Logical Composition for Lifelong Reinforcement Learning

Anonymous Authors[1]

Abstract

The ability to produce novel behaviours from existing skills is an important property of lifelong-learning agents. We build on recent work which formalises a Boolean algebra over the space of tasks and value functions, and show how this can be leveraged to tackle the lifelong learning problem. We propose an algorithm that determines whether a new task can be immediately solved using an agent's existing abilities, or whether the task should be learned from scratch. We verify our approach in the Four Rooms domain, where an agent learns a set of skills throughout its lifetime, and then composes them to solve a combinatorially large number of new tasks in a zero-shot manner.

1. Introduction

A major challenge in reinforcement learning (RL) is building general-purpose agents that are able to use existing knowledge to solve new tasks quickly. While RL has achieved recent success in a number of difficult, high-dimensional domains (Mnih et al., 2015; Levine et al., 2016; Lillicrap et al., 2016; Silver et al., 2017), these methods require millions of samples from the environment to learn optimal behaviours. This is ultimately a fatal flaw, since learning to solve complex, real-world tasks from scratch is typically infeasible.

One approach to improving sample-efficiency is *composition* (Todorov, 2009). This allows an agent to leverage its existing skills to build complex, novel behaviours, which can then be used to solve or speed up learning of a new task (Todorov, 2009; Saxe et al., 2017; Haarnoja et al., 2018; Van Niekerk et al., 2019; Hunt et al., 2019; Peng et al., 2019). More recently, Tasse et al. (2020) propose a framework for defining a Boolean algebra over the space of tasks and optimal value functions. This allows tasks and value functions to be composed using union, intersection and negation operators in a principled manner.

In this work, we demonstrate how such a framework can be used to tackle the lifelong learning problem. We assume that the agent is faced with a series of tasks drawn from some distribution, and that each task is specified by a vector specifying the goals of the task. Given this information, we propose an algorithm for iteratively solving such tasks, while at the same time constructing a *library* of skills which can be used to solve future tasks in a zero-shot manner.

We empirically verify our approach in the lifelong RL setting using the Four Rooms domain (Sutton et al., 1999), where an agent must determine what knowledge to add to its library and how to combine its current knowledge to solve new tasks. Results demonstrate that our approach is able to quickly learn a set of skills, resulting in a combinatorial explosion in the agent's abilities. Consequently, even when tasks are sampled randomly from some unknown distribution, an agent is able leverage its existing skills to solve new tasks without further learning.

2. Background

We consider tasks modelled by Markov Decision Processes (MDPs). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \rho, r, \gamma)$, where (i) $\mathcal{S}$ is the state space, (ii) $\mathcal{A}$ is the action space, (iii) ρ is a Markov transition kernel $(s, a) \mapsto \rho_{(s,a)}$ from $\mathcal{S} \times \mathcal{A}$ to $\mathcal{S}$, (iv) r is the real-valued reward function bounded by $[0, r_{\text{MAX}}]$, and (v) $\gamma \in [0, 1)$ is the discount factor, which determines the agents preference for short--term to long-term rewards. In this work, we focus on goal based tasks where an agent is required to reach some goal and so receives rewards r_{MAX} when it reaches desired goals and 0 everywhere else. We therefore consider the class of discounted MDPs with an absorbing set $\mathcal{G} \subseteq \mathcal{S}$.

The goal of the agent is to compute a Markov policy π from $\mathcal{S}$ to $\mathcal{A}$ that optimally solves a given task. A given policy π is characterised by a value function $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, specifying the expected return obtained under π starting from state s.[1] The *optimal* policy

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the 4th Lifelong Learning Workshop at ICML 2020. Do not distribute.

[1]Since we consider undiscounted MDPs, we can ensure the value function is bounded by augmenting the state space with a virtual state ω such that $\rho_{(s,a)}(\omega) = 1$ for all (s, a) in $\mathcal{G} \times \mathcal{A}$, and

π^* is the policy that obtains the greatest expected return at each state: $V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s)$ for all s in $\mathcal{S}$. A related quantity is the Q-value function, $Q^\pi(s,a)$, which defines the expected return obtained by executing a from s, and thereafter following π. Similarly, the optimal Q-value function is given by $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ for all s in $\mathcal{S}$ and a in $\mathcal{A}$.

3. Logical Composition

Tasse et al. (2020) recently propose the notion of a Boolean task algebra, which allows an agent to perform logical operations—conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$)—over the space of tasks. For clarity, we present the statement of the theorems here, but note that all proofs can be found in the original paper.

