School of Informatics

Informatics Research Review
Advancing Multi-Agent Reinforcement Learning through Hierarchical Paradigms

May 2024

Abstract

Exploring Multi-Agent Hierarchical Reinforcement Learning (MA HRL), this paper systematically reviews the literature, delving into the theoretical underpinnings of MA HRL and emphasizing its applicability to tasks decomposable into subtasks within semi-Markov decision processes. The review identifies HRL's advantages in handling high dimensionality, scalability needs, and scenarios involving partial observability. The proposed categorization of MA HRL methods, based on the type of abstraction employed in the hierarchy—Temporal Abstraction, State Abstraction, and Communication Abstraction—provides a systematic framework for understanding and comparing diverse approaches. This categorization aims to facilitate a nuanced exploration of MA HRL methods. Key criteria for practical adoption are identified, including scenarios with high environment dimensionality, a need for faster learning, scalability requirements, dynamic environments, challenges related to partial observability, and tasks conducive to inherent decomposition. This work aims to provide a comprehensive understanding of MA HRL, bridging theoretical foundations with practical applications. and to fill in a critical knowledge gap in the field, due to the lack of comprehensive literature surveys.

Date: Saturday 11th May, 2024

Supervisor: Zonglin Ji

1 Introduction

[1] Reinforcement learning is a machine learning paradigm focused on training agents to make sequential decisions in an unknown environment in order to maximize long-term rewards [1]. In reinforcement learning, the agent learns these actions by attempting to maximize the cumilative reward over each episode of the interaction. A fundamental goal of AI is to develop autonomous agents capable of executing diverse and intricate tasks in an environment through the formulation of optimal action sequences. However, a significant challenge encountered by reinforcement learning agents is the high dimensionality of the state and action spaces, hindering learning efficiency and practicality in real world applications, in other words this can be described as RLs "Curse of Dimensionality"[2].

This issue is further magnified in the case of multi-agent reinforcement learning due to the complexities of interactions among multiple autonomous agents in an environment. In a multi-agent system, each agent contributes its own high dimensional state and action space, resulting in an exponential increase in the overall joint state and action space[3]. The exponential growth in dimensionality can quickly make the problem intractable. Further, with increasing dimensionality, coordination and cooperation among agents becomes increasingly computationally challenging. Under these conditions, training of multi-agent systems can become prohibitive or severely limiting, making the issue of dimensionality a pressing concern in the field of multi-agent reinforcement learning.[4]

Hierarchical reinforcement learning(HRL) is one approach to mitigate the challenges posed by high dimensionality in multi-agent reinforcement learning. This strategy works by decomposing the problem into a hierarchy of subtasks. The higher level policies learn to accomplish tasks at a more abstract level, while lower level policies focus on specific actions in each subtask. The hierarchical structure allows for more efficient learning and decision-making, as it reduces the complexity of the overall joint state and action space, resulting in a dimensionality reduction. This structure also results in the decomposition of complex problems into multiple levels of abstraction, making it easier for agents to learn and coordinate their actions. By incorporating hierarchical structures and subgoal creation, the agents can leverage the environment's structure and organize their tasks in a way that improves policy learning. This strategy further results in a more efficient exploration of the state space and leads to a more flexible, adaptable and interpretable model. It also results in a more scalable approach, especially in the case of population based training paradigms consisting of a large number of agents.

Hierarchical methods in multi-agent reinforcement learning are especially significant due to the pressing issue of dimensionality, resulting in higher computational costs with larger and more complex tasks and environments. This review holds importance as it addresses an important challenge faced by researchers and practitioners in the field. By systematically examining and synthesizing existing literature on hierarchical methods in multi-agent reinforcement learning, it provides valuable insights into effective strategies for managing dimensionality in complex scenarios.

The primary hypothesis of this review is the effect of hierarchical multi-agent reinforcement learning on dimensionality reduction and further gauging the effects of the division of decision making in reference to the advancements in the relatively nascent field. Specifically, the review aims to evaluate the specific adaptations of hierarchical networks to the multi-agent reinforcement learning paradigm, providing an in-depth analysis of the current state of research in hierarchical MARL and a quantitative analysis of these methods on computational costs,

[1] All images in appendix

sample efficiency and overall system performance. Further, the review aims to explore the transferability of hierarchical systems in more practical aspects, as opposed to purely simulated environments, such as real-world autonomous driving systems or complex industrial automation processes. The review will draw on a range of sources to provide an in-depth analysis of the current state of research in hierarchical multi-agent reinforcement learning. By addressing these research questions, the literature review not only establishes a comprehensive understanding of the current state of hierarchical methods in MARL but also lays the groundwork for empirical investigations to validate the hypothesized benefits and uncover potential avenues for further refinement and development.

Looking at this intuitively, we can say that by introducing a hierarchical architecture, we are effectively increasing the model capacity while maintaining a separation of roles between each level of the network, i.e. with the higher levels focusing more on strategy and the lower levels focusing on, the finer details. This helps simplify complex tasks into more manageable subtasks by division of decision making and allows for greater flexibility. Hierarchical reinforcement learning is a promising approach to address the challenges of high dimensionality and computational cost in multi-agent systems.

2 Literature Review

The literature review section delves into the landscape of Multi-Agent Hierarchical Reinforcement Learning (MA HRL), providing a comprehensive exploration of its significance within the Multi-Agent Reinforcement Learning (MARL) paradigm. Despite the potential of hierarchical approaches in handling complex tasks, especially those arising in MARL, there is a notable absence of a comprehensive survey specifically dedicated to Multi-Agent HRL. This literature review aims to address this gap, presenting an extensive overview of existing works while proposing a categorization based on the type of abstraction employed in MA HRL algorithms. The examination navigates through key considerations, emphasizing the unique challenges posed by multi-agent scenarios and the strategic role of hierarchical structures in achieving more effective and scalable solutions.

Based on the review of the papers in this field, we propose a categorization of multi-agent hierarchical reinforcement learning methods, based on the type of abstraction implemented in the hierarchy:

1. **Temporal Abstraction:** Methods that introduce hierarchy by varying time granularity. This is most used in scenarios such as trading prices.
2. **State Abstraction:** Methods that introduce hierarchy through the abstraction of state space. This category is most used in scenarios such as wargame simulations, where command and control structures exhibit inherent hierarchical features not adequately captured by flat (non-hierarchical) algorithms.
3. **Communication Abstraction:** Methods that introduce a hierarchical structure into the communication structures between multiple agents, hence introducing an implicit hierarchy into the multi-agent system.

This section first outlines a brief history of the use/introduction of HRL, reviews the state-of-the-art, and explores other novel research in this field, categorized by the methods introduced above. Finally, some papers applying MA HRL in a practical setting are introduced.

2.1 History

The first hierarchical architecture reinforcement learning was by [5], or the Feudal Q network. This was inspired by societal feudal, or in more modern terms, managerial paradigms. The idea behind it was a basic hierarchical RL system based on state abstraction, where the manager interacts directly with the environment while only giving intrinsic rewards or subtasks to the worker nodes. The worker nodes are expected to complete the delegated subtask irrespective of its effect on the environmental reward. This can be viewed as a direct application of the dynamic programming rules of division and conquest [6] - finding smaller and simpler subtasks to break down a more complex problem. Feudal control has two main principles: reward hiding and information hiding. Reward hiding means that the reward is observed only by the higher-level agents. The lower-level agents focus solely on the subtask assigned, even if this might be suboptimal. Information hiding is the fact that the agent at any level of the hierarchy is aware only of the information from their level of granularity. This opens the possibility of learning simultaneously at multiple resolutions in space and time. Here, the non-Markovian nature of the higher levels was a concern and remains an issue in most hierarchical systems. However, there have been multiple ways these concerns have been mitigated in more modern algorithms, as will be seen ahead in this review.

