Learning Multi-Level Skill Hierarchies with Graphwave

Simon Bele
sb95099@student.uni-lj.si
University of Ljubljana
Faculty of Computer and
Information science,
Ljubljana, Slovenia

Jure Žabkar
jure.zabkar@fri.uni-lj.si
University of Ljubljana
Faculty of Computer and
Information science,
Ljubljana, Slovenia

ABSTRACT

We introduce a novel framework for learning multi-level skill hierarchies in reinforcement learning environments by leveraging structural similarities in state-space graphs. To obtain structural embeddings, we use the Graphwave algorithm, which places structurally similar states in close proximity in the latent space. In the latent space, we perform hierarchical clustering of states while respecting the topology of the state-space graph. At different levels of the hierarchy we learn the options that represent the skills; a skill at each level of the hierarchy is defined using the skills from the level below. We compare our approach with the state-of-the-art method across several environments. Our results show that structural embeddings can speed up option learning significantly in certain domains.

KEYWORDS

Skill Hierarchy, Reinforcement Learning, Options, Structural similarity, Graph Embeddings

1 INTRODUCTION

In Reinforcement Learning (RL), an agent learns to make decisions by interacting with an environment; it operates on the principles of trial and error and obtains positive or negative feedback (rewards) from the environment. The overall goal of the agent is to maximize the cumulative rewards. Traditional RL approaches can struggle with scalability and efficiency as the complexity of the environment increases or the task become increasingly difficult.

A possible way to tackle this challenge is to introduce skill hierarchies in RL [1, 6]. Skill hierarchies enable the decomposition of complex tasks into simpler sub-tasks, usually improving the generalization of learned behaviors across different scenarios. This usually leads to a more efficient learning process but also produces more robust and interpretable actions.

Traditional approaches in developing these hierarchies have primarily focused on single-level structures, where skills are often defined through predefined policies or through the clustering of state transitions without considering the deeper structural relationships between these transitions. Recently, Evans et al. [3] introduced a method for learning skill hierarchies based on Louvain clustering of the state-space graph, which optimizes its modularity.

In this paper, we introduce an approach that goes beyond modularity: we use the Graphwave algorithm that identifies structural similarities within a graph. We cluster structural embeddings in latent space, thus providing a more robust foundation for skill learning. Our approach also preserves the topology of the state graph and so enables us to learn the options framework on the obtained clustering.

We evaluate our method by comparing it to the approach of Evans et al. [3]. We integrate our code into their framework and observe the learning efficiency on four domains. We show that in three out of four, our method performs significantly better while in the Four Rooms domain that features extreme modularity, the approach of Evans et al. outperforms ours.

2 RELATED WORK

A common approach in reinforcement learning involves modeling the underlying Markov Decision Process (MDP), wherein a policy $\pi : S \times A \rightarrow [0, 1]$ is learned to maximize a reward function. Specifically, the action-value function $Q^{\pi}(s, a)$ for a policy π encapsulates the expected reward for states in the environment. The action-value function adheres to the Bellman equations, and the task can thus be rephrased as optimizing these equations to find the optimal policy.

The options framework in reinforcement learning is a well-established method for reasoning across multiple levels of temporal abstraction, effectively implementing Semi-Markov Decision Processes [4, 8].

An option is defined by a 3-tuple $\omega = (I_{\omega}, \pi_{\omega}, \beta_{\omega})$, where $I_{\omega} \subseteq S$ represents the subset of the state space in which the option is executable, $\pi_{\omega} : S \times A \rightarrow [0, 1]$ is a policy determining the probability of taking action a in state s, and $\beta_{\omega} : S \rightarrow [0, 1]$ specifies the termination condition, indicating the probability of option termination in a given state.

To hierarchically cluster the state space, one can derive higher-level options over lower-level options, where the initiation set of the higher-level option is the union of initiation sets of lower-level options, thereby enabling options at multiple time scales.

Training options across multiple time scales necessitates generalizing the usual Bellman equations to be defined over options rather than actions, termed intra-option learning [7].

The MDP induces a state transition graph, with nodes representing states and edges denoting possible actions between states.

Several approaches leverage the state transition graph of the underlying MDP to learn skills. A notable advancement by Xu et al. [9] employs the Louvain graph clustering algorithm to partition the state transition graph into clusters, subsequently defining options as traversals across the aggregate graph of these clusters.