3.1. A Boolean Algebra for Tasks

Theorem 1. *(Tasse et al., 2020) Let $\mathcal{M}$ be a set of tasks. Define $\mathcal{M}_{MAX}, \mathcal{M}_{MIN} \in \mathcal{M}$ to be tasks with the respective reward functions*

$$r_{\mathcal{M}_{MAX}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$(s,a) \mapsto \max_{M \in \mathcal{M}} r_M(s,a)$$

$$r_{\mathcal{M}_{MIN}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$(s,a) \mapsto \min_{M \in \mathcal{M}} r_M(s,a)$$

Then, under the assumption of deterministic transition dynamics and sparse rewards, $\mathcal{M}$ forms a Boolean algebra with universal bounds $\mathcal{M}_{MIN}$ and $\mathcal{M}_{MAX}$ when equipped with the operators $\neg, \vee, \wedge$ given by:

$$\neg : \mathcal{M} \to \mathcal{M}$$
$$M \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{\neg M}, \gamma),$$

$$\vee : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$$
$$(M_1, M_2) \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{M_1 \vee M_2}, \gamma),$$

$$\wedge : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$$
$$(M_1, M_2) \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{M_1 \wedge M_2}, \gamma),$$

where

$$r_{\neg M} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$(s,a) \mapsto r_{\mathcal{M}_{MAX}}(s,a) - r_M(s,a)$$

$$r_{M_1 \vee M_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$(s,a) \mapsto \max\{r_{M_1}(s,a), r_{M_2}(s,a)\}$$

$$r_{M_1 \wedge M_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$(s,a) \mapsto \min\{r_{M_1}(s,a), r_{M_2}(s,a)\}$$

Theorem 1 allows us to compose existing tasks together to create new tasks in a principled way, and also gives us a notion of base tasks. If we know the set of goals upfront, then it is easy to select a minimal set of base tasks that can be composed to produce the largest number of composite tasks. We first assign a Boolean label to each goal in a table, and then use the columns of the table as base tasks. The desirable goals for each base task are then those goals with a value of 1 according to the Boolean table. For example, for a domain with four goals, we can select 2 base tasks which can be used to specify all 16 possible tasks. This is illustrated in Table 1.

In general, for $\mathcal{M}$ with finite $\mathcal{G}$, we need only a logarithmic number of base tasks $\lceil \log_2 |\mathcal{G}| \rceil$ (for $|\mathcal{G}| > 1$) to specify an exponential number of composed tasks $2^{|\mathcal{G}|} = |\mathcal{M}|$.

3.2. Extended Value Functions

Tasse et al. (2020) also define goal-oriented versions of the reward and value functions, over which a Boolean algebra can also be constructed. These are defined as follows:

Definition 1. *The extended reward function $\bar{r} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s,g,a) \mapsto \begin{cases} 0 & \text{if } g \neq s \in \mathcal{G} \\ r(s,a) & \text{otherwise,} \end{cases} \tag{1}$$

Definition 2. *The extended Q-value function $\bar{Q} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s,g,a) \mapsto \bar{r}(s,g,a) + \gamma \int_{\mathcal{S}} \bar{V}^{\bar{\pi}}(s',g)\rho_{(s,a)}(ds'), \tag{2}$$

where $\bar{V}^{\bar{\pi}}(s,g) = \mathbb{E}_{\bar{\pi}}\left[\sum_{t=0}^{\infty} \bar{r}(s_t, g, a_t)\right]$.

By not rewarding the agent for achieving goals different from the one it wanted to reach, the extended reward function has the effect of driving the agent to learn how to separately achieve all desirable goals. Note that the standard reward functions and value functions can be recovered from their extended versions (Tasse et al., 2020).

The greedy action can be extracted from the extended value function by first maximising over goals, and then selecting the maximising action: $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} \max_{g \in \mathcal{G}} \bar{Q}^*(s,g,a)$. If we consider the extended value function to be a set of standard value functions (one for each goal), then this is equivalent to first performing generalised policy improvement (Barreto et al., 2017), and then selecting the greedy action.

3.3. A Boolean Algebra for Value Functions

Similarly, a Boolean algebra can be constructed a set of optimal extended Q-value functions:

Theorem 2. *(Tasse et al., 2020) Let $\bar{\mathcal{Q}}^*$ be the set of optimal extended $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Define*

$r = 0$ after reaching ω.