The paper [7] attempts to solve the issue of a non-Markovian higher-level agent by introducing a MAXQ graph with separated Max nodes and Q nodes representing primitive actions and subtasks, respectively. A key principle of this is that the reward function of the parent task is the value function of the child task. This graph can be interpreted as the value function for a hierarchical policy. It is further extended to a multi-agent setting in [8]. In this case, the root node of the MAXQ graph is set up as a joint action value function, resulting in a cooperative MAXQ algorithm. The graphs for each agent can be separated out as well, resulting in a selfish MAXQ algorithm. This is tested on the Automated Guided Vehicle (AGV) task and the Trash Collection Task. The experiments (Figure 13) by the authors demonstrate that the multi-agent MAXQ performs significantly better than single-agent MAXQ. Furthermore, cooperative MAXQ results in better performance than selfish MAXQ, but the selfish algorithm learns faster. This can be attributed to the complexity of attempting to learn a joint action space as compared to several disjointed ones.

These three papers form the cornerstone of multi-agent hierarchical reinforcement learning, exerting a significant influence on contemporary algorithms, particularly under the category of state abstraction.

2.2 Temporal Abstraction

Another approach to incorporating a hierarchical design paradigm in MARL is to break down the problem into different time scales or levels of abstraction, known as temporal abstraction. This approach differs from most others, as it introduces different time scales for higher-level agents and lower-level agents instead of different scales in the state space or value space. In terms of implementation, this method differs from others examined here, as higher-level agents do not handle multiple lower-level agents. Hence, it can be construed as the standard multi-agent algorithms, with the distinction that each agent is a hierarchical agent with temporal abstraction. Very limited work exists in this sub-domain, and temporal abstraction, in general, represents a novel endeavor.

[9] implements hierarchical temporal abstraction on three different MARL paradigms: hierarchical independent learning, hierarchical communication network learner, and hierarchical QMIX

network. The independent learner is the standard independent Q learner algorithm applied to the hierarchical paradigm, where all the high-level and low-level gradients are updated independently of each other (as illustrated in Figure 12a). In hierarchical communication network, inspired by [10], a shared Comm module is placed between the hidden layers of the networks. It takes in the hidden layers of all the agents, and the same output then becomes an extra input to the next layers of all the agents (as shown in Figure 12b). In hierarchical QMIX, the standard QMIX algorithm [11] is implemented between the agents. Here, all the different MARL paradigms are applied to the higher-level agents. In hierarchical systems, the lower-level agents are relatively independent, allowing for parameter reuse and parameter sharing, further improving efficiency.

Two concerns in hierarchical MARL are the sparsity of transitions in the high-level learning tasks and the non-stationarity in multi-agent policies. Since high-level transitions are sparse, the sub-transitions/intermediate states are not utilized, resulting in low sample efficiency. High-level experiences, when replayed independently, may become outdated and provide inaccurate guidance owing to the non-stationary nature of the environment. To mitigate this, the authors proposed the Augmented Concurrent Experience Replay (ACER), consisting of two stages: experience augmentation and concurrent sampling. In experience augmentation, the sub-transitions at the higher level agents are added to the experience replay, resulting in a denser experience set (Figure 11a). To counter the issue of non-stationarity, the augmented experiences of agents are stored along the timestep axis (Figure 11b). Hence, when the experiences are sampled, concurrent experiences are sampled together instead of randomly.

The experiments were conducted in two multi-agent environments: Multi-Agent Trash Collection (MATC) and the Fever Basketball Defense (FBD)[12] environment. In MATC, hierarchical independent learning (h-IL) resulted in a much better average reward compared to Independent Deep Q Learning (IL-DQN). In FBD, in addition to h-IL resulting in better performance than IL-DQN, h-Comm outperforms h-IL and h-QMIX, although h-QMIX and h-Comm have comparable performance. This is attributed to the enablement of high-level communication in both these methods (Figure 10).

2.3 State Abstraction

In state abstraction, the hierarchical agents' level of abstraction is determined by the state space encapsulating the agents. Hence, higher-level agents possess a more abstract representation of the state, while lower-level agents operate with a more granular representation. This category stands out as one of the most widely employed techniques in hierarchical reinforcement learning.

2.3.1 Feudal Multi-Agent Hierarchical Network (FMH)

In [13], the authors extend the paper [5] from a single-agent model to a multi-agent method, incorporating deep neural networks as well. As in the original paper, the intuition is drawn from feudal hierarchies in human societies, where the manager or higher-level node sets tasks for the workers or lower-level nodes. Only the higher-level nodes or managers keep track of the environment rewards and disseminate equivalent intrinsic rewards down the chain.

The authors aim to address issues such as non-stationarity, scalability, and coordination. They implement a decentralized training approach to maintain generality in the experiments. There is no differentiable communication channel between the agents or a differentiable model of the environment. The hierarchy implemented here is in the form of a rooted directed tree (Figure

9), which is suitable for most cases. However, in certain cases, it can be any graph structure, as long as it is acyclic to avoid feedback cycles for the rewards. For example, there can be cases where a worker node reports to two or more managers. As in [5], the manager nodes define the reward functions for the worker nodes, and only the highest-level node will have access to the extrinsic environment rewards. Since the manager node handles goal setting, coordination is also dependent on the manager nodes.

To mitigate the issue of efficient communication between the manager and worker nodes, a bottom-up pertaining procedure is introduced. Here, the worker learns to follow the goals set by the manager before the training schedule of the manager. In accordance with a feudal architecture, the worker learns to always follow the manager's goals even in case of suboptimal solutions in the environment. Among the lower agents, parameter sharing is implemented due to the agents being identical. In this case, due to the similar nature, even with no parameter sharing, the performance remains the same, indicating that the agents learn a similar profile. The learning algorithm used here is the multi-agent deep deterministic policy gradient (MADDPG). Furthermore, a communication-repeat heuristic is introduced, which repeats the subgoals set by the manager for each agent for a specified number of times. This helps mitigate the stationarity problem, as initially, when the manager is untrained, it results in ephemeral subgoals, preventing the workers from being able to work effectively toward any single subgoal.

The experiments have been run on the cooperative communication environment, as well as a custom environment by the authors, cooperative coordination. In cooperative communication [14], the environment consists of a listener agent and a speaker agent. The listener agent has to navigate to one of the targets of a specified color. However, only the speaker agent can perceive the color, forcing a cooperation and communication paradigm between the agents. Cooperative coordination is a version of cooperative communication but with three listener agents and one speaker agent, and with decoy targets which the listeners have to avoid. In the cooperative communication environment, the addition of the FMH architecture to the MADDPG results in a better cumulative reward compared to both MADDPG and single-agent DDPG (Figure 6a). Furthermore, the higher-level agent or manager achieves a near 100% probability of navigating to the correct goal (Figure 6b). The same pattern repeats in the cooperative coordination environment, where the FMH architecture performs better than MADDPG and DDPG. The introduction of communication repeats also demonstrates faster convergence, although the cumulative rewards converge to approximately the same value (Figure 8). Scaling up, the performance improvement compared to MADDPG seems to improve, demonstrating that the FMH architecture is better suited for higher coordination tasks with a larger number of agents.

2.3.2 Dynamic Termination

One issue noted in most multi-agent hierarchical methods is the inflexibility of the agents, especially concerning extended duration options or dynamic options. This can be observed in the case of [8], where there may be a delay in the agent's reaction to modifications in the options selected by other agents. This delay is attributed to the fact that in [8], the agents become locked onto the current action until its termination and subsequent return of control to the parent node.

Premature termination is a potential solution to mitigate this issue. In this approach, the agents are interrupted every T timesteps, and the optimal actions are recalculated based on the current state of the system, both from the parent subtasks and the current subtask. This concept was introduced in single-agent settings in [15] to address the problem of imperfect options. However,

it cannot be directly applied in a multi-agent setting, as frequent termination and switching among agents could lead to diminished predictability of other agents' actions and, consequently, lower efficiency. Therefore, there is a need to strike a balance between sacrificing the agents' flexibility and maintaining predictability.

