Guiding Robot Exploration in Reinforcement Learning via Automated Planning

Yohei Hayamizu[1] **Saeid Amiri**[2], **Kishan Chandan**[2], **Keiki Takadama**[1], and **Shiqi Zhang**[2]
[1] The University of Electro-Communications, Tokyo, Japan
[2] The State University of New York at Binghamton, Binghamton, NY, USA
hayamizu@cas.lab.uec.ac.jp, keiki@inf.uec.ac.jp
{samiri1; kchanda2; zhangs}@binghamton.edu

1 Introduction

Reinforcement learning (RL) enables an agent to learn from trial-and-error experiences toward achieving long-term goals; automated planning aims to compute plans for accomplishing tasks using action knowledge. Despite their shared goal of completing complex tasks, the development of RL and automated planning has been largely isolated due to their different computational modalities. Focusing on improving RL agents' learning efficiency, we develop Guided Dyna-Q (GDQ) to enable RL agents to reason with action knowledge to avoid exploring less-relevant states. The action knowledge is used for generating artificial experiences from an optimistic simulation.

Researchers have developed algorithms that consolidate model-free RL and automated planning to avoid taking unreasonable actions in exploration (Ferreira et al. 2017; Efthymiadis and Kudenko 2013). The research on leveraging knowledge to improve RL agents' learning performance is introduced in the recent survey paper (Zhang and Sridharan 2020). One of the methods DARLING, leverages human knowledge to avoid risky or useless state visits in RL, and has been applied to grid world domains and mobile robot navigation (Leonetti, Iocchi, and Stone 2016). Such a domain where humans can provide an explicit world model includes potentially many similar tasks, rendering goal-independent methods more suitable. Once the world model is known, we can plan the optimal policy for tasks without extra sampling. We develop Guided Dyna-Q (GDQ) that helps the agent learn the world model while avoiding exploring less-relevant states. We have evaluated GDQ with 2D navigation tasks in simulation. Its results show that GDQ significantly improves the learning efficiency compared to existing model-based and model-free RL methods, including Q-Learning, Dyna-Q, and DARLING (Sutton and Barto 2018; Leonetti, Iocchi, and Stone 2016).[1]

2 Guided Dyna-Q Algorithm

The overview of the procedure of Guided Dyna-Q (GDQ) is shown in Figure 1. In particular, we use Answer Set Programming (ASP) to formulate action knowledge (Lifschitz 2002; Erdem, Gelfond, and Leone 2016), and use Dyna-Q for model-based RL (Sutton and Barto 2018).

We use $\Pi(\mathcal{S}^A, \mathcal{A}^A, M)$ to represent our automated planner, where $\mathcal{S}^A$ and $\mathcal{A}^A$ are the state and action sets respectively. A task to the automated planner is defined as $M = (s_0^A, s_G^A)$ where $s_0^A, s_G^A \in \mathcal{S}^A$ are the initial state and goal states respectively. Given task M, an automated planning system can use $\Pi(\mathcal{S}^A, \mathcal{A}^A, M)$ to compute a set of plans, $\mathcal{H}$, where $p \in \mathcal{H}$ is in the form of a sequence of state-action pairs, and each action sequence leads state transitions from the initial state s_0^A all the way to the goal state s_G^A.

$$p = \left\langle \left\langle s_0^A, a_0^A \right\rangle, \left\langle s_1^A, a_1^A \right\rangle, \cdots, \left\langle s_G^A \right\rangle \right\rangle \quad (1)$$

The plans computed by the automated planner are referred to as *optimistic* plans, because real-world domain uncertainty is frequently overlooked in building the planners. For instance, a robot taking the action of "*navigate to room R*" sometimes does not result in the robot being in *room R* due to the possibility of obstacles blocking the way. The goal of *optimistic initialization* (OPTINIT) is to use the plans computed by the automated planner to initialize Q-values, and prevent the agent from exploring less-relevant states.

Given the initialized Q-value function, the agent is able to compute an initial policy, and use this policy to interact with the real world. The interaction experience is used to update the Q-value function at runtime, along with the automated planner (POLICYUP). Intuitively, the automated planner serves as an optimistic simulator to enable the reinforcement learner to learn from interaction experience in simulation.

GDQ is simply an integration of the two sub-procedures for OptInit and repeatedly conducted runtime PolicyUP. Informally, OptInit helps the agent avoid the near-random exploration behaviors through a "warm start" enabled by our automated planner, and PolicyUP guides the agent to only try the actions that can potentially lead to the ultimate goal. Next, we validate GDQ, demonstrating experimental results from comparisons between GDQ and a number of baseline methods selected from the literature.

3 Experiments

This section focuses on experimentally evaluating that GDQ performs better than existing RL methods from the literature in cumulative reward. GDQ has been compared with

[1] The full paper of this abstract has been accepted for publication at ICAPS-2021 under the same title.

