Globally Optimal Hierarchical Reinforcement Learning for Linearly-Solvable Markov Decision Processes

Guillermo Infante, Anders Jonsson, Vicenç Gómez
Dept. Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
{guillermo.infante,anders.jonsson,vicen.gomez}@upf.edu

Abstract

We present a novel approach to hierarchical reinforcement learning for linearly-solvable Markov decision processes. Our approach assumes that the state space is partitioned, and defines subtasks for moving between the partitions. We represent value functions on several levels of abstraction, and use the compositionality of subtasks to estimate the optimal values of the states in each partition. The policy is implicitly defined on these optimal value estimates, rather than being decomposed among the subtasks. As a consequence, our approach can learn the globally optimal policy, and does not suffer from non-stationarities induced by high-level decisions. If several partitions have equivalent dynamics, the subtasks of those partitions can be shared. We show that our approach is significantly more sample efficient than that of a flat learner and similar hierarchical approaches when the set of boundary states is smaller than the entire state space.

1 Introduction

A major challenge in reinforcement learning is to design agents that are able to learn efficiently and to adapt their existing knowledge to solve new tasks.

One way to reduce the complexity of learning is hierarchical reinforcement learning (Sutton, Precup, and Singh 1999; Dietterich 2000; Barto and Mahadevan 2003). By decomposing a task into subtasks, each of which can be solved independently, a solution to the original task can then be composed of the solutions to the subtasks. If each subtask is easier to solve than the original task, this may significantly reduce the learning effort of an agent that is learning to perform the task.

We consider Linearly-solvable Markov decision processes (LMDPs), a class of control problems whose Bellman optimality equations are linear in the (exponentiated) value function (Kappen 2005; Todorov 2006). Because of this, solution methods for LMDPs are more efficient than those for general Markov decision processes (MDPs). Though not as expressive as MDPs, LMDPs can nevertheless model a wide range of decision problems, and there exist methods for approximating MDPs with LMDPs (Todorov 2006).

LMDPs frequently appear under the names of path-integral or Kullback-Leibler control in the context of optimal control as probabilistic inference (Kappen, Gómez, and Opper 2012; Dvijotham and Todorov 2013; Kappen 2013). LMDPs are also strongly related to maximum-entropy reinforcement learning, which is known to have favorable properties and is quickly becoming the state-of-the-art for reinforcement learning (Ziebart 2010; Mnih et al. 2016; Haarnoja et al. 2018b; Levine 2018; Vieillard, Pietquin, and Geist 2020; Bas-Serrano et al. 2021).

One of the computational advantages of LMDPs is compositionality, which allows for zero-shot learning of new skills by linearly combining previously learned base skills which only differ in their cost or reward at boundary states (Todorov 2009; da Silva, Durand, and Popović 2009).

In this paper we propose a novel approach to hierarchical reinforcement learning in LMDPs that takes advantage of the compositionality of LMDPs. Our approach assumes that the state space is partitioned into subsets, and the subtasks consist in moving between these partitions. The subtasks are parameterized on the current value estimates of boundary states. Instead of solving the subtasks each time the value estimates change, we take advantage of compositionality to express the solution to an arbitrary subtask as a linear combination of a set of base LMDPs. The result is a form of value function decomposition which allows us to express an estimate of the optimal value of an arbitrary state as a combination of multiple value functions with smaller domains.

Concretely, our work makes the following contributions:

- We define a novel scheme based on compositionality for solving subtasks, defining local rewards that constitute a convenient basis for composite rewards.
- The subtask decomposition is at the level of the value function, not of the actual policy. Hence our approach does not suffer from non-stationarity in the online setting, unlike approaches that select among subtasks whose associated policies are being learned.
- Even though the subtasks have local reward functions, under mild assumptions our approach converges to the globally optimal value function.
- We analyze experimentally our proposed learning algorithm and show in two classical domains that it is more sample efficient compared to a flat learner and similar hierarchical approaches when the set of boundary states is smaller than the entire state space.

2 Related Work

Several authors have recently exploited concurrent compositionality of tasks in the context of transfer learning. Van Niekerk et al. (2019) use the linear compositionality of LMDPs to solve new tasks that can be expressed as combinations of a series of existing base tasks. They show that, while disjunctions of base tasks (OR-compositionality) can be performed exactly, the AND composition (when the goals of base tasks partially overlap) can only be performed approximately.