Goals	M_a	M_b	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
g_0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
g_1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
g_2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
g_3	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 1. Boolean table for a domain with four goals. Each column represents a task, where **0** or **1** for goal g on task M means respectively reward of $r_M(g,a) = 0$ or $r_M(g,a) = r_{\text{MAX}}\ \forall a \in \mathcal{A}$. Note how the base tasks M_a and M_b can generate all other tasks by finite Boolean operations.

$\bar{Q}^*_{MIN}, \bar{Q}^*_{MAX} \in \bar{\mathcal{Q}}^*$ to be the optimal $\bar{Q}$-functions for the tasks $\mathcal{M}_{MIN}, \mathcal{M}_{MAX} \in \mathcal{M}$. Then $\bar{\mathcal{Q}}^*$ forms a Boolean algebra when equipped with the operators $\neg, \vee, \wedge$ given by:

$$\begin{aligned} \neg :\ & \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\ & \bar{Q}^* \mapsto \neg \bar{Q}^*, \end{aligned}$$

$$\begin{aligned} \vee :\ & \bar{\mathcal{Q}}^* \times \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\ & (\bar{Q}^*_1, \bar{Q}^*_2) \mapsto \bar{Q}^*_1 \vee \bar{Q}^*_2, \end{aligned}$$

$$\begin{aligned} \wedge :\ & \bar{\mathcal{Q}}^* \times \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\ & (\bar{Q}^*_1, \bar{Q}^*_2) \mapsto \bar{Q}^*_1 \wedge \bar{Q}^*_2, \end{aligned}$$

where

$$\begin{aligned} \neg \bar{Q}^* :\ & \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\ & (s,g,a) \mapsto \bar{Q}^*_{MAX}(s,g,a) - \bar{Q}^*(s,g,a) \end{aligned}$$

$$\begin{aligned} \bar{Q}^*_1 \vee \bar{Q}^*_2 :\ & \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\ & (s,g,a) \mapsto \max\{\bar{Q}^*_1(s,g,a), \bar{Q}^*_2(s,g,a)\} \end{aligned}$$

$$\begin{aligned} \bar{Q}^*_1 \wedge \bar{Q}^*_2 :\ & \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\ & (s,g,a) \mapsto \min\{\bar{Q}^*_1(s,g,a), \bar{Q}^*_2(s,g,a)\} \end{aligned}$$

3.4. Between Task and Value Function Algebras

Given a Boolean algebra over tasks and extended value functions, there exists an equivalence between the two. As a result, if we can write down a task under the Boolean algebra, we can immediately write down the optimal value function for the task.

Theorem 3. *(Tasse et al., 2020) Let $\mathscr{A} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathscr{A}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then $\mathscr{A}$ is a homomorphism.*

4. Lifelong Transfer Through Composition

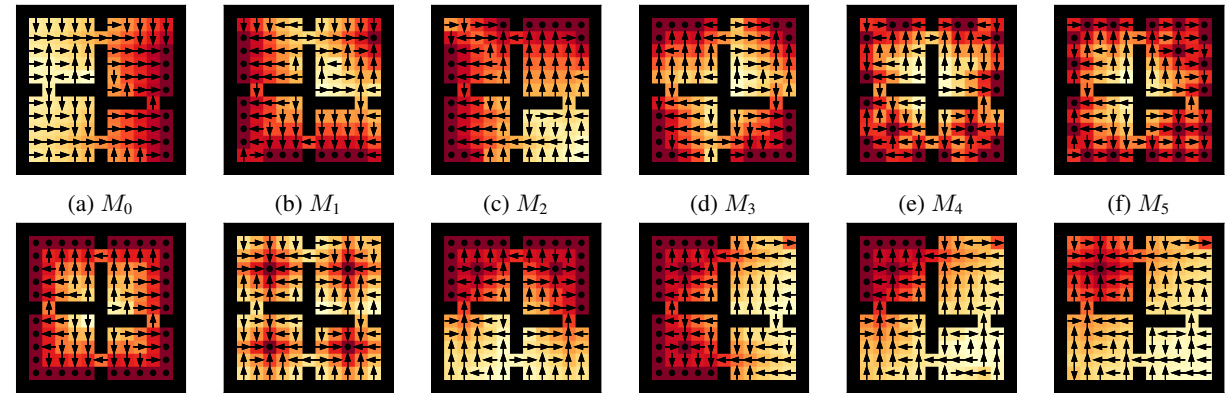
In lifelong RL, an agent is presented with a series of tasks sampled from some distribution $\mathcal{D}$. The goal of the agent is to transfer knowledge learned from previous tasks to solve new but related tasks quickly. We can use the theory developed in the previous sections to perform zero-shot transfer during lifelong RL by immediately solving new tasks that are expressible as logical compositions of previously learned tasks. We let $\mathcal{D}$ be an unknown distribution over (T, M) where

- $T \in \{0,1\}^{|\mathcal{G}|}$ is a binary vector specifying the task goals and
- $M = (\mathcal{S}, \mathcal{A}, \rho, r, \gamma) \in \mathcal{M}$ is an MDP defining a task.

