A Hierarchical Approach for MARLÖ Challenge

Linjie Xu
School of Software
Nanchang University
Nanchang, China
ncuxlj@email.ncu.edu.cn

Yihong Chen
Department of Electronic Engineering
Tsinghua University
Beijing, China
cyh16@mails.tsinghua.edu.cn

***Abstract*—Recently reinforcement learning has been showing remarkable performance in playing games. However, the majority of conventional approaches merely solve games with a single task. It is not yet well studied whether reinforcement learning is effective in games like Minecraft, where players are required to finish multiple different tasks while cooperating with other collaborators. In such games, AIs are confronted with dual challenges – finishing multiple tasks and building a multi-agent system. We propose a hierarchical approach with reinforcement learning policies to address the challenges. Experiments show that our approach performs well when dealing with multiple tasks and multiple agents simultaneously. Our approach got the second runner-up in MARLÖ Challenge, demonstrating its potential in tackling the challenges.**

***Index Terms*—Game AI, Multi-Task, Multi-Agent, Reinforcement Learning, Minecraft**

I. Introduction

Game AIs are playing an important role in a diverse set of games. They can serve as competent opponents for human players in fighting games [1] or plausible non-player characters (NPCs) that react robustly to novel game contents [2]. They are also capable of creating complex and interesting levels in procedural generation [3], [4]. Furthermore, game AIs can be used to play various existing games [5]. Although multi-agent games and multi-task games are common, few game environments for AI combine both characteristics, which distinguishes the MARLÖ environment [6] from others. Built on a popular sand-box game, Minecraft, the MARLÖ environment requires the player to control two agents, which cooperate to complete three tasks: *Build Battle*, *Treasure Hunt* and *Mob Chase*. Each of the three tasks has 5 different scenes for training and one withheld scene for test. At each time step, the agents receive a scalar reward after taking some actions.

As shown in Figure 1, the three tasks in MARLÖ environment are quite different. In *Build Battle*, the agents are required to observe the target cuboid structure and then build the same one nearby. In *Treasure Hunt*, the scenes are in a dangerous cave with lava and monsters. An agent, serving as the collector, fetches treasure and reach the destination while the other agent, as the defender, tries to protect the collector from being attacked by the monsters. In *Mob Chase*, the two agents cooperate to chase and capture various animals.

We have many thanks to EPFL Enable Office for offering us a student travel grant on MARLÖ 2018 challenge.

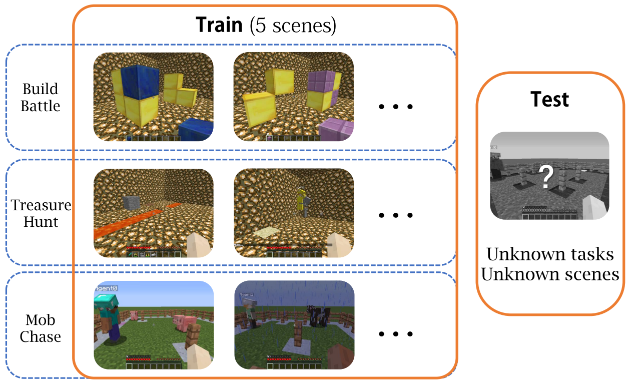

Fig. 1: Overview of the MARLÖ environment.

Such diverse tasks and complex interactions between teammates bring immersive experience to human players, together with a great challenge for game AIs – simultaneously controlling multiple agents to complete multiple tasks. Specifically, we list two aspects of the challenge as follows.

Multi-Task. In MARLÖ, the game AI has to complete three diverse tasks. Although modern AI algorithms [7]–[9] has been showing promising results on single-task learning, multi-task learning, especially multi-task reinforcement learning, has not been fully studied. Differed from traditional reinforcement learning, multi-task reinforcement learning aims to generate policies that can solve multiple tasks at the same time. Given the fact that most real-world applications require AIs to accomplish several related tasks at the same time(e.g. tagging the barrier and meanwhile predicting the distance for an autonomous vehicle), it is of great importance to find effective solutions to such multi-task reinforcement learning problem.