Haarnoja et al. (2018a) exploit a similar idea to transfer knowledge from existing tasks to new tasks by averaging their reward functions. Hunt et al. (2019) further extended this by introducing the so-called compositional optimism, and apply divergence correction in case compositionality does not transfer well.

More recently, Nangue Tasse, James, and Rosman (2020) derive a formal characterization of union and intersection of tasks in terms of Boolean algebra. They show that learning (extended) value functions that account for all achievable goals, exact zero-shot transfer learning using both AND- and OR- compositionality is possible, achieving an exponential increase in skills compared to the previous works.

All the aforementioned results are derived for general MDPs with *deterministic* dynamics and, possibly, entropy regularization. This setting is no more general than the class of LMDPs or path-integral control.

In this work, we aim to integrate both concurrent task composition, as done in the above approaches, together with hierarchical composition, where skills are chained in a temporal sequence, under the framework of LMDPs.

Several authors have proposed hierarchical versions of LMDPs. Jonsson and Gómez (2016) extend MAXQ (Dietterich 2000) to LMDPs by defining subtasks that represent high-level decisions. The top-level policy chooses multi-step transitions, which introduces non-stationarity in the high-level decision process if subtasks are learned concurrently, and also prevents global optimality. The authors discuss the idea of compositionality, but do not explore the concept further. Saxe, Earle, and Rosman (2017) propose a hierarchical multi-task architecture that does exploit compositionality. Their Multitask LMDP maintains a parallel distributed representation of tasks, reducing the complexity through stacking. However, the approach requires to augment the state space with many additional boundary (subtask) states. Further, the stacking introduces additional costs (cf. their Equation 10), and does not provide global optimality.

The Options Keyboard (Barreto et al. 2019) combines a successor feature representation with generalized policy improvement to obtain subtask policies from a set of base subtasks without learning, similar to our use of subtask compositionality. However, unlike in our approach, the composition weights have to be set manually, and although the composed policy is guaranteed to be better than the individual base policies, it is not guaranteed to be optimal.

Our work is similar to that of Wen et al. (2020) in that we define a hierarchical decomposition based on a partition of the state space, and exploit the equivalence of subtasks to reduce the learning effort. Unlike previous work, however, our approach is not restricted to single initial states, does not suffer from non-stationarity in the online setting, proposes a more general definition of equivalence that captures more structure, and guarantees convergence to the optimal value function for stochastic dynamics.

The concept of equivalent subtasks is strongly related to factored (L)MDPs, which capture conditional independence among a set of state variables (Boutilier, Dearden, and Goldszmidt 1995; Koller and Parr 2000). Equivalence arises whenever a subset of state variables are conditionally independent of another subset. Several authors have shown how to automatically discover the structure of factored MDPs from experience (Strehl, Diuk, and Littman 2007; Kolobov, Mausam, and Weld 2012), which in turn could be used to define equivalence classes of subtasks.

3 Background

Given a finite set $\mathcal{X}$, let $\Delta(\mathcal{X}) = \{p \in \mathbb{R}^{\mathcal{X}} : \sum_x p(x) = 1, p(x) \geq 0 \ (\forall x)\}$ denote the probability simplex on $\mathcal{X}$. Given a probability distribution $p \in \Delta(\mathcal{X})$, let $\mathcal{B}(p) = \{x \in \mathcal{X} : p(x) > 0\} \subseteq \mathcal{X}$ denote the support of p.

Linearly-Solvable Markov Decision Processes

A linearly-solvable Markov decision process, or LMDP (Kappen 2005; Todorov 2006), can be defined as a tuple $\mathcal{L} = \langle \mathcal{S}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \mathcal{J} \rangle$, where $\mathcal{S}$ is a set of non-terminal states, $\mathcal{T}$ is a set of terminal states, $\mathcal{P} : \mathcal{S} \to \Delta(\mathcal{S}^+)$ is an uncontrolled transition function, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is a reward function for non-terminal states, and $\mathcal{J} : \mathcal{T} \to \mathbb{R}$ is a reward function for terminal states. We use $\mathcal{S}^+ = \mathcal{S} \cup \mathcal{T}$ to denote the full set of states, and $S^+ = |\mathcal{S}^+|$ (resp. $S = |\mathcal{S}|$) to denote the number of (non-terminal) states. We also use $B = \max_{s \in \mathcal{S}} |\mathcal{B}(\mathcal{P}(\cdot|s))|$ to denote an upper bound on the support of P.