The task parameter T describes the task that needs to be solved, and enables the agent to determine whether or not the current task is expressible as logical compositions of previously learned tasks.

We conduct a series of experiments in the Four Rooms domain (Sutton et al., 1999), where an agent must navigate in a grid world to particular locations. The goal locations are placed along the sides of the walls and at the center of rooms giving a goal space with $|\mathcal{G}| = 40$. The agent can move in any of the four cardinal directions at each timestep, but colliding with a wall leaves the agent in the same location. We add a 5th action for "stay" that the agent chooses to achieve goals. A goal location only becomes terminal if the agent chooses to stay in it. All rewards are 0 at non-terminal states, and 1 at the desirable goals. The transition dynamics are stochastic with a slip probability (*sp*). That is, with probability *1-sp* the agent moves in the direction it chooses, and with probability *sp* it moves in one of the other 3 chosen uniformly at random.

We begin by demonstrating zero-shot composition after an agent has learned all $\lceil \log_2 |\mathcal{G}| \rceil = 6$ base tasks. The base tasks are obtained by using a Boolean table similarly to Table 1.



(a) M_0 (b) M_1 (c) M_2 (d) M_3 (e) M_4 (f) M_5

(g) $(M_0 \wedge \neg(M_1 \wedge M_2)) \vee (\neg M_0 \wedge (M_2 \vee M_2 \vee M_3))$ (h) $\neg((M_0 \wedge \neg(M_1 \wedge M_2)) \vee (\neg M_0 \wedge (M_2 \vee M_2 \vee M_3)))$ (i) $\neg M_5 \vee \neg(M_1 \wedge M_2)$ (j) $\neg(M_1 \wedge (M_1 \vee M_2))$ (k) $(\neg M_5 \vee \neg(M_1 \wedge M_2)) \wedge (\neg(M_1 \wedge (M_1 \vee M_2)))$ (l) $\neg M_0 \wedge \neg M_1 \wedge \neg M_2 \wedge \neg M_3 \wedge \neg M_4 \wedge \neg M_5$

Figure 1. An example of zero-shot logical composition using the learned extended value functions. Arrows represent the optimal action in a given state. (a–f) The learned optimal goal oriented value functions for the base tasks. (g–l) Zero-shot compositions. Note that the resulting optimal value function can attain a goal not explicitly represented by the base tasks.

4.1. Zero-shot composition

We use a modified version of Q-learning (Watkins, 1989) to learn extended Q-value functions described previously. This algorithm is introduced in (Tasse et al., 2020) and differs in a number of ways from standard Q-learning: It keeps track of the set of terminating states seen so far, and at each timestep updates the extended Q-value function with respect to both the current state and action, as well as all goals encountered so far. It also uses the definition of the extended reward function, and so the agent receives 0 reward if it encounters a terminal state different from the one it wanted to reach.

Having learned the ϵ-optimal extended value functions for our base tasks, we can now leverage Theorems 1–3 to solve new tasks with no further learning. Figure 1 illustrates this composition, where an agent is able to solve complex task compositions without further learning. We illustrate a few composite tasks here, but note that after learning all the six base tasks the agent has enough information to solve all 2^{40} composite tasks. We demonstrate this by comparing the average returns from the optimal and the composed value functions of fifty random tasks whose goals are chosen uniformly at random over $\mathcal{G}$. To obtain the composed value function for each task M_i, we use the *sum of products* method to determine a Boolean expression for the task parameter T_i in terms of the task parameters of the learned base tasks. The value functions of the base tasks are then composed according to the generated Boolean expression. The results are given by Figure 2, and indicate that the policies obtained from our composition approach are either identical or very close to optimal.

4.2. Lifelong Transfer

We have demonstrated how an agent can solve any new task in an environment after training on the base tasks. However this requires the goal space be known upfront, and that the agent be pretrained on these tasks before solving new ones.