[16] proposes dynamic termination, wherein the option to terminate any agent's actions prematurely is incorporated into the higher-level controller. This controller manages the tradeoff between flexibility and predictability, with predictability being crucial in a cooperative multi-agent environment. This approach can represent all possible termination scenarios consistently with little additional computational overhead.

The authors implemented dynamic termination in the multi-agent taxi environment, similar to [8]. In this setup, a delayed communication channel is introduced, giving each agent access to all other agents' options. The agents attempt to estimate the joint Q value by choosing options that maximize the delayed Q value function. In the case of standard greedy termination, this would be:

$$o_{t-1}^{-j} := (o_{t-1}^1, ..., o_{t-1}^{j-1}, o_{t-1}^{j+1}, ..., o_{t-1}^n)$$

$$Q^j(s_t, o_{t-1}^{-j}, o_t^j) := \mathbb{E}[r_t + \gamma U^j(s_{t+1}, o_t^{-j}, o^j)]$$

where, o_t^j represents the options available to agent j at timestep t, and

$$U^j(s_{t+1}, o_t^{-j}, o^j) := (1 - \beta^{o^j}(s_{t+1}))Q^j(s_{t+1}, o_t^{-j}, o^j) + \beta^{o^j}(s_{t+1})max_{o'^j \in O^j} Q^j(s_{t+1}, o_t^{-j}, o'^j)$$

In this equation, U is the TD target, setting the target to choose the next optimal action if o is terminating; otherwise, it continues with the same o as in the previous timestep. β represents the termination condition, i.e., 0 if terminating and 1 otherwise.

In greedy termination, an individual agent's performance improves due to added flexibility. However, its behaviour becomes less predictable to other agents. The authors make the termination dynamic by adding a negative reward δ on the decision to terminate, including it as part of the agent's policy. This results in the modified, dynamic termination Bellman equation:

$$Q^j(s_t, o_{t-1}^{-j}, o_t^j \neq T) := \mathbb{E}[r_t + \gamma U^j(s_{t+1}, o_t^{-j}, o'^j)]$$

$$Q^j(s_t, o_{t-1}^{-j}, o_t^j = T) := max_{o'^j \in O^j} Q^j(s_{t+1}, o_{t-1}^{-j}, o'^j) - \delta$$

Hence, each agent's actions are based on the current state and the last known options of all other agents. As seen from the above equation, a termination action would improve the agent's performance, but this improvement is offset by the added negative reward. Therefore, the termination becomes learnable by the agent, striking a balance between flexibility and predictability. The delayed action joint Q value is then updated by temporal difference learning with experience replay.

The authors demonstrate empirically that the introduction of dynamic termination significantly improves the agent performance (Figure 5) in multiple cooperative multi-agent environments, namely the multi-agent taxi tasks[17] and the multi-agent pursuit[17].

2.3.3 Hierarchical Cooperative MARL with Skill Discovery

In hierarchical systems, the higher-level agent is utilized to obtain a more abstract view of the environment and act upon it, facilitating cooperative learning. The lower-level agent, with more granular control, is better suited for independent learning. This constitutes a core aspect of

the work done in [18]. Inspired by team sports, the authors propose a two-level architecture, with independent Q learning at the lower level and a centralized multi-agent training paradigm at the higher level. This is akin to coaches training individual players in different skills at the team level, where the players exert low-level control to achieve the target using their skills.

To achieve this, the authors implement a high-level action space as a set of latent variables, with the policy able to choose a latent variable and with a centralized training routine. Hence, resulting in an "unsupervised" approach to skill discovery, as opposed to hard-coding the skills into the architecture. The reward target is split up, with the extrinsic or environmental reward being used to train the higher level and an intrinsic reward being used to train the lower level, in a decentralized manner. The intrinsic reward is defined as a decoder which predicts the ground truth latent variable, as set by the higher level control, from the trajectories generated by the lower level control (i.e., $p(z|\tau$ in Figure 3). In other words, the intrinsic reward attempts to display how close the lower level trajectory is to the skill set assigned to it.

Hence, the system is a bilevel optimization problem,

$$\max_{\mu,\pi} \mathbb{E}_{z\sim\mu,P}[\mathbb{E}_{s_t,a_t\sim\pi,P}[\sum_{t=1}^{T}\gamma^t R(s_t,a_t)]]$$

$$\pi^n \in_\pi \mathbb{E}_{z\sim\mu,P}[\sum_{k=1}^{K}\mathbb{E}_{\tau_k^n\sim\pi,P}[R_L(z_k^n,\tau_k^n)]] \quad \forall n \in [N]$$

where, R is the extrinsic reward function and R_L is the intrinsic reward function.

z and Z are the action and action space for the higher level policy, and similarly a and A are for the lower level action space.

μ is the joint high-level policy, optimizing the extrinsic reward, and π is the joint low-level policy.

The time steps are further partitioned as $T = kt_{seg}$ where T is the length of the entire episode, i.e., the episode is partitioned into k episodes, and the higher level action is taken at these k points. Hence, this can be thought of as an extension of the dynamic termination discussed in [16], i.e., the higher level controller intermittently intervenes and can alter the skill set attributed to the lower level agents, effectively bringing about flexibility without compromising predictability.

The decoder for the intrinsic rewards is trained on the dataset $D = \{(z,\tau)\}$, hence the training of p_ψ becomes a supervised learning problem on the data accumulated during online training. The intrinsic reward is defined as a combination of the joint reward R and the decoder p_ψ:

$$R_L(z^n,\tau^n) := \alpha \sum_{s_t,a_t\in\tau^n} \gamma^t R(s_t,a_t) + (1-\alpha)p_\psi(z^n|\tau^n)$$

here, τ is the trajectory of the agent.

Here, the Simple Team Sports Simulator (STS2)[19] environment is used for implementing the algorithm. The STS2 captures the high-level rules and dynamics of N versus N team sports while abstracting away the non-strategic elements of gameplay. The experiments run on the environment demonstrate that the HSD (Hierarchical Skill Discovery) agent learns faster than QMIX [11] and IQL, although the asymptotes converge to the same performance within a margin of error (Figure 4). Also, the low performance of the HSD-ext algorithm, with no access to extrinsic reward, supports the hypothesis of the requirement of extrinsic rewards. HSD-scripted, using cooperative training at the higher level, gives better overall performance.

2.3.4 Intrinsic Reward Rectification

Building on the previous paper [18], HRL is a promising approach for complex multi-agent tasks, wherein the higher-level policy guides the training process of the lower-level policies using intrinsic rewards and macro actions (actions selected by the higher control level). However, in this context, the lower-level agents do not consider the impact of macro actions on decision-making. In other words, if the macro action has less impact on the decisions, and hence, the higher-level policy does not significantly affect overall decision-making, it naturally follows that the intrinsic reward should be lowered. This is the hierarchical credit assignment problem. In [20], the authors propose an intrinsic reward rectification method to handle this exact scenario and correct it depending on the macro actions.

This work is based on the HAVEN framework [21], known as the Hierarchical Value Decomposition with Rectification (HVDR), a standard hierarchical multi-agent RL algorithm as discussed in the previous section [18], but using the advantage function $A(s_t, a_t^h)$ as the intrinsic reward. Here, the QMIX value decomposition function is used in both the high and low-level policy, i.e., including an agent network and a mixing network.

In this paper, the authors suggest an improvement over the HAVEN algorithm by adding an additional intrinsic reward rectification function. Hence, the intrinsic reward now consists of the rectification net in addition to the advantage net from the HAVEN algorithm (Figure 2).