Multi-Agent. In a multi-agent system, an agent interacts with not only the environment but also other agents, leading to a more dynamic system and exponential computation complexity with respect to the number of agents. Although teamwork is crucial in multi-agent systems [10], [11], it is usually hard for the agents to learn how to collaborate without any prior knowledge [12]. In MARLÖ, the game AI has to control two agents at the same time. Hence, addressing multi-agent reinforcement learning appropriately is critical to tackle MARLÖ challenge.

II. Related Work

Although Minecraft has been addressed by integrating graph neural networks for agents' communication [13] and injecting important area features to deep reinforcement learning [14], both works merely offer solutions to single scene learning(i.e. training and test using the same scene), far from tackling the MARLÖ challenge completely. Context-dependent memory Q network [15] is able to generalize to unseen maps in Minecraft, yet hard to generalize among different tasks.

To solve multiple different tasks efficiently, the agents are expected to learn general skills that can be used in many tasks. Previous works proposed to combine multiple expert policies in order to create a student policy that can handle a set of tasks [16], [17]. However, in our MARLÖ experiments, we observe that the student policy distilled from expert policies rarely generalize well in unseen environments, resulting in a task-specific policy rather than an integral policy. Some works employed hierarchical policies [18], [19] to learn different skills among tasks. In our approach, we use a simpler hierarchical architecture and focus on task-recognition and the performance on each task.

As for multi-agent learning, sharing information between agents has been proposed as an efficient way to learn to cooperate [10], [11], [20], [21]. In MARLÖ environment, cooperation mechanisms are totally different among tasks (For instance, cooperation in *Treasure Hunt* is that the defender help the collector attack monsters, while in *Mob Chase*, the agents chase the animals together), making it hard to directly employ the above methods. We thus resort to task-specific policies, through which the agents can learn different cooperation mechanisms with different training procedures.

III. Preliminary

Reinforcement Learning (RL) [22] is a popular method for playing video games. Many RL methods [7]–[9] have shown remarkable performance on various games, including classic Atari Games [7], Vizdoom [23] and lately StarCraft [24]. Before presenting our hierarchical approach to solve MARLÖ, we briefly review reinforcement learning in this section. RL aims to solve Markov Decision Process (MDPs) problems. A finite horizon discounted MDP $\mathcal{M}$, as often assumed in RL problems, can be defined as a tuple: $<\mathcal{S},\mathcal{A},T,R,\gamma>$ where $\mathcal{S}$ is a finite set of all possible states in an environment, $\mathcal{A}$ is a finite set of actions which are available from each state s. $T:\mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$ is a transition function that describes how states change under the action made by agent. And $R:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$ is a reward function that assigns a scalar r_t to the agent after every interaction. $\gamma\in[0,1]$ is a discounted coefficient that controls the weight of future rewards in the discounted accumulative rewards:

$$G_t=\sum_{i=1}^{t}\gamma^{i-1}r_i \tag{1}$$

The purpose of RL is to find an optimal policy $\pi:\mathcal{S}\rightarrow\mathcal{A}$ that maximizes the expected return $\mathbb{E}[G_t]$.

In our approach, we regard the screen capture at each step as state s_t. And we use different action configurations for different tasks (see the details in appendix A).

IV. A Hierarchical Approach with Reinforcement Learning

In [25], they designed a classifier to select an effective MPC controller from a candidate set. Inspired by this, we propose a hierarchical approach with reinforcement learning policies to tackle MARLÖ environment. The hierarchical approach shown in Figure 2 is composed of a selection module and task-aware policy modules. For MARLÖ, the policy module consists of three task-specific policies. The selection module recognizes the task type and chooses a corresponding policy to finish the task. Formally, given a task set $\mathcal{K}$ and its size $|\mathcal{K}|=N$, our target is to maximize

$$\mathbb{E}_{k\sim\mathcal{K}}\mathbb{E}[G_t^k]=\frac{1}{N}\sum_{k\in\mathcal{K}}\mathbb{E}[G_t^k] \tag{2}$$

where $\mathbb{E}[G_t^k]$ denotes the expected return of task k.