Previous efforts to create single-level skill hierarchies have primarily utilized different measures of centrality or various graph partitioning algorithms.

Evans et al. [3] introduce a multi-level skill hierarchy trained on the entire hierarchical clustering of Louvain. Due to the intractable problem of modularity maximization, the greedy-natured Louvain

algorithm optimizes for moving nodes between partitions at each step if and only if this move results in a positive modularity gain. This can be seen as a local approach to modularity optimization. They employ macro-Q learning [5] and intra-option learning [7] to train hierarchical agents.

The above approach is novel in producing the first multi-level skill hierarchy, where it is produced automatically with no human intervention. Through it they obtain options reflecting optimizing for modularity, which they show to be useful for navigating at the top-most level of the skill hierarchy.

3 METHODOLOGY

3.1 Structural similarity embeddings

To obtain an embedding of nodes that places structurally similar nodes in close proximity within the latent space, we employ Graphwave [2], a methodology that provides strict guarantees regarding the separation of structurally equivalent nodes.

Consider an undirected graph $G = (V, E)$ with its graph Laplacian defined as $L = D - A$, where D is the degree matrix and A is the adjacency matrix. Let the eigenvector decomposition of L be given by

$$L = U\Lambda U^T, \tag{1}$$

where U is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

By applying a heat diffusion wavelet $g_s(\lambda) = e^{-\lambda s}$, define the spectral graph wavelet centered at node a as

$$\Psi_a = U \,\mathrm{Diag}(g_s(\lambda_1), \ldots, g_s(\lambda_N))U^T\delta_a, \tag{2}$$

where δ_a is the Dirac delta function at node a.

To circumvent computational intractability [2], the wavelet is treated as a probability distribution over the graph:

$$\phi_a(t) = \frac{1}{N}\sum_{m=1}^{N} e^{it\Psi_{ma}}, \tag{3}$$

for time point t. The empirical characteristic function is then sampled and transformed into a vector embedding:

$$\chi_a = [\mathrm{Re}(\phi_a(t_i)), \mathrm{Im}(\phi_a(t_i))]_{t_1,\ldots,t_d}, \tag{4}$$

with d being the number of samples.

This resulting $2d$-dimensional embedding ensures that structurally equivalent nodes in the graph will be at most a predefined ϵ distance apart in the ℓ_2 norm, thereby providing rigorous guarantees on the proximity of such nodes [2].

3.2 Clustering

To hierarchically cluster nodes based on their embeddings, our approach utilizes an agglomerative clustering algorithm.

This algorithm iteratively merges the nearest clusters based on the average linkage criterion. To maintain the integrity of the graph's topology, clusters are only compared if there exists a direct path between them that bypasses other clusters. The height of the hierarchy was chosen to match the height of Louvain for the sake of fair comparison between the two approaches [3], but could also be defined through any dendrogram cutting strategy.

3.3 Option learning

For the sake of comparisons with Evans et al. [3], we similarly construct the skill hierarchy as follows.

Let h represent the number of partitions produced by our algorithm when applied to the state transition graph. Each of these h partitions defines a skill layer, forming an action hierarchy with h levels of abstract actions above primitive actions. Each hierarchy level consists of skills designed to efficiently navigate between clusters of the state transition graph.

We define options for moving from cluster c_i to cluster c_j is defined by: initiation states in c_i, a policy to navigate from any state in c_i to a state in c_j, and termination upon reaching c_j.

Leveraging the hierarchical structure of the partitions, we define skills at each level of the hierarchy using the skills from the preceding level. At each hierarchy level, the policies for higher-level actions call actions (either options or primitive actions) from the level below, with primitive actions only invoked directly at the base level.

4 EVALUATION

4.1 Domains

The skill hierarchy is evaluated in four environments (Figure 1), the first three of which are different examples of the rooms environment. An empty room, two rooms connected by a bottleneck and four rooms as in [3]. The agent is given a starting position and a goal position and attempts to navigate between them as effectively as possible. The last domain we look at is the Towers of Hanoi, a classic mathematical puzzle. It involves moving a set of disks from one peg to another, following specific rules.