The rectification net here corrects the advantage values generated by the advantage net [20] to make the intrinsic reward proportional to the impact of macro actions on the lower-level agents. This can be represented as a posterior probability distribution of the ground truth macro actions (derived from the higher-level agents) conditioned on the trajectory at the next macro time $P(a_T^{h,i}|\tau_{T:T+1}^i)$. The joint posterior probability would be the product of all the agents' posteriors:

$$P(a_T^h|\tau_{T:T+1}) = \prod_{i=1}^{n} P(a_T^{h,i}|\tau_{T:T+1}^i)$$

Instead, we take a simple average to prevent high instability due to performance variation across different random seeds, since the dataset still draws from online RL episodes.

$$P(a_T^h|\tau_{T:T+1}) = \frac{\sum_{i=1}^{n} P(a_T^{h,i}|\tau_{T:T+1}^i)}{n}$$

Hence, the rectified intrinsic reward function is calculated by using this posterior as an adaptive weight on the advantage function:

$$r_T^I = p(a_T^h|\tau_{T:T+1})A(s_T, a_T^h)$$

The algorithm has been tested on the StarCraft Multi-Agent Challenge environment[22], on the hard (*3s vs 5z* and *5m vs 6m*) and super hard (*27m vs 30m* and *MMM2*). As shown in Figure 1, the HVDR outperforms the state-of-the-art value decomposition algorithms on all of the selected maps. Further, ablation studies also empirically establish the significance of intrinsic reward rectification in performance improvements.

2.4 Communication Abstraction

In communication abstraction, the hierarchical structure is manifested in the way agents communicate. This introduces an implicit hierarchy in the multi-agent system, where communication

is organized in a structured manner. Imagine it as creating tiers of communication, allowing agents to share information more efficiently. This could prove beneficial when there isn't a predefined hierarchy, but having an organized way for agents to exchange information enhances their collaborative efforts. This is again a relatively unexplored subfield.

2.4.1 Hierarchical Graph Attention

In [23], the authors propose the Hierarchical graph Attention-based Multi-Agent actor-critic (HAMA), integrating representation learning and policy derivation for multi-agent systems. It combines a representation learning framework, utilizing a hierarchical graph attention network (HGAT) for processing states represented as a graph, with a multi-agent actor-critic network for deriving decentralized policies for each of the individual agents.

In the representation learning phase, HAMA employs HGAT to process the state graph, which represents the global state of the game in a Partially Observable Markov Game (POMG) setting. The HGAT comprises a network stacking multiple Graph Attention Networks (GATs) hierarchically (Figure 15). In other words, instead of a hierarchical stacking of agents, there is a stacking of the communication network between the agents, facilitating multi-faceted communication. This clusters the agents into distinct groups using prior knowledge, allowing for effective modelling of the hierarchical inter-agent and inter-group relationships. This method is more effective in the case of a hybrid competitive cooperative environment. The hierarchical state representation obtained using HGAT is instrumental in mixed cooperative-competitive games, providing contextualized state representation for individual agents.

The multi-agent actor-critic network (note that the model here is not hierarchical), uses the Hierarchical Graph Attention network (HGAT), to adaptively extract state-dependent relationships among multiple agents. The attention mechanisms guide each agent's focus on specific groups, enabling flexible adaptation to cooperative or competitive scenarios. This method showcases advantages in terms of scalability, interpretability, and transferability of learned policies, addressing key challenges in multi-agent reinforcement learning.

The experiments, performed on a variety of multi-agent cooperative-competitive environments, such as Cooperative Navigation[24], 3 vs 1 Predator-Prey, 3 vs 3 Predator-Prey[17], and The More-The Stronger games, demonstrate the algorithm's superior performance. This can be seen from Figure 14, where the HAMA algorithm receives lower penalties on the cooperative navigation environment as compared to other state-of-the-art multi-agent algorithms.

2.5 Practical Applications of Multi-Agent HRL

Theoretically, the Multi-Agent HRL paradigm can be applied to any task M where the task can be decomposed into a finite set of subtasks $\{M_0, ..., M_{m-1}\}$ consisting of semi-Markov decision processes. These subtasks can be cooperative or non-cooperative. Intuitively, it can be said that any task that has an inherent hierarchy along which the tasks can be split and abstracted out can effectively use HRL. These splits can be in temporal abstraction, for example, varying the time granularity in trading prices, or can be in the state space abstraction, for example, in a wargame scenario where a command and control structure has an inherent hierarchical structure not captured by most flat algorithms. However, implementing multi-agent HRL brings its own drawbacks, including communication between different levels of agents, high non-stationarity due to the additional agent interactions compared to a simple flat multi-agent algorithm. In short, the introduction of HRL would bring about certain complications owing to the additional

complexity of the system. Hence, practically HRL should be looked at only when:

- **High Environment Dimensionality**: The flat multi-agent system has very high dimensionality, owing to the state spaces and the number of agents. The introduction of a hierarchical system can control the dimensionality of the system [18].
- **Faster Learning**: When faster learning is crucial, or the flat algorithms converge very slowly.
- **Scalability**: When scalability is a requirement, abstracting the higher-level tasks away leaves a simplified lower-level agent, hence making it easier to scale up the agents with significantly lower overhead.
- **Dynamic Environments**: In the case of dynamic environments, HRL provides a flexible framework for agents to adapt and reorganize their strategies.
- **Partial Observability**: When partial observability hinders performance. The higher-level agents have greater observability due to the abstraction of either the state space or temporal space.
- **Task Decomposition**: Inherent task decomposition and specificity, where the tasks can be decomposed into a hierarchy of specific subtasks that contribute to the overall goal.
- **Cooperative Tasks**: For multi-agent HRL to be applied, there requires at least some element of cooperation involved within the tasks, i.e., can be a hybrid of cooperative and competitive tasks or can be purely cooperative. However, in the case that the task is purely competitive, the MA HRL decomposes into a single-agent HRL with multi-agent interactions, wherein all the levels of agents would interact among one another in the same manner.

Some of the successful implementations of multi-agent HRL in practical applications are:

- Multi-agent dispatching logistics: Cooperative multi-agent system, working towards the management and optimization of dispatching goods, services, or resources. For example, in [25], the dispatch of repair crews in a microgrid environment, such as gas, electricity, or transportation, is handled using hierarchical multi-agent RL, with the higher level handling the grid features and the lower level agent handles the routing and repair decision for the individual crews. Another example of this, albeit not entirely in a practical setting, would be the use of the multi-agent taxi environment in many of the papers reviewed in the sections above. Another implementation of this is handling the traffic interactions in a fleet of autonomous vehicles, as implemented in [26]. Similarly, in [27], it has been used in a traffic control scenario but not specific to autonomous vehicles.
- Wargaming Scenarios: Command and control scenarios, simulating a mix of competitive and cooperative interactions in the environments. In both [28] and [29], multi-agent HRL has been implemented to simulate the command and control for multi-agent close-range air combat and air combat maneuvering respectively. Here the algorithm obtains a high-level strategy for the target and a low-level strategy for selection. In [29], the hierarchical split is between a high-level commander policy and a low-level individual unit policy. Similarly, [30] applies multi-agent HRL in the intelligent game environment, which is a wargame simulation.

- Resource Utilization and Optimization: Optimization of a limited set of resources. This is especially suited to HRL due to the high environment complexity and dimensionality. In [31], the computational offloading in multi-access edge computing is handled using multi-agent HRL. The higher-level agent handles the offloading location selection problem, and the lower-level agents handle the offloading ratio problem.

- A rare yet very interesting application involves the use of multi-agent HRL in [32], where the paper introduces a decentralized auto-focusing system for cameras utilizing a hierarchical approach. The environment in this scenario is highly dynamic, as the nature of photos and conditions can change relatively quickly. The high-level agent is responsible for managing camera settings, evaluating image histograms, and computing rewards for the lower levels. On the other hand, the low-level agent executes specific actions such as adjusting lens position, capturing images, and detecting objects, thereby having a direct interaction with the environment information.

3 Summary & Conclusion

In conclusion, the exploration of hierarchical reinforcement learning (HRL) in multi-agent systems reveals essential patterns and notable advancements, and this paper is able to lay out a chronological development pattern for all the different advancements in this niche field. This paper also introduced a novel method of categorizing the different multi-agent HRL algorithms based on the type of abstraction utilized here.