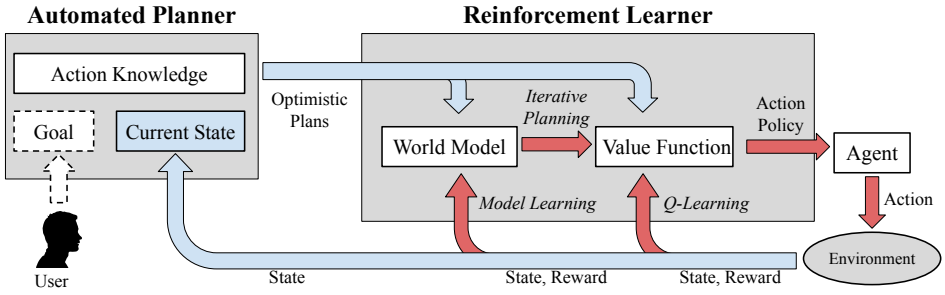

Figure 1: An overview of GDQ. The red-color loop corresponds to the Q-learning and Model learning loop. The agent (robot) interacts with the environment to update both its world model, and its Q-value function. The blue-color loop corresponds to the combination of an automated planner with the learning process where goal-independent action knowledge (highly sparse, and potentially inaccurate) is used for computing action sequences toward goal achievement.

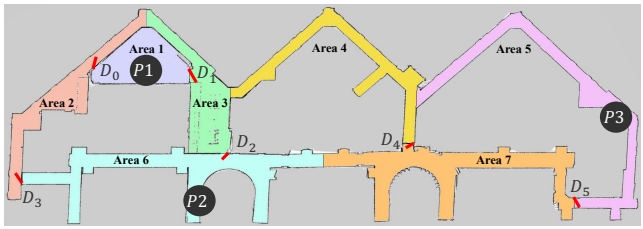

Figure 2: A map of an indoor office environment.

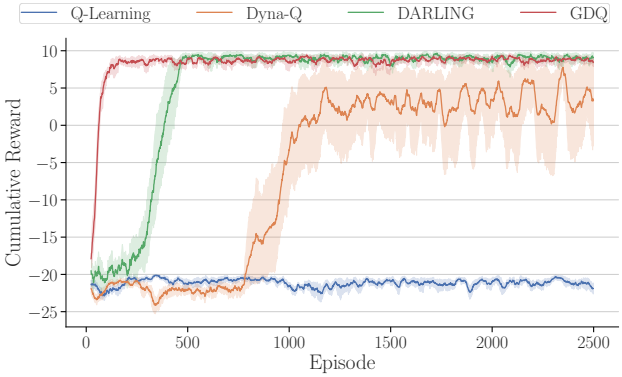

Figure 3: Average cumulative rewards of Task $M(P1, P2)$

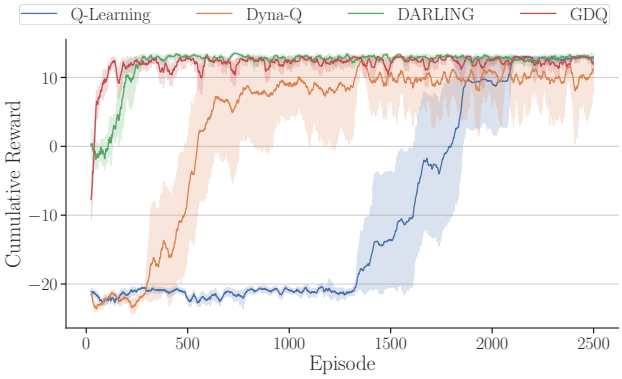

Figure 4: Average cumulative rewards of Task $M(P3, P1)$

Q-Learning, Dyna-Q, and DARLING (Leonetti, Iocchi, and Stone 2016) that reasons with action knowledge to avoid "unreasonable" exploratory behaviors.

We consider a mobile robot navigation domain, where the robot needs to navigate in an indoor office environment showed in Figure 2. In each trial (episode), the robot is tasked with navigating from its initial position X to a goal position Y, defined as $M(X, Y)$. The two tasks, $M(P1, P2)$ and $M(P3, P2)$, are conducted in this paper. There are doors connecting rooms and corridors, and there are different costs and success rates in navigation and door opening actions. The robot has four types of actions for navigational purposes. We have manually labeled six doors on the map. All doors are automatic, and the robot must get close to it and open it before taking the *gothrough* action. Each door is associated with a success-rate distribution, and another distribution over action costs. $D0$, $D2$, and $D5$ are difficult doors, where $D2$ is the most difficult to be opened. $D1$, $D3$, and $D4$ are easy, where $D3$ is the easiest.

Figures 3, 4 present the cumulative rewards collected from the robot conducting the tasks. These experiments were repeated 10 times for computing the averages and standard errors. We observe that GDQ performed the best in learning rate in comparison to the other three baselines. GDQ has the most sharpest learning curves at these results, supporting its learning efficiency.

4 Conclusion

In this paper, we develop Guided Dyna-Q (GDQ) that consolidates model-based RL and automated planning. The goal is to help the agent avoid exploring less-relevant states toward speeding up the learning process. GDQ has been demonstrated and evaluated in the simulation of an indoor office environment. From the experimental results, using the widely available action knowledge, GDQ performed significantly better than competitive baseline methods from the literature, demonstrating the best performance in learning efficiency.

References

Efthymiadis, K.; and Kudenko, D. 2013. Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *CIG*. IEEE.

Erdem, E.; Gelfond, M.; and Leone, N. 2016. Applications of answer set programming. *AI Magazine* 37(3): 53–68.

Ferreira, L.; Bianchi, R.; Santos, P.; and de Mantaras, R. L. 2017. Answer set programming for non-stationary markov decision processes. *Applied Intelligence* .

Leonetti, M.; Iocchi, L.; and Stone, P. 2016. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artif. Intell.* .

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence* 138(1-2): 39–54.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Zhang, S.; and Sridharan, M. 2020. A Survey of Knowledge-based Sequential Decision Making under Uncertainty. *arXiv preprint arXiv:2008.08548* .