Supplementary Material for: Hierarchical Reinforcement Learning: A Comprehensive Survey

SHUBHAM PATERIA, Nanyang Technological University
BUDHITAMA SUBAGDJA and AH-HWEE TAN, Singapore Management University
CHAI QUEK, Nanyang Technological University

1 TASK DOMAINS FOR EVALUATING THE HIERARCHICAL REINFORCEMENT LEARNING (HRL) APPROACHES

For evaluation of the HRL approaches, it is necessary to use the task domains that are sufficiently challenging for standard RL, with complexities such as sparse rewards, long task horizon, requirement of intricate skills, and so on. A few interesting domains have emerged over the course of HRL research that present such complexities. These domains are listed as follows:

1.1 Classical Single-agent Grid-world Domains

The following domains are suitable for preliminary evaluation of an HRL approach applicable in discrete state and action spaces.

Grid-world Four rooms[1] [2, 15, 16, 25]: This is a spatial navigation domain. An agent has to traverse a discrete state space made up of spatial coordinates. The navigation space consists of four rooms separated by walls and connected by narrow doors/passages. The doors naturally form the bottlenecks (please refer to bottleneck discovery in Section 3.3.1 of the main article). A positive reward is only received at the goal location. A few relevant approaches for this domain are: References [2, 3, 4, 15, 17, 23, 25].

Taxi domain[2] [4]: This is an object collection and movement domain. The object is a "passenger" and a "taxi" agent has to navigate to the passenger, pick up, navigate to a destination, and drop the passenger. There are four locations, and positive rewards are given for picking up the passenger at one location and dropping off at another. A few relevant approaches for this domain are: References [4, 17, 23, 25].

1.2 Single-agent Continuous Control Domains

The continuous control domains commonly used in HRL research use the MuJoCo simulation engine [28]. The MuJoCo domains involve continuous state and action spaces.

MuJoCo Maze domain[3] [8, 19]: This is a spatial navigation domain. The continuous actions correspond to the limb and joint movements of an agent. The agent has to reach a goal state by navigating through a maze whose structure makes the navigation space constrained. This constraint, coupled with the large continuous state space, makes the maze domain quite challenging.

[1] https://github.com/jeanharb/option_critic/tree/master/fourrooms.
[2] https://gym.openai.com/envs/Taxi-v2/.
[3] https://github.com/tensorflow/models/tree/master/research/efficient-hrl.

© 2021 Association for Computing Machinery.
0360-0300/2021/05-ART109 $15.00
https://doi.org/10.1145/3453160

A positive reward is only received at the goal state. A few relevant approaches for this domain are: References [1, 5, 8, 13, 19, 20, 24].

MuJoCo Gather[4] [5, 19, 24]: This is an object collection domain with similar action space as the MuJoCo Maze. The agent has to navigate and collect apples. It receives a positive reward for each collected apple. The states corresponding to reaching and collecting each apple form the implicit bottlenecks (please refer to bottleneck discovery in Section 3.3.1 of the main article). A few relevant approaches for this domain are: References [1, 5, 13, 19, 20, 24].

MuJoCo Kitchen simulator[5] [9]: This is a complex continuous control domain requiring miscellaneous skills such as object collection, movement, and manipulation. The agent is a *robot arm* interacting with a kitchen scene that includes an openable microwave, four turnable oven burners, an oven light switch, a freely movable kettle, two hinged cabinets, and a sliding cabinet door. Different goals can be set in this environment, and positive rewards are only received at the goal states. This domain has been shown to be quite challenging even for a state-of-the-art HRL approach (HIRO [19]). A few relevant approaches for this domain are: References [1, 13, 19, 20, 24]

1.3 Complex Single-agent Discrete Action Games

Atari games (Arcade Learning Environment[6]*)* [2, 12, 15, 29]: The Atari games have been used as popular benchmarks for Deep Reinforcement Learning [18]. Few of these games, such as Montezuma's Revenge, Seaquest, Ms Pacman, and so on, are among the more challenging ones for the standard RL algorithms. These games involve various elements of difficulties, such as constrained navigation spaces, object manipulation, object collection, and so on, along with sparse rewards and high-dimensional state spaces (states observed as images). A few relevant approaches for these games are: References [2, 12, 15, 29].