The learning agent follows a policy $\pi : \mathcal{S} \to \Delta(\mathcal{S}^+)$ that, for each non-terminal state $s \in \mathcal{S}$, chooses a probability distribution over next states in the support of $\mathcal{P}(\cdot|s)$, i.e. $\pi(\cdot|s) \in \Delta(\mathcal{B}(\mathcal{P}(\cdot|s)))$. In each round t, the learning agent observes a state $s_t \in \mathcal{S}^+$. If s_t is non-terminal, the agent transitions to a new state $s_{t+1} \sim \pi(\cdot|s_t)$ and receives an immediate reward

$$\mathcal{R}(s_t, \pi) = \mathcal{R}(s_t) - \lambda \cdot \mathrm{KL}(\pi(\cdot|s_t) \| \mathcal{P}(\cdot|s_t)),$$

where $\mathcal{R}(s_t)$ is the reward associated with state s_t, $\mathrm{KL}(\pi(\cdot|s_t) \| \mathcal{P}(\cdot|s_t))$ is the Kullback-Leibler divergence between $\pi(\cdot|s_t)$ and $\mathcal{P}(\cdot|s_t)$, and λ is a temperature parameter. Hence the agent can set the probability distribution $\pi(\cdot|s_t)$ freely, but gets penalized for deviating from the uncontrolled distribution $\mathcal{P}(\cdot|s_t)$. On the other hand, if s_t is terminal, the agent receives reward $\mathcal{J}(s_t)$ and then the current episode ends. The aim of the agent is to compute a policy π that maximizes the expected future reward (i.e. value), defined in each non-terminal state $s \in \mathcal{S}$ as

$$v^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{T-1} \mathcal{R}(S_t, \pi) + \mathcal{J}(S_T) \,\middle|\, S_1 = s \right].$$

Here, T is a random variable representing the time at which the current episode ends, and S_t is a random variable representing the state at time t. The expectation is over the stochastic choice of next state $S_{t+1} \sim \pi(\cdot|S_t)$ at each time t, and the time T it takes for the episode to end. We assume that the reward of all non-terminal states is negative, i.e. $\mathcal{R}(s) < 0$ for each $s \in \mathcal{S}$. As a consequence, $\mathcal{R}(s,\pi) < 0$ holds for any policy π, and the value $v^\pi(s)$ has a well-defined upper bound.

We are interested in computing the optimal value function $v^* : \mathcal{S} \to \mathbb{R}$, i.e. the maximum expected future reward among all policies. For simplicity, in what follows we omit the asterisks and refer to the optimal value function simply as the value function. We extend the value function to each terminal state $\mathrm{t} \in \mathcal{T}$ by defining $v(\mathrm{t}) \equiv \mathcal{J}(t)$. The value function v satisfies the Bellman equations

$$\frac{1}{\lambda}v(s) = \frac{1}{\lambda}\max_\pi \left[\mathcal{R}(s,\pi) + \mathbb{E}_{s'\sim\pi(\cdot|s)}v(s')\right]$$
$$= \frac{1}{\lambda}\mathcal{R}(s) + \max_\pi \mathbb{E}_{s'\sim\pi(\cdot|s)}\left[\frac{1}{\lambda}v(s') - \log\frac{\pi(s'|s)}{\mathcal{P}(s'|s)}\right] \quad \forall s.$$

We introduce the notation $z(s) = e^{v(s)/\lambda}$ for each $s \in \mathcal{S}^+$, and often abuse notation by referring to $z(s)$ as the (optimal) value of s. The maximization in the Bellman equations can be resolved analytically, yielding the following Bellman equations that are linear in z:

$$z(s) = e^{\mathcal{R}(s)/\lambda}\sum_{s'}\mathcal{P}(s'|s)z(s'). \quad (1)$$

We can express the Bellman equation in matrix form by defining an $S\times S$ diagonal reward matrix $R = \mathrm{diag}(e^{\mathcal{R}(\cdot)/\lambda})$ and an $S \times S^+$ stochastic transition matrix P whose entries (s,s') equal $\mathcal{P}(s'|s)$. We also define a vector $\mathbf{z}$ that stores the values $z(s)$ for each non-terminal state $s \in \mathcal{S}$, and a vector $\mathbf{z}^+$ extended to all states in $\mathcal{S}^+$. We can now write the Bellman equations in matrix form:

$$\mathbf{z} = RP\mathbf{z}^+. \quad (2)$$

Given z, the optimal policy π is given by the following expression for each pair of states (s,s'):

$$\pi(s'|s) = \frac{\mathcal{P}(s'|s)z(s')}{\sum_{s''}\mathcal{P}(s''|s)z(s'')}. \quad (3)$$

The solution for z corresponds to the largest eigenvector of RP. If the dynamics $\mathcal{P}$ and $\mathcal{R}$ are known, we can iterate (2) (Todorov 2006). Alternatively, we can incrementally learn an estimate $\hat{z}$ using stochastic updates based on state transitions sampled from the uncontrolled dynamics (s_t, r_t, s_{t+1})

$$\hat{z}(s_t) \leftarrow (1-\alpha_t)\hat{z}(s_t) + \alpha_t e^{r_t/\lambda}\hat{z}(s_{t+1}),$$

where α_t is a learning rate. The above update rule is called Z-learning (Todorov 2006) and suffers from slow convergence in very large state spaces and when the optimal policy differs substantially from the uncontrolled dynamics $\mathcal{P}$. A better choice is importance sampling, which uses samples from the estimated policy $\hat{\pi}$ derived from the estimated values $\hat{z}$ and (3) and updates $\hat{z}$ according to the following update

$$\hat{z}(s_t) \leftarrow (1-\alpha_t)\hat{z}(s_t) + \alpha_t e^{r_t/\lambda}\hat{z}(s_{t+1})\frac{\mathcal{P}(s_{t+1}|s_t)}{\hat{\pi}(s_{t+1}|s_t)}. \quad (4)$$

However, this requires local knowledge of $\mathcal{P}(\cdot|s_t)$ to correct for the different sampling distribution. Though this seems like a strong assumption, in practice $\mathcal{P}$ usually has a simple form, e.g. a random walk. Further, as shown in Jonsson and Gómez (2016), the corrected update rule in (4) can also be used to perform off-policy updates in case transitions are sampled using a policy different from $\hat{\pi}$,

Compositionality

Todorov (2009) introduced the concept of compositionality for LMDPs. Consider a set of LMDPs $\{\mathcal{L}_1,\ldots,\mathcal{L}_n\}$, where each LMDP $\mathcal{L}_i = \langle\mathcal{S},\mathcal{T},\mathcal{P},\mathcal{R},\mathcal{J}_i\rangle$ has the same components $\mathcal{S},\mathcal{T},\mathcal{P},\mathcal{R}$ and only differ in the reward $\mathcal{J}_i(\mathrm{t})$ of each terminal state $\mathrm{t} \in \mathcal{T}$, as well as its exponentiated value $z_i(\mathrm{t}) = e^{\mathcal{J}_i(\mathrm{t})/\lambda}$.

Now consider a new LMDP $\mathcal{L} = \langle\mathcal{S},\mathcal{T},\mathcal{P},\mathcal{R},\mathcal{J}\rangle$ with the same components as the n LMDPs above, except for $\mathcal{J}$. Assume that there exist weights $w_1,\ldots,w_n$ such that the exponentiated value of each terminal state $\mathrm{t} \in \mathcal{T}$ can be written as

$$e^{\mathcal{J}(\mathrm{t})/\lambda} = z(\mathrm{t}) = w_1 z_1(\mathrm{t}) + \ldots + w_n z_n(\mathrm{t}) = \sum_{k=1}^n w_k z_k(\mathrm{t}).$$

Since the Bellman optimality equation of each non-terminal state $s \in \mathcal{S}$ is linear in z, the optimal value of s satisfies the same equation:

$$z(s) = \sum_{k=1}^n w_k z_k(s).$$

Consequently, if we previously compute the optimal values $z_1,\ldots,z_n$ of the n LMDPs and know the weights $w_1,\ldots,w_n$, we immediately obtain the optimal values of the new LMDP $\mathcal{L}$ without learning.