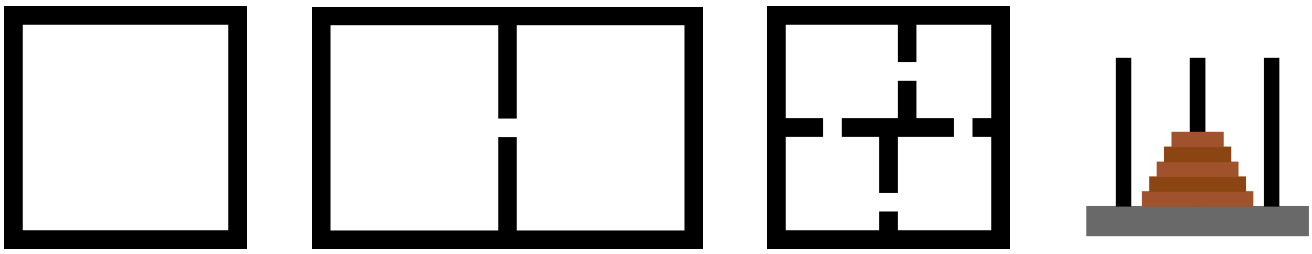
Each of the rooms environments feature four basic movements: north, south, east, and west. These movements steer the agent in the chosen direction unless obstructed by a wall, in which case the agent stays in place. Each action incurs a penalty of -0.001, with a bonus of +1.0 awarded upon reaching the goal state. Each run begins from a designated start state and aims for a goal state.

The Towers of Hanoi involves four disks of varying sizes positioned on three pegs. An episode commences with all disks stacked on the leftmost peg. Actions involve moving the top disk from one peg to another, ensuring no larger disk is placed on a smaller one. Each action incurs a -0.001 penalty, with an additional +1.0 reward granted upon achieving the goal state, which is when all disks are stacked on the rightmost peg.

4.2 Structural Skill Hierarchy

The hierarchy obtained through the above clustering method will cluster structurally similar nodes together.

To showcase an example, we show the hierarchical clustering of Two Rooms (Figure 2), at the lowest level the walls of the room as well as the bottleneck state are clustered together. The corners of the room are given their own individual clusters and then the center of the room is partitioned symmetrically with respect to the bottleneck state. The second level then merges most of the interiors of the individual rooms while still giving the corner states their individual clusters. The final level then merges the corner states into the walls of the room and gives three clusters which are the



(a) Empty room (b) Two rooms (c) Four rooms (d) Towers of Hanoi

Figure 1: The environments used in the experiments. (a) Empty Room: A simple environment with no obstacles. (b) Two Rooms: An environment divided into two connected rooms. (c) Four Rooms: A more complex environment divided into four connected rooms. (d) Towers of Hanoi: A classic puzzle environment where the goal is to move disks between pegs according to specific rules.

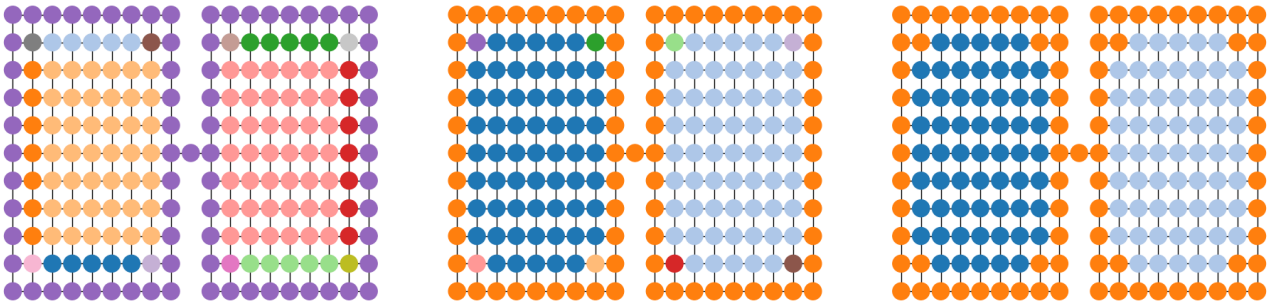


Figure 2: Hierarchical clustering of the Two Rooms environment at various levels. The lowest level (left-most image) clusters walls and bottleneck states, the second level (middle image) merges room interiors while keeping corners separate, and the final level (right-most image) combines corners into walls, resulting in three main clusters.

two interiors of the rooms and the final cluster essentially contains these two rooms.

