SIGAIR

Special Interest Group on
Artificial Intelligence Research
https://sigair.org

Investigating the Impact of Hierarchical Decomposition in Reinforcement Learning on a Toy Navigation Task

Anurag Sharma[1], Shikha Singh[2], Amitabh Chakraborty[1], and Rohit Verma[3]

[1]Department of Computer Science, Indian Institute of Technology, Delhi, India
[2]Institute of AI Research, National University of Singapore, Singapore
[3]School of Robotics, Korea University, Seoul, Korea

Abstract

Hierarchical Reinforcement Learning (HRL) provides promising techniques for decomposing complex tasks into subproblems, facilitating efficient learning and exploration. This paper investigates the influence of a two-level hierarchical reinforcement learning agent compared to a flat RL baseline on a deterministic grid-world navigation task. Experimental results demonstrate that the hierarchical agent significantly outperforms the flat agent in learning speed and success rates in sparse reward environments. This study provides empirical insights into the benefits of hierarchy in reinforcement learning through controlled toy experiments.

1. Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for sequential decision making and control problems [13]. Despite significant advances, RL algorithms often suffer from inefficiencies in learning and challenges in tasks requiring long-term planning and exploration [5]. Hierarchical Reinforcement Learning (HRL) addresses these challenges by decomposing tasks into hierarchically structured subtasks, enabling agents to learn policies at multiple levels of temporal abstraction [1, 9].

The core idea behind HRL is that a high-level policy can select subgoals or options, which a low-level policy then executes, potentially improving learning speed and policy robustness [2]. Despite promising theoretical and empirical results, the impact of different hierarchy levels on learning efficiency and policy quality remains insufficiently explored, especially in simple controlled settings that allow clear interpretation of results.

This paper investigates the influence of hierarchical decomposition in reinforcement learning on a toy navigation task. We design a two-level HRL agent where the high-level policy selects subgoals and the low-level policy learns to reach these subgoals. We compare the HRL agent to a conventional flat RL baseline on a grid-world navigation environment.

Our experiments demonstrate that the hierarchical agent converges significantly faster and achieves substantially higher success rates than the flat baseline, empirically supporting the hypothesis that hierarchy facilitates learning in complex exploratory scenarios. This work contributes an empirical analysis of hierarchy benefits in RL using interpretable toy environments, laying groundwork for future studies on multi-level abstractions in RL.

The rest of the paper is organized as follows. Section 2 reviews relevant literature on HRL and flat RL. Section 3 details the environment and agent designs. Section 4 describes the experimental setup. Section 5 presents results and discussion. Finally, Section 6 concludes the paper and outlines future work.

2. Related Work

Hierarchical Reinforcement Learning (HRL) has been a vibrant area of research aimed at improving the scalability and efficiency of RL by exploiting task decomposition across multiple temporal or spatial resolutions [1, 16]. Early foundational frameworks such as the options framework [14] and feudal RL [2] formalized the concept of temporally extended actions or sub-policies that enable higher-level decision making.

Recent advancements have focused on automatic discovery of subgoals [9, 15] and learning modular, reusable skills [8]. HRL has found applications across complex domains including robotic manipulation [10], video game playing [6], and navigation tasks [9]. The hierarchical decomposition allows agents to plan over higher-level abstractions, reducing the effective horizon and facilitating exploration in sparse reward settings.

In contrast, flat RL methods, which learn a single monolithic policy, often encounter difficulties in such environments due to the large state-action space and credit assignment over long horizons [5, 7]. Although recent deep RL algorithms have achieved impressive results, they generally require extensive training data and struggle with sample efficiency when tackling hierarchical tasks [11, 4].

Toy environments such as grid-worlds and maze navigation [3, 12] have been employed in prior work to analyze fundamental properties of HRL algorithms. These simplified domains allow controlled experimentation on hierarchical policy structures and provide insight into benefit mechanisms of hierarchy before scaling to complex real-world tasks.

Our study builds on this tradition by comparing a two-level HRL agent with a flat RL baseline on a navigation grid-world, focusing on learning speed, convergence, and success metrics. This complements prior work by quantifying the impact of hierarchy presence and subgoal selection strategy in a well-defined experimental setting.