Minecraft domain[7] [27]: Minecraft is yet another complex game that requires various skills, such as navigating through constrained spaces, collecting objects, manipulating objects, and so on. The state space is high-dimensional and the rewards are sparse. A special characteristic of this domain is that it can be used to create multiple tasks within the same environment through different configurations. Hence, **it is suitable for transfer learning research**. A few relevant approaches for this domain are: References [8, 27, 29].

1.4 Multi-agent Discrete Action Games

The domains listed below can be used to benchmark **Multi Agent HRL (MAHRL)** approaches. We also include a complex multiplayer game, Starcraft II, from the standard **Multi Agent RL (MARL)** research, which is also suitable for MAHRL.

Simple Team Sports Simulator (STS2)[8] [30]: The STS2 simulates soccer games between k agents each on two opposing teams. The state space consists of the coordinates of the agents, their velocities, ball possession, and so on. The agents may have to implicitly learn several complex coordination skills, such as passing the ball, organizing defense, attack, shooting for goal, and so on. The agents on a team are *homogeneous*. The team receives reward +1 for scoring a goal and −1 when the opponent scores. Yang et al. [30] use STS2 for evaluating their unified skill discovery approach called HSD (please refer to Section 3.5.2 of the main article). This domain is challenging enough for further benchmarking of MAHRL approaches. A few relevant approaches for this domain are: References [26, 30].

[4] https://github.com/florensacc/snn4hrl/tree/master/envs/mujoco/gather.
[5] https://github.com/google-research/relay-policy-learning.
[6] https://github.com/mgbellemare/Arcade-Learning-Environment.
[7] https://github.com/vkurenkov/hierarchical-skill-acquisition.
[8] https://github.com/electronicarts/SimpleTeamSportsSimulator; https://github.com/011235813/hierarchical-marl.

Starcraft Multi-agent Challenge (SMAC)[9]; *in MARL* [22]: SMAC is based on the popular real-time strategy game StarCraft II. It involves battle scenarios that require a variety of complex skills, such as coordinating attacks, shielding teammates, supporting teammates' health, and so on. When the positive rewards are associated with winning the battle, they are extremely sparse. So far, SMAC has only been benchmarked using standard MARL approaches [7, 21]. However, the requirement of various implicit and unpredictable skills also makes it an exciting domain for evaluating MAHRL approaches. A few relevant approaches for this domain are: References [26, 30].

2 PRACTICAL APPLICATIONS OF HRL

Although HRL is still an active area of research, it is gradually being applied to important real-world problems, a few of which are discussed below:

Disease diagnosis. Symptom checking is a fundamental component of a disease diagnosis process. Online symptom checkers have been deployed by sites such as WebMD and Mayo Clinic to identify possible causes and treatments for diseases. A symptom checker can be designed as an agent that solves a sequential decision problem. At each timestep, the agent chooses a symptom out of all possible ones to inquire the patient about. The patient then responds to the agent with a true/false answer for that particular symptom. The agent then presents a new inquiry based on the previous response. At the end of the diagnosis process, the agent predicts a disease that the patient may have from the set of all diseases. The two goals of a symptom-checker agent are to achieve a high accuracy of diagnosis and to minimize the inquiry length. Kao et al. [11] introduced an HRL approach in which the symptom checking is decomposed into subtasks according to the anatomical parts of the body, such as abdomen, arm, back, buttock, chest, and so on. The symptom checking for each anatomical part is treated as one subtask with its own symptom-checking policy. The higher-level policy chooses one of the anatomical parts given the current state, and then the subtask policy for that part performs the inquiries. A version of this symptom-checker agent is implemented within a system called DeepQ Tricorder, which was awarded second prize in the Qualcomm Tricorder XPrize Competition in 2017 [11].

Industrial Robotics. Robots are gradually finding their place in the manufacturing and warehousing industries for various repetitive tasks, such as stacking objects, order picking, packaging, arranging items on induction belts, and so on. These operations require manipulation of robotic parts, such as limbs and claws using long-horizon sequential actions. Hard-coding the robotic manipulation for every possible scenario in a factory or a warehouse is extremely challenging. Reinforcement learning provides a way to avoid hard-coding and enable the robots to learn the action policies under various scenarios. One of the recent successful applications of RL in industrial robotics comes from covariant.ai (https://covariant.ai/). The research work that underlies the conception of covariant.ai also includes a few of the key HRL approaches developed for robot skill learning [5, 14] and automatic goal/subgoal generation [6].