Foundational papers, including the Feudal Q network ([5], MAXQ [7]), and Temporal Abstraction [9], have left a lasting impact on contemporary algorithms, particularly those emphasizing state abstraction. The Feudal Q network introduced a fundamental hierarchical RL structure based on state abstraction, drawing inspiration from societal feudal paradigms. MAXQ addressed the challenge of non-Markovian higher-level agents by introducing a graph structure with distinct Max nodes and Q nodes, showcasing improved performance in multi-agent settings. Temporal abstraction, exemplified in [9], presented a novel approach by dissecting problems into different time scales, enabling agents to learn simultaneously at multiple resolutions.

Subsequent papers, such as the Feudal Multi-Agent Hierarchical Network [13], Dynamic Termination [16], Hierarchical Cooperative MARL with Skill Discovery [18], and Intrinsic Reward Rectification [20], have extended the capabilities of HRL. FHR introduced a decentralized training approach, addressing non-stationarity, scalability, and coordination challenges. Dynamic termination brought flexibility by allowing agents to terminate actions prematurely, enhancing performance in multi-agent environments. Hierarchical Cooperative MARL with Skill Discovery proposed a two-level architecture inspired by team sports, achieving enhanced learning efficiency. Intrinsic Reward Rectification improved the HAVEN algorithm by incorporating a rectification net, adjusting intrinsic rewards based on the impact of macro actions.

Further the Hierarchical Multi Agent Graph Attention[23] (HAMA) integrated a Hierarchical Graph Attention network (HGAT) for state representation with a multi-agent actor-critic network, achieving superior scalability, interpretability, and transferability, particularly in hybrid cooperative-competitive environments, as demonstrated through experiments in various challenging scenarios. This demonstrated the use of hierarchical communication abstraction to induce a hierarchy in a simple multi agent environment.

Despite these advancements, challenges persist. The hierarchical credit assignment problem, sparsity of transitions in high-level tasks, and non-stationarity in multi-agent policies pose

hurdles. Further, the trade-off between flexibility and predictability in dynamic termination and the necessity for extrinsic rewards in skill discovery highlight areas that warrant further investigation.

Theoretical foundations establish its applicability to tasks decomposable into subtasks, whether cooperative or non-cooperative, providing a framework for effective hierarchical abstraction. However, practical implementation introduces challenges such as agent communication, increased non-stationarity, and added complexity. Therefore, it is advisable to consider MA HRL when faced with high environment dimensionality, faster learning, scalability requirements, dynamic environments, partial observability, task decomposition, and cooperative task elements. Successful applications in multi-agent dispatching logistics, wargaming scenarios, and resource optimization demonstrate its effectiveness in addressing diverse real-world challenges ([25], [28], [29], [31]). Despite the inherent complications, MA HRL emerges as a valuable tool for enhancing multi-agent system capabilities and navigating hierarchical complexities in tasks.

Future research should focus on addressing these challenges and enhancing scalability, coordination, and adaptability in complex multi-agent scenarios. Exploring the application of HRL in real-world domains and evaluating its robustness in dynamic environments will contribute to the ongoing evolution of hierarchical reinforcement learning in multi-agent systems.

References

[1] Xiang Jin, Wei Lan, and Xin Chang. Neural path planning with multi-scale feature fusion networks. *IEEE Access*, 10:118176–118186, 2022.

[2] RICHARD BELLMAN. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[3] Patrick Phillips. Reinforcement learning in two player zero sum simultaneous action games, 2021.

[4] Annie Wong, Thomas Bäck, Anna V. Kononova, and Aske Plaat. Deep multiagent reinforcement learning: challenges and directions. *Artificial Intelligence Review*, 56(6):5023–5056, Jun 2023.

[5] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.

[6] Satinder Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 05 1992.

[7] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *CoRR*, cs.LG/9905014, 1999.

[8] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In *Proceedings of the Fifth International Conference on Autonomous Agents*, AGENTS '01, page 246–253, New York, NY, USA, 2001. Association for Computing Machinery.

[9] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. Hierarchical deep multiagent reinforcement learning. *CoRR*, abs/1809.09332, 2018.

[10] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2252–2260, Red Hook, NY, USA, 2016. Curran Associates Inc.

[11] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. *CoRR*, abs/1803.11485, 2018.

[12] Hangtian Jia, Yujing Hu, Yingfeng Chen, Chunxu Ren, Tangjie Lv, Changjie Fan, and Chongjie Zhang. Fever basketball: A complex, flexible, and asynchronized sports game environment for multi-agent reinforcement learning, 12 2020.

[13] Sanjeevan Ahilan and Peter Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning. *CoRR*, abs/1901.08492, 2019.

[14] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc.

[15] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112:181–211, 1999.

[16] Dongge Han, Wendelin Boehmer, Michael Wooldridge, and Alex Rogers. Multi-agent hierarchical reinforcement learning with dynamic termination, 2019.

[17] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[18] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *CoRR*, abs/1912.03558, 2019.

[19] Yunqi Zhao Igor Borovikov Jiachen Yang Ahmad Beirami Caedmon Somers, Jason Rupert. Simple Team Sports Simulator (STS2), February 2020.

[20] Zhihao Liu, Zhiwei Xu, and Guoliang Fan. Hierarchical multi-agent reinforcement learning with intrinsic reward rectification. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[21] Zhiwei Xu, Yunpeng Bai, Bin Zhang, Dapeng Li, and Guoliang Fan. HAVEN: hierarchical cooperative multi-agent reinforcement learning with dual coordination mechanism. *CoRR*, abs/2110.07246, 2021.

[22] 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. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[23] Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-agent actor-critic with hierarchical graph attention network. *CoRR*, abs/1909.12557, 2019.

[24] Corban Rivera, Edward Staley, and Ashley Llorens. Learning multi-agent cooperation. *Frontiers in Neurorobotics*, 16, 2022.

[25] Dawei Qiu, Yi Wang, Tingqi Zhang, Mingyang Sun, and Goran Strbac. Hierarchical multi-agent reinforcement learning for repair crews dispatch control towards multi-energy microgrid resilience. *Applied Energy*, 336:120826, 2023.

[26] Xiaoyuan Fu, Quan Yuan, Shifan Liu, Baozhu Li, Qi Qi, and Jingyu Wang. Communication-efficient decision-making of digital twin assisted internet of vehicles: A hierarchical multi-agent reinforcement learning approach. *China Communications*, 20(3):55–68, 2023.

[27] Pengqian Zhao, Yuyu Yuan, and Ting Guo. Extensible hierarchical multi-agent reinforcement-learning algorithm in traffic signal control. *Applied Sciences*, 12(24), 2022.

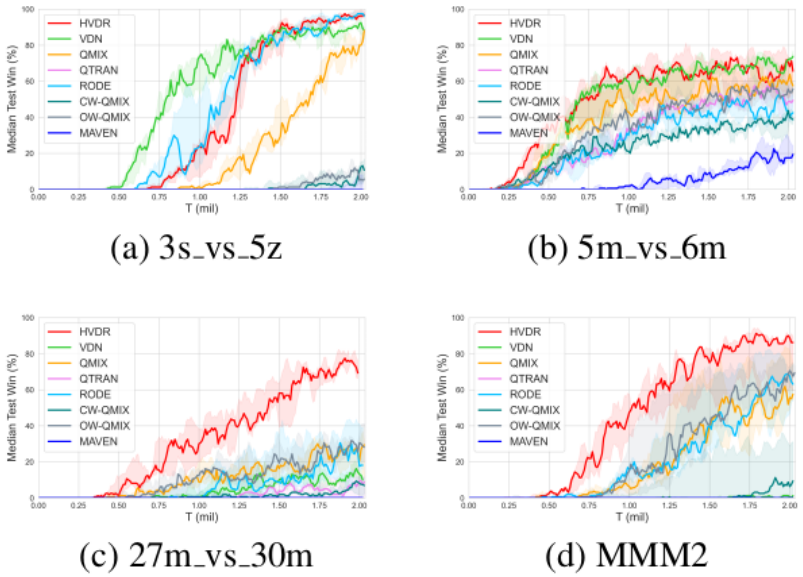
[28] Wei-ren Kong, De-yun Zhou, Yong-jie Du, Ying Zhou, and Yi-yang Zhao. Hierarchical multi-agent reinforcement learning for multi-aircraft close-range air combat. *IET Control Theory & Applications*, 17(13):1840–1862, 2023.