4.3 Results

To compare with Evans et al. [3], in the analysis, we created options by building the full state transition graph and then learned their policies offline using macro-Q learning [5].

We trained all hierarchical agents with macro-Q learning and intra-option learning [7]. Shaded areas in the learning curves show the standard error, based on 40 independent trials.

The parameters set were the same as in [3], a learning rate of $\alpha = 0.4$, a discount factor of $\gamma = 1$, and initial action values of $Q_0 = 0$. An ϵ-greedy strategy with $\epsilon = 0.1$ was used for exploration. The shown learning curves represent evaluation performance. Post each training epoch, the policy was assessed (with exploration and learning disabled) in a separate environment instance.

We observe (Figure 3) that on the domain of Empty Room we quickly converge to a rewarding strategy before eventually seeing the Louvain skill hierarchy catch up. In the Two Room environment we observe much faster convergence as well as Louvain starting to catch up relatively slowly. The Four Rooms domain favours the Louvain skill hierarchy clearly, one might deduce this due to it being more important to traverse between rooms quickly which is optimally done through a clustering that relies on modularity. We outperform the Louvain skill hierarchy in the Towers of Hanoi, converging faster and having it catch up.

5 CONCLUSIONS

In this paper, we introduced a novel approach to hierarchical skill learning by leveraging Graphwave to obtain structural embeddings of the states. By clustering the states and preserving the topology of the state-space graph, we enabled efficient option learning, where options represent skills at various levels of abstraction. In our experiments, we compared the proposed approach to the state-of-the-art method by Evans et al. [3] and showed that our method can speed up the learning process significantly in some cases.

However, in some domains, optimizing for modularity obviously yields better skill hierarchies and faster option learning. It remains an open question for future research to determine which properties of a domain's state-space graph are more suited for each method. This question is related to another open challenge, namely the characterizations of a useful skill: for a given complex task, what defines a proper skill hierarchy.

Future work could also explore incrementally building the state-space graph and deriving the optimal skill hierarchy for the partially observed graph. This approach may influence how confidently the