We first find the task-aware policy π^k that maximizes the corresponding expected return using Proximal Policy Optimization (PPO) [9]. For task k, PPO maximizes the following objective:

$$L(\theta^k)=\hat{\mathbb{E}}[min(u_t(\theta^k),1-\epsilon,1+\epsilon)\hat{A}_t^k] \tag{3}$$

where $u_t(\theta^k)=\frac{\pi_{\theta^k}(a_t^k|s_t^k)}{\pi_{\theta^k_{old}}(a_t^k|s_t^k)}$ and $\hat{}$ indicates that the value is estimated from a finite collection of trajectories. θ^k, $\pi_{\theta^k_{old}}$ and $A^k(s^k,a^k)=Q^k(s^k,a^k)-V^k(s^k)$ respectively denote the parameters of the policy, the policy before current update and the state-dependent action advantage function. ϵ is a clipping hyper-parameter. This objective function is based on *conservative policy iteration* (CPI) [26], which finds an approximately optimal policy within a finite number of policy updates.

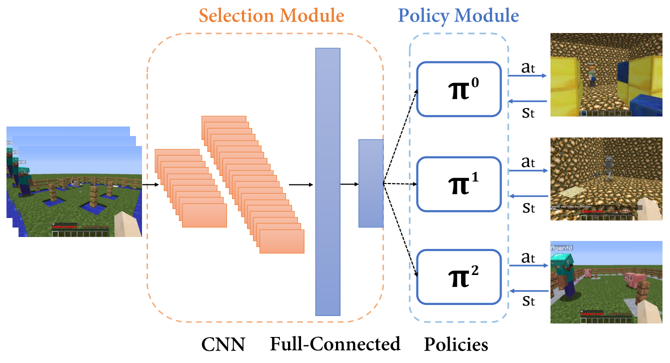

Fig. 2: A brief view of our hierarchical approach. The selection module C is constructed by CNN and full-connected layers. The policy modules consist of task-aware policies, through which the agents interact with environments.

In the test procedure of MARLÖ challenge, the agents are confronted with unknown tasks. Making decisions under such conditions is tough. Therefore, we construct a selection module $C : \mathcal{S} \rightarrow \mathcal{T}$ that recognizes the task effectively from the first few screen frames and then choose the corresponding policy. In our implementation, the selection module C is a CNN classifier.

To simplify the multi-agent problem, our approach considers a customized cooperation mechanism for each task. This is achieved by separate training of the agents and tailored reward shaping. In *Treasure Hunt*, the defender is trained first to gain the ability to protect itself and the collector. Then the collector is trained to learn how to fetch the treasure and reach the exit. In *Mob Chase*, one of the two agents is trained by introducing an external reward that encourages it to stay on the left-hand side, while the other is trained in a similar way to encourage it to stay on the right-hand side. Such cooperation on both sides makes chasing animals much easier. As for *Build Battle*, only one agent is used to tackle the task, in consideration of trade-off between single-agent performance and the complexity of two-agent cooperation.

V. Experiments

To demonstrate the effectiveness of our hierarchical approach, we run experiments similar to MARLÖ challenge.

A. Experimental settings

500 frames from each task (100 frames per scene) are collected to train the selection module C with another 440 ones for test. To train each task-specific policy, the scenes are randomly chosen from a finite scene set. As the setting in MARLÖ challenge, we construct 2 extra scenes per task, unseen in the training set, for test. Specifically, for *Build Battle*, we build different target cuboid structures, using different types of cubes. And for *Treasure Hunt*, the monster type is different from training. For *Mob Chase*, we change the animal types and the weather. In *Treasure Hunt* and *Mob Chase*, we use different maps for test as well. During the end-to-end test, one of the 6 test scenes are uniformly chosen. The test stops when we accumulate 100 episodes for each task.