[29] Ardian Selmonaj, Oleg Szehr, Giacomo Del Rio, Alessandro Antonucci, Adrian Schneider, and Michael Rüegsegger. Hierarchical multi-agent reinforcement learning for air combat maneuvering, 2023.

[30] Bin Li. Hierarchical architecture for multi-agent reinforcement learning in intelligent game. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[31] Yu Sun and Qijie He. Computational offloading for mec networks with energy harvesting: A hierarchical multi-agent reinforcement learning approach. *Electronics*, 12(6), 2023.

[32] Anna Anikina, Oleg Y. Rogov, and Dmitry V. Dylov. Detect to focus: Latent-space autofocusing system with decentralized hierarchical multi-agent reinforcement learning. *IEEE Access*, 11:85214–85223, 2023.



(a) 3s_vs_5z (b) 5m_vs_6m

(c) 27m_vs_30m (d) MMM2

Figure 1: Win rates of HVDR versus multi agent state of the art value decomposition algorithms[20].

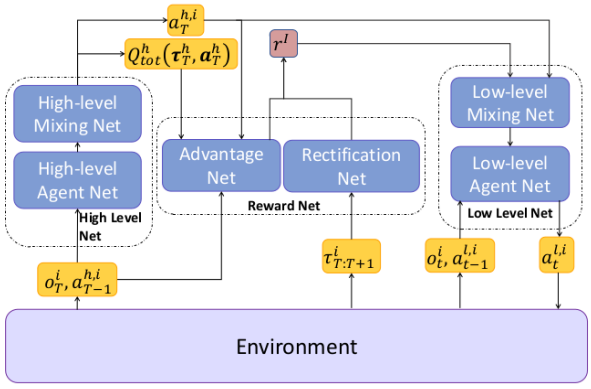


Figure 2: Architecture of HVDR algorithm[20]

4 List of Figures

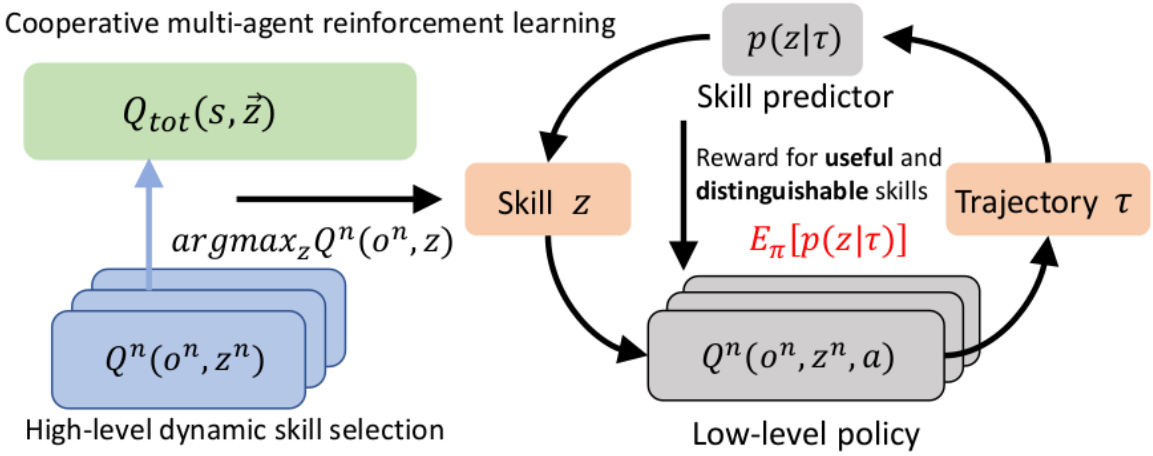
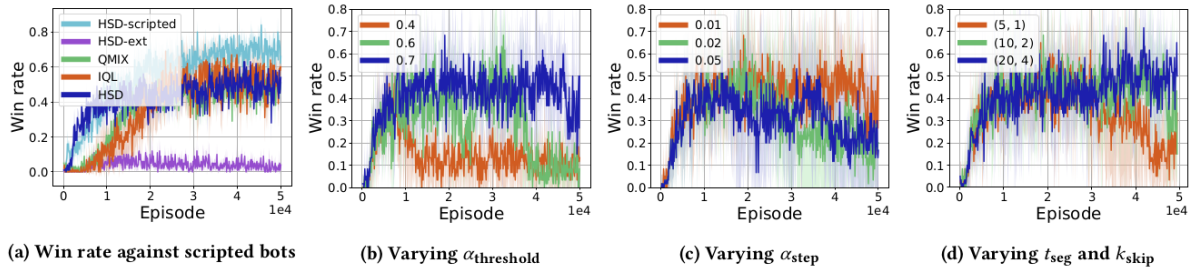
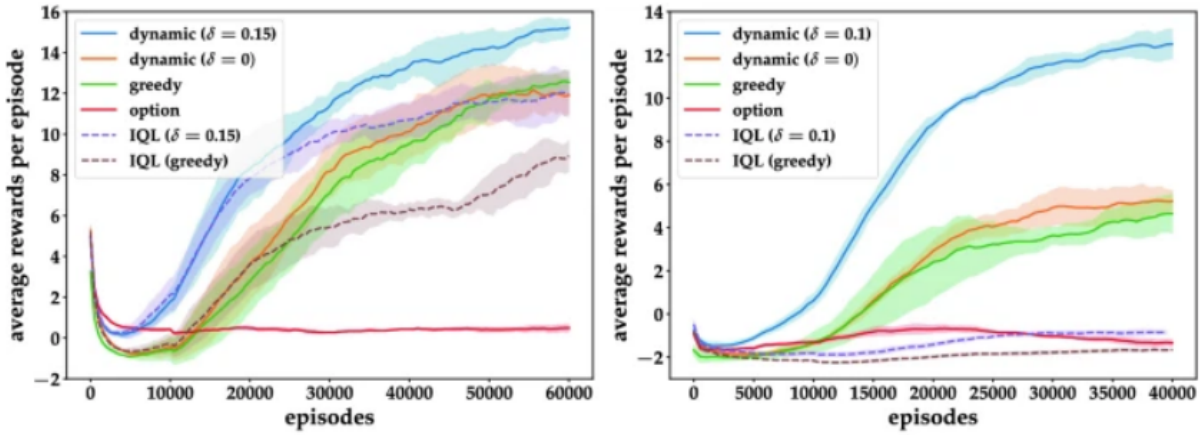


Figure 3: Architecture of the high level and low level hierarchical policies.[18]



(a) Win rate against scripted bots (b) Varying $\alpha_{\text{threshold}}$ (c) Varying α_{step} (d) Varying t_{seg} and k_{skip}

Figure 4: Win rate against scripted opponent team over training episodes.[18]



(a) 19 × 19 taxi with 10 agents (b) 19 × 19 pursuit with 10 agents

Figure 5: Results on the multi agent taxi pickup and the multi agent pursuit environments[16]. (IQL - Independent Q Learning)