Fleet Management for ride-hailing platforms. Jin et al. [10] proposed a framework called CoRide for autonomous order dispatching and fleet management of ride-hailing vehicles. CoRide is developed in collaboration with the Didi Chuxing company (https://www.didiglobal.com). In their approach, a geographical area is first divided into hexagonal grids. Then, groups of six neighboring grids are marked as districts. Each grid becomes a reinforcement learning agent called *worker*. Similarly, each district becomes a high-level reinforcement learning agent called *manager*. Thus, each manager has six workers under its command in the HRL hierarchy. A manager agent (district) assigns subgoals to its workers. A subgoal is a transition vector in a latent state space. Each worker agent (grid) ranks the list of orders to be assigned to various vehicles such that the observed change

[9]https://github.com/oxwhirl/smac.

in the distribution of orders matches the provided subgoal. Various manager agents (districts) jointly learn to optimize the main goals of maximizing the city-level accumulated driver income and order response rate. CoRide is shown to successfully maximize these factors on the real-world historical transportation data from the Didi Chuxing company.

REFERENCES

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. 2018. Variational option discovery algorithms. arxiv:1807.10299 (2018).

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. AAAI Press, 1726–1734.

[3] Peter Dayan and Geoffrey E. Hinton. 1993. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 5. Morgan-Kaufmann, 271–278.

[4] Thomas G. Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Int. Res.* 13, 1 (Nov. 2000), 227–303.

[5] Carlos Florensa, Yan Duan, and Pieter Abbeel. 2017. Stochastic neural networks for hierarchical reinforcement learning. arxiv:1704.03012 (2017).

[6] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic goal generation for reinforcement learning agents. In *Proceedings of Machine Learning Research*, Vol. 80. PMLR, 1515–1528.

[7] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual multi-agent policy gradients. arxiv:1705.08926 (2017).

[8] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. 2017. Meta learning shared hierarchies. arxiv:1710.09767 (2017).

[9] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. 2020. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research)*, Vol. 100. PMLR, 1025–1037.

[10] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, Guobin Wu, and Jieping Ye. 2019. CoRide: Joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*. Association for Computing Machinery, New York, NY, 1983–1992. DOI: https://doi.org/10.1145/3357384.3357978

[11] Hao-Cheng Kao, Kai-Fu Tang, and Edward Y. Chang. 2018. Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI'18)*. AAAI Press, 2305–2313.

[12] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., Red Hook, NY, 3682–3690.

[13] Andrew Levy, George Dimitri Konidaris, Robert Platt Jr., and Kate Saenko. 2019. Learning multi-level hierarchies with hindsight. In *Proceedings of the 7th International Conference on Learning Representations*.

[14] Alexander C. Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. 2019. Sub-policy adaptation for hierarchical reinforcement learning. arxiv:1906.05862 (2019).

[15] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. 2017. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*. JMLR.org, 2295–2304.

[16] Amy McGovern and Andrew G. Barto. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 361–368.

[17] Ishai Menache, Shie Mannor, and Nahum Shimkin. 2002. Q-cut—Dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the 13th European Conference on Machine Learning (ECML'02)*. Springer-Verlag, Berlin, 295–306.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. DOI: https://doi.org/10.1038/nature14236

[19] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, 3307–3317.

[20] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Near-optimal representation learning for hierarchical reinforcement learning. arxiv:1810.01257 (2018).

[21] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. arxiv:1803.11485 (2018).

[22] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. arxiv:1902.04043 (2019).

[23] Özgür Şimşek and Andrew G. Barto. 2008. Skill characterization based on betweenness. In *Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS'08)*. Curran Associates Inc., Red Hook, NY, 1497–1504.

[24] Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Learning goal embeddings via self-play for hierarchical reinforcement learning. arxiv:1811.09083 (2018).

[25] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112, 1–2 (Aug. 1999), 181–211. DOI: https://doi.org/10.1016/S0004-3702(99)00052-1

[26] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. 2018. Hierarchical deep multiagent reinforcement learning. arxiv:1809.09332 (2018).

[27] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. AAAI Press, 1553–1561.

[28] E. Todorov, T. Erez, and Y. Tassa. 2012. MuJoCo: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. DOI: https://doi.org/10.1109/IROS.2012.6386109

[29] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*. JMLR.org, 3540–3549.

[30] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. 2020. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1566–1574.