B. Baselines

To verify the effectiveness of our hierarchical approach, we compare its performance with two baselines. A naive baseline is to train task-specific policies and then uniformly choose one to handle MARLÖ environment. Simple as it is, this baseline can reflect the importance of the selection module in our hierarchical approach. As for the second baseline, we use policy distillation [16], training a student policy using multi-task data produced by teacher policies (well-trained single-task policies).

C. Results

As shown in Table I, our hierarchical approach outperforms the random baseline and the policy distillation baseline significantly. In the random baseline, the selection module is replaced with random selection thus task-specific policies become inefficient confronted with unknown tasks. It is worth noting that as policy distillation baseline outperforms the naive baseline in *Build Battle* and *Treasure Hunt*. However, it is of poor performance in *Mob Chase*.

TABLE I
Performance (cumulative rewards) of different approaches on test. Scores are averaged over 100 episodes.

Task	Build Battle	Treasure Hunt	Mob Chase
Random Policy	-0.45 ±0.83	-0.33 ±0.92	-0.37 ±0.90
Policy Distillation	-0.054 ±0.94	-0.21 ±0.78	-0.54 ±0.66
Task-specific Policies	0.76 ±0.14	0.4 ±0.85	0.61 ±0.55
Our Approach	0.76 ±0.16	0.37 ±0.86	0.55 ±0.63

TABLE II
Performance of selection module on different tasks

Tasks	Build Battle	Treasure Hunt	Mob Chase
Accuracy	96.6%	95.0%	100%

We further analyze our hierarchical approach here. Table II shows that the selection module in our approach can recognize the task successfully, with accuracies of over 95%. Figure 3 shows the performance of the policy module. Benefiting from task-aware policies by using customized configurations (see appendix A) for different tasks, our policy module is able to perform well across the three tasks. From Figure 3 we observe the instability in the training procedure of *Build Battle*. This is due to diverse scenes in this task. Moreover, the gap between train and test in *Treasure Hunt* (Figure 3 middle) is caused by monsters killing agents in test scenes.

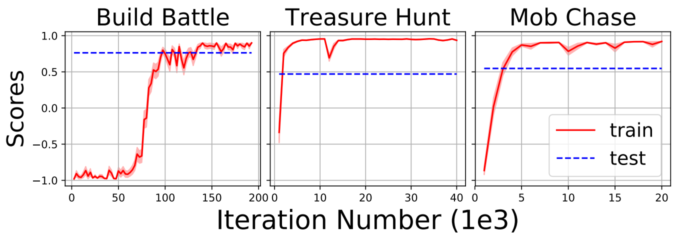

Fig. 3: Performance (cumulative rewards) of policy module on different tasks. The scores are smoothed in every 50 episodes. The task-specific models used in test are selected from the training procedure.

VI. Conclusion and Future Work

In this work, we explore a hierarchical approach with reinforcement learning policies for the MARLÖ challenge, where the game AI is required to simultaneously control multiple agents to accomplish multiple tasks. We decomposes this problem into two parts, task recognition and task finishing. We design modules to handle them, selection module and policy module respectively. Our approach is able to use flexible configurations and learn task-customized cooperation. Compared with the baseline methods, experiments show that our approach performs better, generalizing well to new scenes in MARLÖ environment. This hierarchical approach helped us get a second runner-up in the final tournament of MARLÖ challenge.

There still remain challenges in solving multi-agent and multi-task problems at the same time, for example, the trade-off between computation efficiency and multi-task performance. Although ideally we could use policies dedicately designed for each agent in each task, the computation cost would be unaffordable, and thus the approach would fail to scale up to a large set of tasks and a large number of agents. Another direction worth to explore is how the agents can learn different types of cooperation for different tasks, which is done by hand-crafted reward shaping in our approach. We leave these for the future work.