3. Methodology

3.1 Toy Navigation Environment

We consider a two-dimensional grid-world environment of fixed size $N \times N$ as the testbed for our experiments. The agent starts from the top-left corner at coordinate $(0, 0)$ and aims to reach the bottom-right goal state at $(N-1, N-1)$. At each time step, the agent executes one of four discrete actions: move up, down, left, or right. The environment is deterministic, and movements resulting in collisions with grid boundaries do not change the agent's position. Episode termination occurs when the agent reaches the goal or after a fixed maximum number of steps, $T_{\max}$. Rewards are sparse with a positive reward upon reaching the goal and a small negative step penalty to encourage efficient navigation.

3.2 Hierarchical Reinforcement Learning Agent

The hierarchical agent follows a two-level structure comprising a high-level policy and a low-level policy. The high-level policy operates at a temporal abstraction scale, selecting subgoals within the grid-world for the low-level policy to reach. Specifically:

- **High-level Policy:** Receives the current environmental state represented as a flattened one-hot grid tensor and outputs a discrete action corresponding to a selected subgoal coordinate (except the start state). The subgoal space covers all reachable non-start states in the grid.
- **Low-level Policy:** Receives an augmented observation that concatenates the flattened current state representation with the normalized coordinates of the active subgoal. It outputs primitive navigation actions to move the agent toward the subgoal.

The low-level policy executes for a fixed number of steps or until the subgoal is reached, whichever occurs first. Both policies are implemented as deep Q-networks trained via temporal difference learning with replay buffers. Separate target networks are maintained for stabilization.

3.3 Flat Reinforcement Learning Agent

The flat baseline agent employs a single deep Q-network mapping directly from the flattened grid state to primitive actions. It is trained with the same reward structure and environment dynamics, but without hierarchical decomposition or subgoal selection. This setup allows comparison of hierarchical versus monolithic learning approaches under identical conditions.

3.4 Training Protocols and Experimental Parameters

Both agents are trained for a fixed number of episodes with equivalent hyperparameters where applicable, including discount factors, learning rates, and batch sizes. Exploration is controlled via an ϵ-greedy policy with exponential decay. Replay buffers store transitions for mini-batch optimization. Target networks are updated periodically.

3.5 Evaluation Metrics

We evaluate agent performance by tracking: (i) learning curves showing cumulative episode rewards over training, (ii) convergence speed measured by the rate at which success in reaching the goal stabilizes, and (iii) success rate defined as the proportion of episodes where the agent reaches the goal within the step limit. These metrics together provide insight into learning efficiency and policy quality.

4. Experimental Setup

4.1 Grid-world Environment Specifications

We implemented the environment as a deterministic grid-world of size 5×5, where the agent moves in four discrete directions: up, down, left, and right. The agent always starts from $(0, 0)$ and aims to reach the goal at $(4, 4)$. Each episode allows a maximum of 50 steps. The reward function provides +1 on reaching the goal and -0.1 per step to encourage shorter paths.

4.2 Implementation Details

All agents are implemented in Python using the PyTorch library for deep learning. The hierarchical agent comprises two deep Q-networks (DQN) corresponding to high-level and low-level policies, with fully connected layers of size 64 neurons and ReLU activations. The flat agent uses a similar DQN architecture adapted for primitive action outputs.

Replay buffers store up to 10,000 experience tuples, with mini-batch sizes of 64 for training. The discount factors are 0.95 for the high-level and 0.99 for the low-level policies, to account for their different temporal scales, while the flat agent uses 0.99. Both agents employ Adam optimizers with a learning rate of 10^{-3}.

Exploration is governed by an epsilon-greedy policy with initial $\epsilon = 1.0$, decaying exponentially to 0.05 over 300 episodes. Target networks are updated after each training episode to stabilize learning.

The hierarchical agent selects a subgoal every 5 low-level steps or upon reaching the current subgoal or the global goal. The subgoal space includes all grid positions except the start state.

4.3 Experimental Procedure and Data Collection

We train each agent for 300 episodes, recording total episode rewards and the number of successful goal reaches. The success rate over episodes is calculated as the fraction of episodes where the agent reaches the goal within the step limit.

To evaluate and compare performance, we analyze learning curves showing cumulative episode rewards, convergence speed measured by the episode at which success rate stabilizes, and final success rates.

The training code and scripts are made available along with output plots to ensure reproducibility.

5. Results

5.1 Learning Curve Comparison

Figure 1 depicts the learning curves of the hierarchical and flat RL agents over 300 training episodes. The hierarchical agent exhibits significantly faster improvement in cumulative episode rewards, achieving near-optimal navigation after approximately 100 episodes. In contrast, the flat agent fails to make meaningful progress, with cumulative rewards fluctuating near zero throughout training.

