcement Learning. *ArXiv* abs/2209.08483 (2022). https://api.semanticscholar.org/CorpusID:250602029

[51] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. arXiv:2002.03939 [cs.MA] https://arxiv.org/abs/2002.03939

[52] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. 2021. FOP: Factorizing Optimal Joint Policy of Maximum-Entropy Multi-Agent Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12491–12500. https://proceedings.mlr.press/v139/zhang21m.html

[53] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. 2021. FOP: Factorizing Optimal Joint Policy of Maximum-Entropy Multi-Agent Reinforcement Learning. https://github.com/liyheng/FOP.

[54] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. 2020. Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. arXiv:2007.02529 [cs.LG] https://arxiv.org/abs/2007.02529

[55] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144* (2023).