REFERENCES

[1] Makoto Ishihara, Suguru Ito, Ryota Ishii, et al. Monte-carlo tree search for implementation of dynamic difficulty adjustment fighting game ais having believable behaviors. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[2] Christian Guckelsberger, Christoph Salge, and Julian Togelius. New and surprising ways to be mean. adversarial npcs with coupled empowerment minimisation. *arXiv preprint arXiv:1806.01387*, 2018.

[3] Adam Summerville, Sam Snodgrass, et al. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

[4] Matthew Stephenson and Jochen Renz. Procedural generation of levels for angry birds style physics games. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[5] Niels Justesen, Philip Bontrager, Julian Togelius, et al. Deep learning for video game playing. *IEEE Transactions on Games*, 2019.

[6] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, et al. The multi-agent reinforcement learning in malmö (marlö) competition. *arXiv preprint arXiv:1901.08129*, 2019.

[7] Volodymyr Mnih, Koray Kavukcuoglu, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, et al. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] Jun Feng, Heng Li, Minlie Huang, et al. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1939–1948, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[11] Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, et al. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 122–130, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[12] Katie Genter, Noa Agmon, and Peter Stone. Role-based ad hoc teamwork. In *Proceedings of the Plan, Activity, and Intent Recognition Workshop at the Twenty-Fifth Conference on Artificial Intelligence (PAIR-11)*, August 2011.

[13] Valliappa Chockalingam, Tegg Tae Kyong Sung, Feryal Behbahani, et al. Extending world models for multi-agent reinforcement learning in malmö. In *Joint Proceedings of the AIIDE 2018 Workshops co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2018), Edmonton, Canada, November 13-14, 2018.*, 2018.

[14] Nicolas Bougie and Ryutaro Ichise. Deep reinforcement learning boosted by external knowledge. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, pages 331–338, New York, NY, USA, 2018. ACM.

[15] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, et al. Control of memory, active perception, and action in minecraft. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2790–2799. JMLR.org, 2016.

[16] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, et al. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[17] Glen Berseth, Cheng Xie, Paul Cernek, et al. Progressive reinforcement learning with distillation for multi-skilled motion control. In *International Conference on Learning Representations*, 2018.

[18] Chen Tessler, Shahar Givony, Tom Zahavy, others J, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[19] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[20] Ying Wen, Yaodong Yang, Rui Luo, et al. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019.

[21] Ryan Lowe, Jakob Foerster, Y.-Lan Boureau, et al. On the pitfalls of measuring emergent communication. *CoRR*, abs/1903.05168, 2019.

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Michał Kempka, Marek Wydmuch, et al. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[24] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[25] Francois Robert Hogan, Eudald Romo Grau, et al. Reactive planar manipulation with convex hybrid mpc. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 247–253. IEEE, 2018.

[26] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

APPENDIX A
DETAILS OF NETWORKS AND ENVIRONMENT SETTINGS

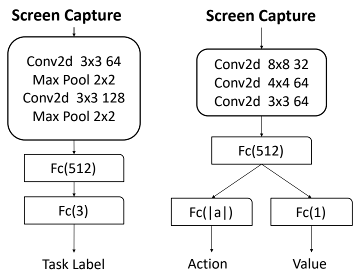

Fig. 4: Architecture of selection module and policy module.

Actions. The action set for *Build Battle* task contains *go forward, go backward, turn left, turn right, place block*. The agent's view is fixed on -45' in pitch. In *Treasure Hunt* task, the collector collects treasure and then go to exit point, while the defender is able to attack but unable to collect treasure. Actions for the defender are *go forward, turn left, turn right, attack* and actions for the collector are *go forward, turn left, turn right*. For *Mob Chase* task, actions are *go forward, turn left, turn right*.

Reward Shaping. Based on default reward setting on MARLÖ environments, we modify all the positive reward to +1 and negative reward to -1 while maintaining the penalty of every step (-0.02). In *Mob Chase*, we give an external positive reward to the agent if it is at the correct side (one agent chase the mob at the left-hand side and another agent is at right-hand side).