In this section we consider the more general setting where the agent is not necessarily given all the base tasks upfront, but rather presented with tasks sampled from some distribution. For each new task, the agent first determines if it is solvable using the knowledge learned so far. This can be done by first using the *Sum Of Products* method to obtain a candidate Boolean expression for the new task in terms of the learned tasks. If evaluating the Boolean expression gives returns the new task, then it is valid and can be used to obtain the composed value function zero-shot. If it is not valid, then the task is learned and added to the library of skills. The full pseudocode is shown in Algorithm 1.

We evaluate Algorithm 1 with Goal Oriented Q-learning as the learning method and for the following task distributions:

- $\mathcal{D}_{sampled}$: The goals for each task are chosen uniformly at random over $\mathcal{G}$.
- $\mathcal{D}_{best}$: The first $\lceil \log_2 |\mathcal{G}| \rceil$ tasks are the base tasks. The rest follow $\mathcal{D}_{sampled}$. This distribution gives the agent the minimum amount of tasks to learn and store since the agent learns the base tasks first before being presented with any other task.
- $\mathcal{D}_{worst}$: The first $|\mathcal{G}|$ tasks are each defined by a single goal that differs from the previous tasks. The rest

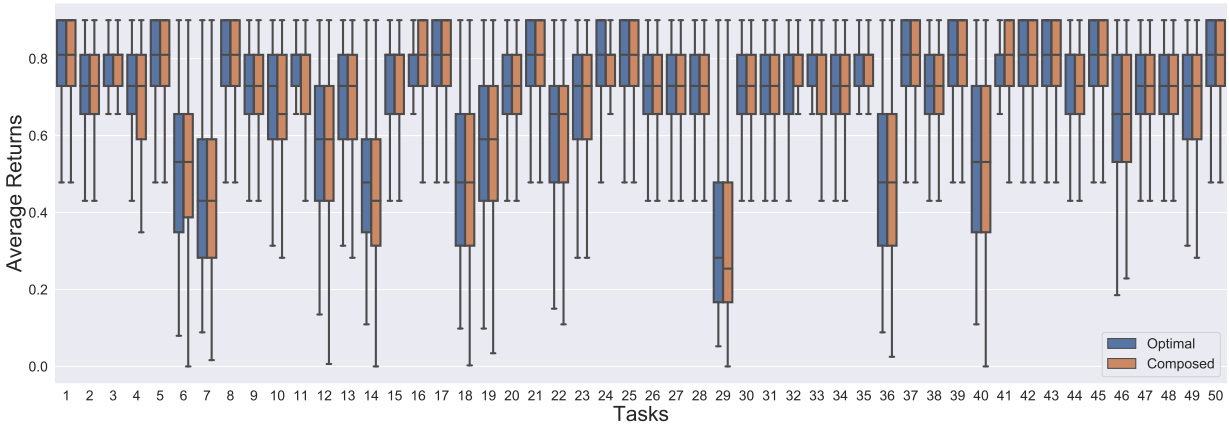


Figure 2. Box plots indicating average returns for optimal and composed value functions of 50 tasks sampled uniformly randomly over $\mathcal{M}$ after learning base tasks. Results are collected over 1000 episodes with random start positions.

Algorithm 1: Lifelong RL with Composition

Input: Learning method $\mathscr{A}$, task distribution $\mathcal{D}$
$\mathcal{T} \leftarrow \varnothing$
$\bar{\mathcal{Q}}^* \leftarrow \varnothing$
while *True* **do**
  $T, M \sim \mathcal{D}$
  $\mathcal{EXP} \leftarrow$ **SumOfProducts**$(\mathcal{T}, T)$
  if $T =$ ***Evaluate***$(\mathcal{EXP}, \mathcal{T})$ **then**
    `// Zero-shot transfer`
    $\bar{Q}^*_T \leftarrow$ **Evaluate**$(\mathcal{EXP}, \bar{\mathcal{Q}}^*)$
  else
    `// Solve and add to library`
    $\bar{Q}^*_T \leftarrow \mathscr{A}(M)$
    $\mathcal{T} \leftarrow \mathcal{T} \cup T$
    $\bar{\mathcal{Q}}^* \leftarrow \bar{\mathcal{Q}}^* \cup \bar{Q}^*_T$
  end
  $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} \max_{g \in \mathcal{G}} \bar{Q}^*_T(s, g, a)$
  Solve task M using $\pi^*(s)$ for all $s \in \mathcal{S}$
end

follow $\mathcal{D}_{sampled}$. This distribution forces the agent to learn and store the maximum number of tasks, since none of the $|\mathcal{G}|$ tasks can be expressed as a logical combination of the others.