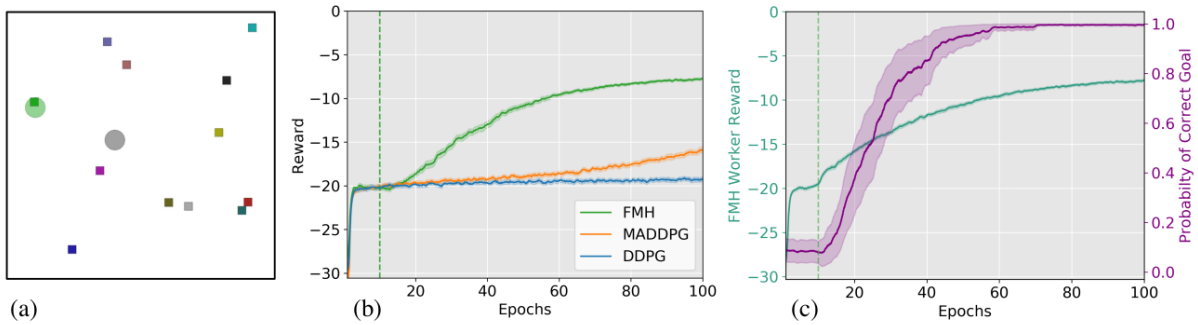


Figure 6: Performance on cooperative communication environment with one listener and twelve targets[13]

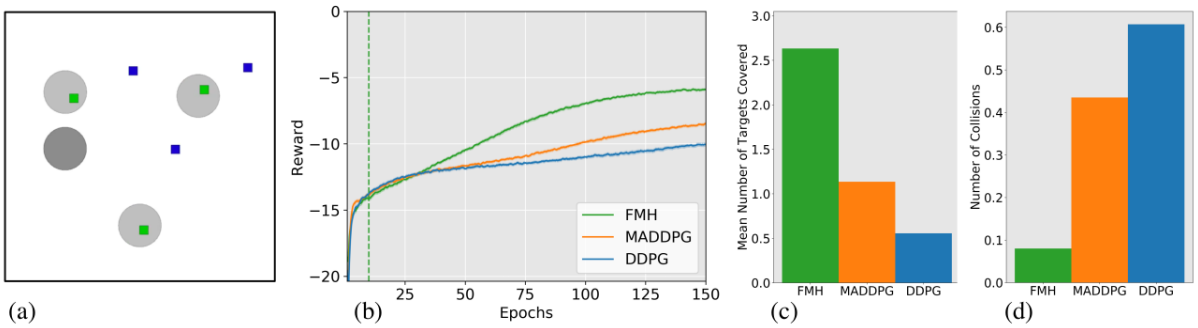


Figure 7: Performance on cooperative coordination environment with three listeners, three green targets and three decoy blue targets[13]

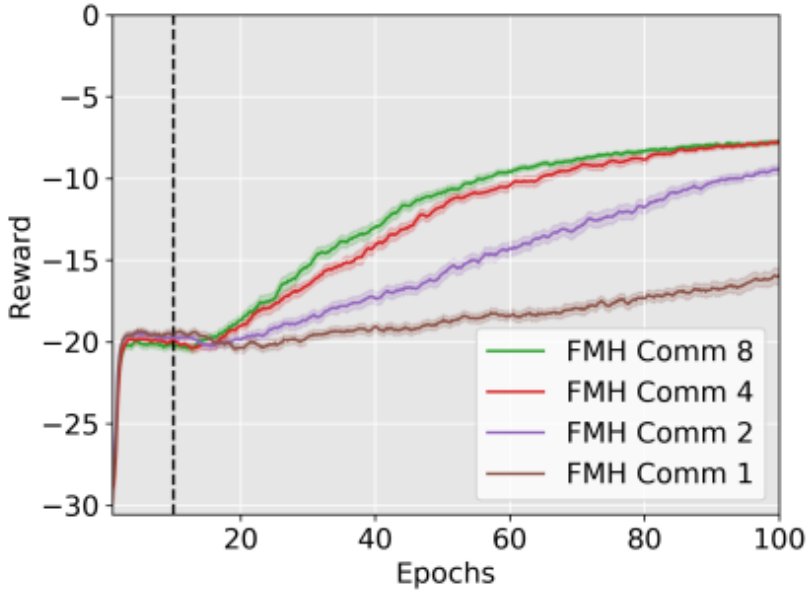


Figure 8: Performance on cooperative coordination environment with different communication repeat hyperparameters. For example, FMH-Comm1 means no communication repeats, FMH-Comm2 means 1 communication repeat, etc.[13]

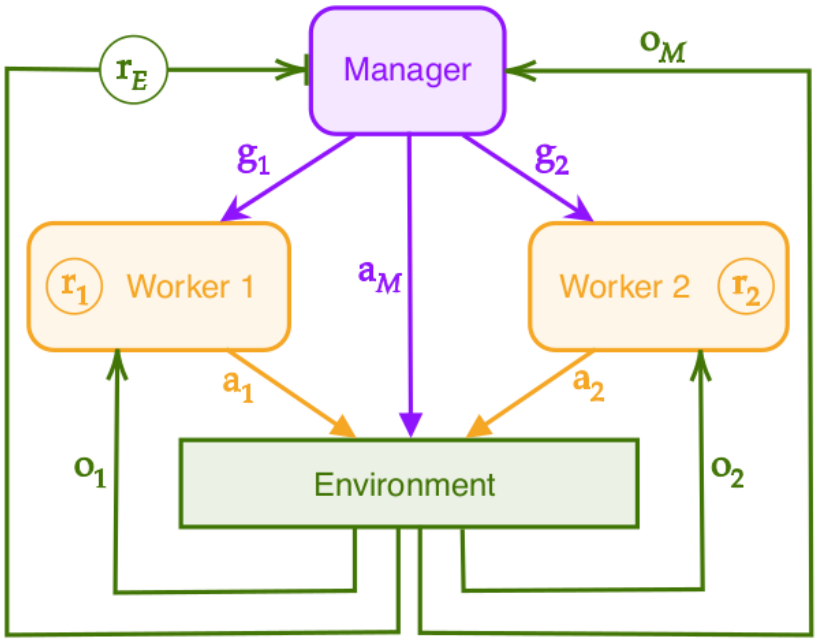
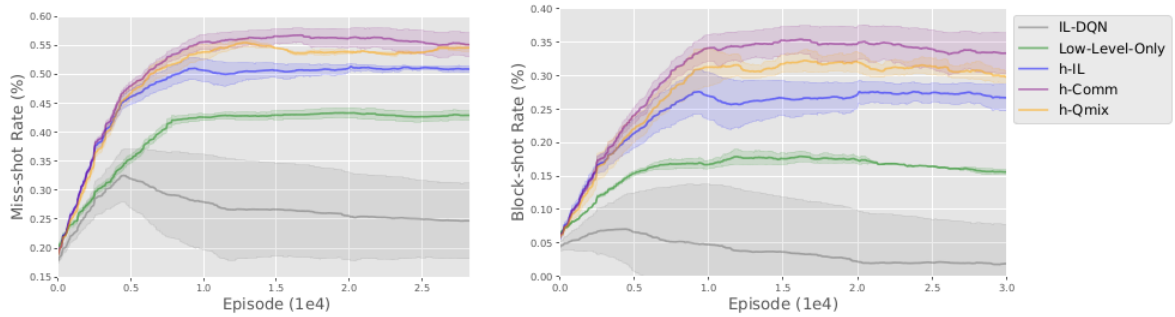
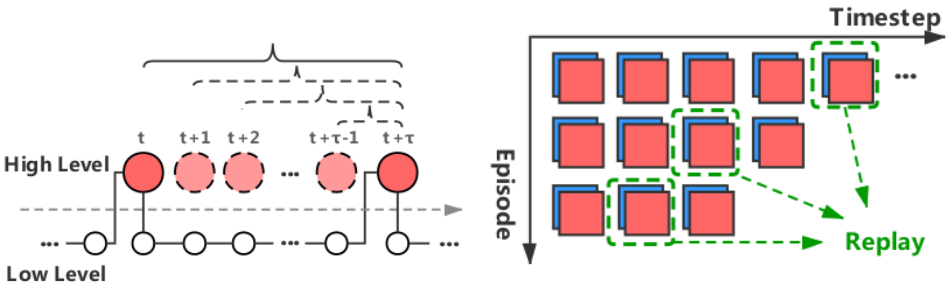