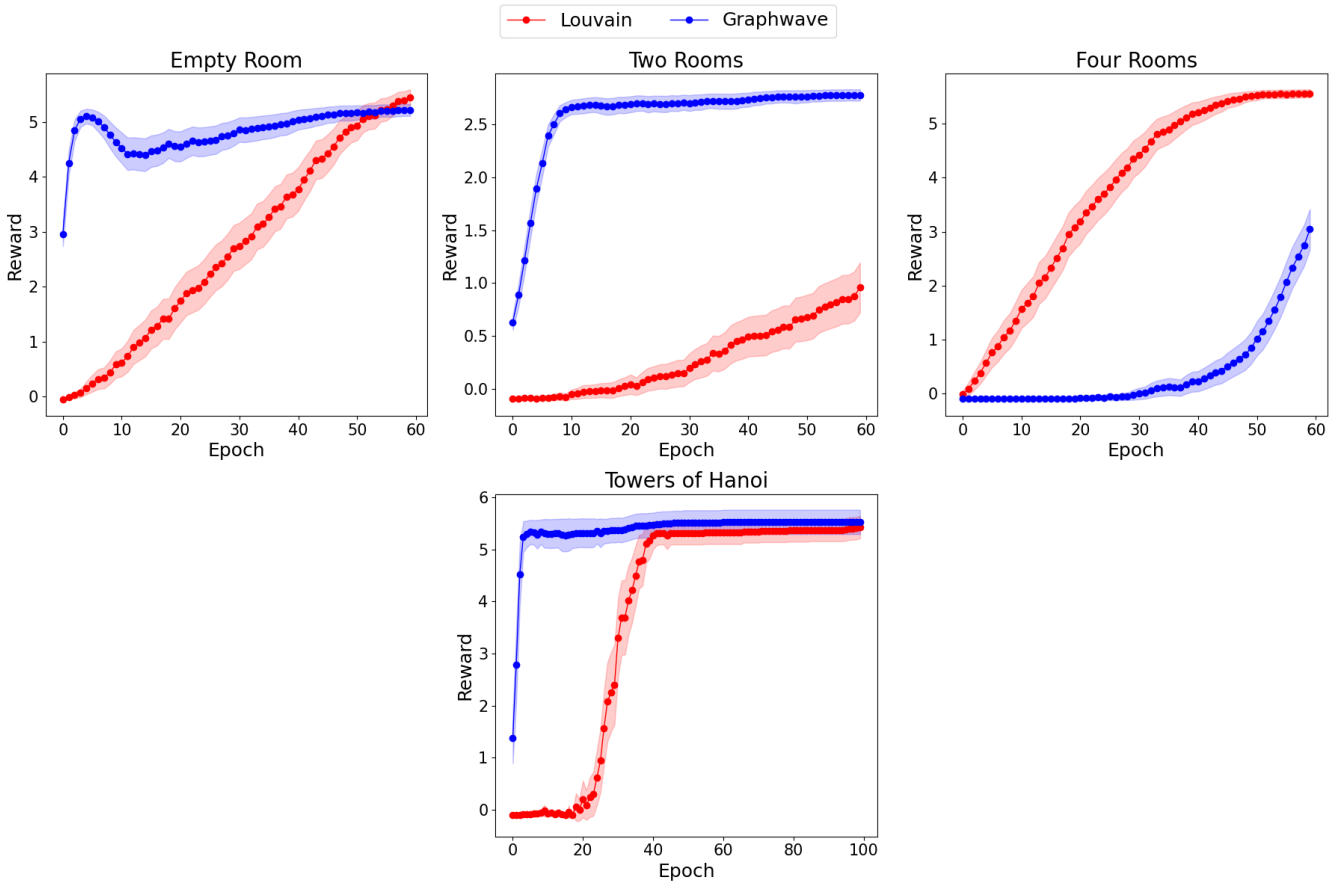


Figure 3: The following figure illustrates the performance of hierarchical agents using Louvain and Graphwave skill hierarchies in different environments: Empty Room, Two Rooms, Four Rooms, and Towers of Hanoi. We observe that in the Empty Room environment, both skill hierarchies converge quickly to a rewarding strategy, with Graphwave performing better initially but Louvain catching up over time. In the Two Rooms environment, Graphwave converges significantly faster than Louvain. The Four Rooms domain favors the Louvain skill hierarchy, likely due to the importance of quickly traversing between rooms using a clustering that relies on modularity. In the Towers of Hanoi, Graphwave outperforms Louvain, showing faster convergence and maintaining an advantage throughout. The provided plots show the reward progression over epochs for each environment, highlighting the differences in performance and convergence rates between the Louvain and Graphwave skill hierarchies.

partitioning is constructed over time, as new information becomes available and the graph evolves. One can develop algorithms that dynamically adjust the skill hierarchy based on the current state of the graph, ensuring that the hierarchy remains optimal as the environment changes. One may also pursue a similar direction in constructing skill hierarchies in problems that involve continuous state-spaces.

REFERENCES

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://doi.org/10.1609/aaai.v31i1.10916

[2] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings Via Diffusion Wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1320–1329. https://doi.org/10.1145/3219819.3220025 arXiv:1710.10321 [cs, stat]

[3] Joshua B. Evans and Özgür Şimşek. 2024. Creating Multi-Level Skill Hierarchies in Reinforcement Learning. arXiv:2306.09980 [cs]

[4] Marlos Machado, Andre Barreto, and Doina Precup. 2021. Temporal Abstraction in Reinforcement Learning with the Successor Representation.

[5] Amy Mcgovern, Richard Sutton, and Andrew Fagg. 1999. Roles of Macro-Actions in Accelerating Reinforcement Learning. (Feb. 1999).

[6] Matthew Riemer, Miao Liu, and Gerald Tesauro. 2018. Learning Abstract Options. *CoRR* abs/1810.11583 (2018). arXiv:1810.11583 http://arxiv.org/abs/1810.11583

[7] Richard S Sutton, Doina Precup, and Satinder Singh. 1998. Intra-Option Learning about Temporally Abstract Actions.. In *ICML*, Vol. 98. 556–564.

[8] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112, 1 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[9] Xiao Xu, Mei Yang, and Ge Li. 2018. Constructing Temporally Extended Actions through Incremental Community Detection. *Computational Intelligence and Neuroscience* 2018, 1 (2018), 2085721. https://doi.org/10.1155/2018/2085721