We additionally use Q-learning with $max\bar{\mathcal{Q}}^*$ initialisation as a baseline. This has been shown by previous works (Abel et al., 2018; Barreto et al., 2017) to be a practical method of initialising value functions with a theoretically optimal optimism criterion that speeds-up convergence during training. We also use Q-learning for its learning method. The results are shown in Figure 3.

Our results shows that zero-shot logical composition enables a lifelong agent to quickly generalise over a task space. Notice how even the worst case task distribution for logical composition still outperforms the optimistic initialisation that maximises over the learned Q-functions.

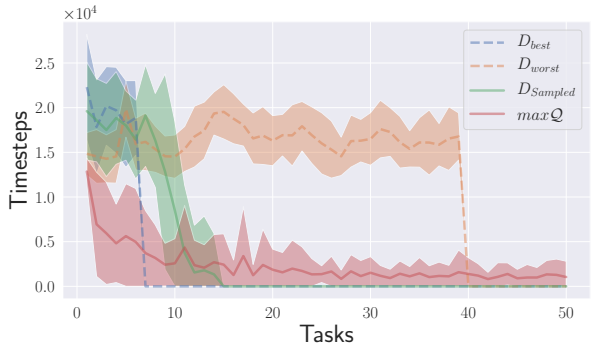


Figure 3. Number of samples required to solve tasks in the 40-goal FourRooms domain. Error bars represent standard deviations over 25 runs.

5. Related Work

The ability to compose value functions was first demonstrated using the linearly-solvable MDP framework (Todorov, 2007), where value functions could be composed to solve tasks similar to the disjunctive case (Todorov, 2009). Van Niekerk et al. (2019) show that the same kind of composition can be achieved using entropy-regularised RL (Fox et al., 2016), and extend the results to the standard RL setting, where agents can optimally solve the disjunctive case. Using entropy-regularised RL, Haarnoja et al. (2018) approximates the conjunction of tasks by averaging their reward functions, and demonstrates that by averaging the optimal value functions of the respective tasks, the agent can achieve performance close to optimal. Hunt et al. (2019) extends this result by composing value functions to

solve the average reward task exactly, which approximates the true conjunctive case. More recently, Peng et al. (2019) introduce a few-shot learning approach to compose policies multiplicatively. Although lacking theoretical foundations, results show that an agent can learn a weighted composition of existing base skills to solve a new complex task.

More recently, Tasse et al. (2020) showed that zero-shot optimal composition can be achieved for all Boolean operators in the stochastic shortest path setting. We have extended this result to discounted goal-based tasks, and have demonstrated the usefulness of the Boolean algebra framework for lifelong RL.

6. Conclusion

We have shown how tasks composed under a Boolean algebra can be immediately solved by first learning goal-oriented value functions, and then composing them in a similar manner. This enables agents in lifelong RL to quickly generalise over a task space. Finally, we note that our zero-shot composition framework can be used together with *MAXQINIT* (Abel et al., 2018) or other few-shot methods to speed up training when new tasks are not expressible as logical compositions of learned tasks. Our proposed approach is a step towards both interpretable RL—since both the tasks and optimal value functions can be specified using Boolean operators—and the ultimate goal of lifelong learning agents, which are able to solve combinatorially many tasks in a sample-efficient manner.

References

Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, pp. 20–29, 2018.

Barreto, A., Dabney, W., Munos, R., Hunt, J., Schaul, T., van Hasselt, H., and Silver, D. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *32nd Conference on Uncertainty in Artificial Intelligence*, 2016.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773*, 2018.

Hunt, J., Barreto, A., Lillicrap, T., and Heess, N. Composing entropic policies using divergence correction. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2911–2920. PMLR, 09–15 Jun 2019.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

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

Peng, X., Chang, M., Zhang, G., Abbeel, P., and Levine, S. MCP: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

Saxe, A., Earle, A., and Rosman, B. Hierarchy through composition with multitask LMDPs. *Proceedings of the 34th International Conference on Machine Learning*, 70: 3017–3026, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

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

Tasse, G. N., James, S., and Rosman, B. A boolean task algebra for reinforcement learning. *arXiv preprint arXiv:2001.01394*, 2020.

Todorov, E. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pp. 1369–1376, 2007.

Todorov, E. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pp. 1856–1864, 2009.

Van Niekerk, B., James, S., Earle, A., and Rosman, B. Composing value functions in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6401–6409. PMLR, 09–15 Jun 2019.

Watkins, C. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.