Figure 9: Feudal Multi Agent Hierarchy with one manager agent and two worker agents. Here the hierarchy structure is a rooted directed tree.[13]



(a) (b)

Figure 10: Performance of different approaches on FBD[9]



(a) (b)

Figure 11: Augmented Concurrent Experience Replay (ACER) a) Experience Augmentation b) Concurrent Sampling[9]

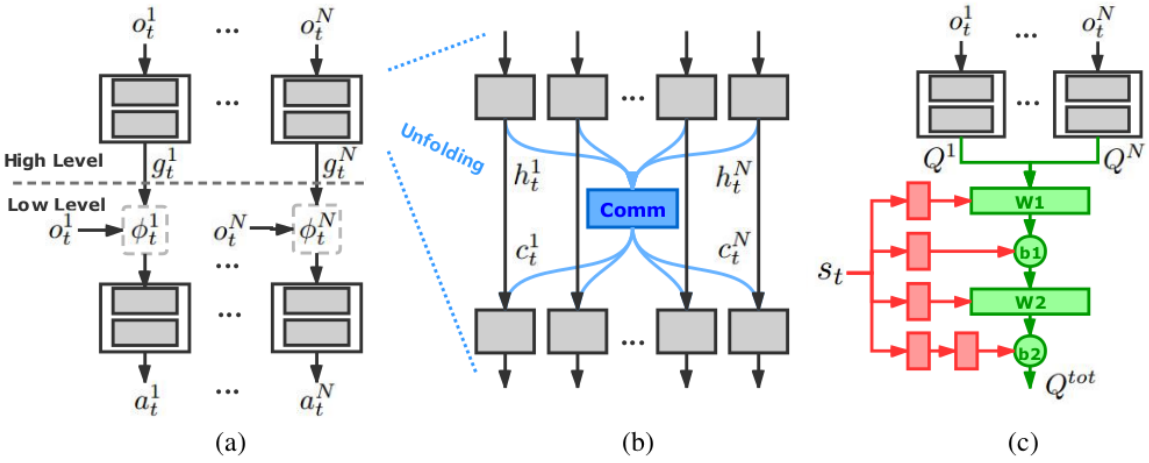


Figure 12: Hierarchical Deep MARL Paradigms a) Hierarchical Independent Learner b) Hierarchical Comm Architecture c) Hierarchical QMIX Architecture[9]

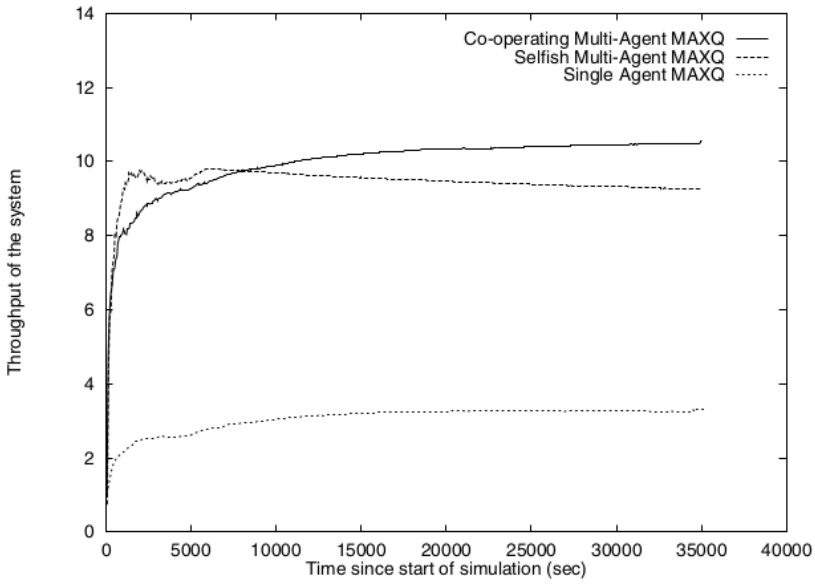


Figure 13: MAXQ variants performance in AGV travel time[8]

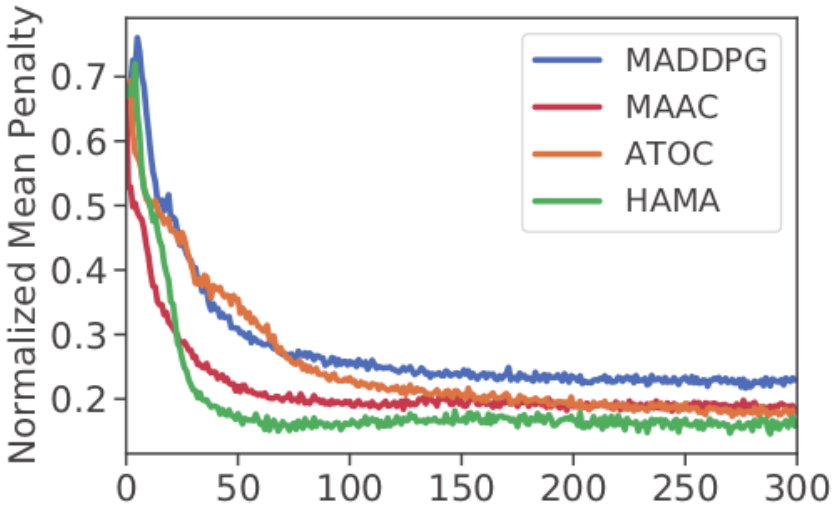


Figure 14: Penalties during training on cooperative navigation with 3 agents.[23]

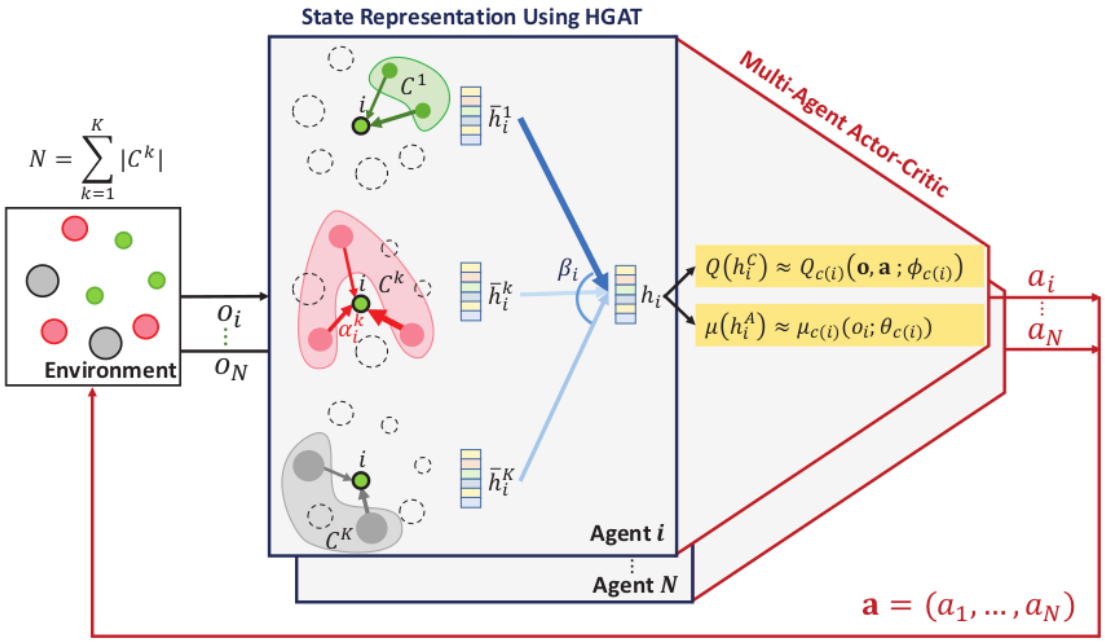


Figure 1: Overview of HAMA.

Figure 15: Overview of HAMA[23]