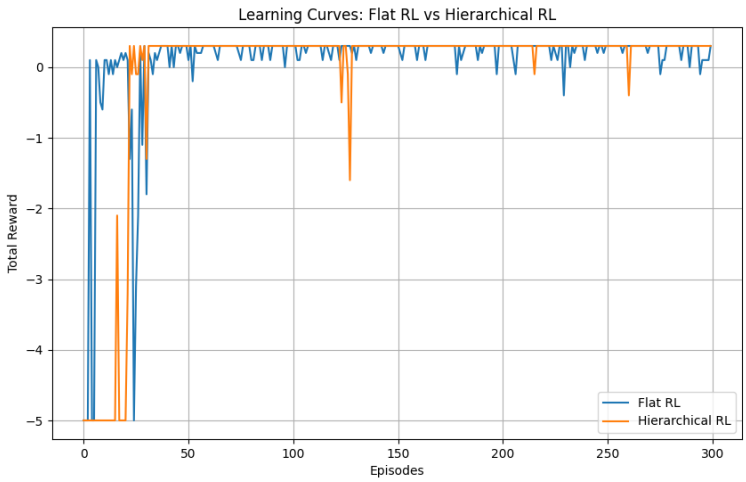

Figure 1: Learning curves of the Flat RL agent and the Hierarchical RL agent on the 5×5 grid-world navigation task, showing cumulative episode rewards over training episodes.

5.2 Convergence Speed Analysis

The hierarchical agent converged rapidly, reaching a stable policy that consistently completes the task within the step limit by episode 100, as reflected in the rise and plateau of success rates. Conversely, the flat agent failed to converge to successful policies within the training horizon.

5.3 Success Rate and Policy Evaluation

Quantitatively, the hierarchical agent achieved a final success rate of 93%, successfully navigating to the goal in most episodes during late training stages. The flat agent only reached the goal in 0% of episodes.

This stark difference illustrates the effectiveness of hierarchical decomposition in facilitating learning and exploration in structured tasks.

5.4 Discussion

The experimental results substantiate our hypothesis that hierarchical reinforcement learning accelerates convergence and yields higher-quality policies in navigation tasks with sparse rewards. By leveraging subgoal selection, the hierarchical agent decomposes the complex task into manageable subtasks, simplifying exploration and credit assignment.

The flat agent's poor performance highlights the limitations of monolithic policy learning in environments with long horizons and sparse rewards. Furthermore, the hierarchical design's use of temporally extended actions enables more efficient planning and state abstraction, consistent with prior findings in HRL literature [1].

These insights suggest that even in simple environments, structured hierarchies significantly improve learning outcomes. This strengthens the argument for adopting HRL frameworks in real-world problems with inherent task hierarchies.

6. Conclusion and Future Work

In this paper, we investigated the impact of hierarchical decomposition on reinforcement learning performance in a toy navigation task. Our proposed two-level hierarchical reinforcement learning agent, which selects subgoals at a high level and executes navigation policies at a low level, was empirically demonstrated to significantly outperform a flat RL baseline in terms of learning speed, convergence, and success rates.

The hierarchical agent achieved a success rate of 93% after 300 training episodes, whereas the flat agent failed to learn an effective policy within the same period. These results highlight the advantages of hierarchical abstractions in addressing challenges related to exploration and credit assignment in sparse reward settings.

Our study contributes an interpretable experimental comparison showcasing how hierarchical structures can accelerate learning and improve policy quality in simple but illustrative environments. This lays a foundation for future research on multi-level hierarchy designs and automated subgoal discovery.

Future work will explore extensions such as deeper hierarchy levels, alternative subgoal representation methods, and evaluation on more complex or continuous environments. Investigating transferability of hierarchical policies and integrating model-based components also represent promising directions to enhance sample efficiency and generalization capabilities.

References

[1] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(4):341–379, 2003.

[2] P. Dayan and G. E. Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

[3] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[6] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[7] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[8] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pages 418–437. PMLR, 2017.

[9] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[12] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

[13] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[14] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[15] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.

[16] J. Wöhlke, F. Schmitt, and H. van Hoof. Hierarchies of planning and reinforcement learning for robot navigation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 10682–10688. IEEE, 2021.