4 Hierarchical LMDPs

In this section we describe our novel approach to hierarchical LMDPs. We first describe the particular form of hierarchical decomposition that we consider, and then present algorithms for solving a decomposed LMDP.

Hierarchical Decomposition

Our hierarchical decomposition is similar to that of Wen et al. (2020). Formally, given an LMDP $\mathcal{L} = \langle\mathcal{S},\mathcal{T},\mathcal{P},\mathcal{R},\mathcal{J}\rangle$, the set of non-terminal states $\mathcal{S}$ is partitioned into L subsets $\{\mathcal{S}_i\}_{i=1}^L$. For each such subset $\mathcal{S}_i$, we define an induced subtask $\mathcal{L}_i = \langle\mathcal{S}_i,\mathcal{T}_i,\mathcal{P}_i,\mathcal{R}_i,\mathcal{J}_i\rangle$, i.e. an LMDP whose components are defined as follows:

- The set of non-terminal states is $\mathcal{S}_i$.
- The set of terminal states $\mathcal{T}_i = \{\tau \in \mathcal{S}^+ \setminus \mathcal{S}_i : \exists s \in \mathcal{S}_i \text{ s.t. } \tau \in \mathcal{B}(\mathcal{P}(\cdot|s))\}$ includes all states in $\mathcal{S}^+ \setminus \mathcal{S}_i$ (terminal or non-terminal) that are reachable in one step from a state in $\mathcal{S}_i$.
- $\mathcal{P}_i : \mathcal{S}_i \to \Delta(\mathcal{S}_i^+)$ and $\mathcal{R}_i : \mathcal{S}_i \to \mathbb{R}$ are the restrictions of $\mathcal{P}$ and $\mathcal{R}$ to $\mathcal{S}_i$, where $\mathcal{S}_i^+ = \mathcal{S}_i \cup \mathcal{T}_i$ denotes the full set of subtask states.

- The reward of a terminal state $\tau \in \mathcal{T}_i$ equals $\mathcal{J}_i(\tau) = \mathcal{J}(\tau)$ if $\tau \in \mathcal{T}$, and $\mathcal{J}_i(\tau) = \hat{v}(\tau)$ otherwise, where $\hat{v}(\tau)$ is the estimated value in $\mathcal{L}$ of the non-terminal state in $\tau \in \mathcal{S} \setminus \mathcal{S}_i$.

Intuitively, if the reward $\mathcal{J}_i(\tau)$ of each terminal state $\tau \in \mathcal{T}_i$ equals its optimal value $v(\tau)$ for the original LMDP $\mathcal{L}$, then solving the subtask $\mathcal{L}_i$ yields the optimal values of the states in $\mathcal{S}_i$. In practice, however, we only have access to an estimate $\hat{v}(\tau)$ of the optimal value. In this case, the subtask $\mathcal{L}_i$ is *parameterized* on the value estimate $\hat{v}$ of terminal states in $\mathcal{T}_i$, and each time the value estimate changes, we can solve $\mathcal{L}_i$ to obtain a new value estimate $\hat{v}(s)$ for each state $s \in \mathcal{S}_i$.

We define a set of *exit states* $\mathcal{E} = \cup_{i=1}^{L} \mathcal{T}_i$, i.e. the union of the terminal states of each subtask in $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$. For convenience, we use $\mathcal{E}_i = \mathcal{E} \cap \mathcal{S}_i$ to denote the set of (non-terminal) exit states in the subtask $\mathcal{L}_i$. We also introduce the notation $K = \max_{i=1}^{L} |\mathcal{S}_i|$, $N = \max_{i=1}^{L} |\mathcal{T}_i|$ and $E = |\mathcal{E}|$.

Just like Wen et al. (2020), we define a notion of equivalent subtasks.

Definition 4.1 *Two subtasks $\mathcal{L}_i$ and $\mathcal{L}_j$ are equivalent if there exists a bijection $f : \mathcal{S}_i \to \mathcal{S}_j$ such that the transition probabilities and rewards of non-terminal states are equivalent through f.*

Unlike Wen et al. (2020), we do *not* require the sets of terminal states $\mathcal{T}_i$ and $\mathcal{T}_j$ to be equivalent. Instead, for each class of equivalent subtasks, our approach is to define a single subtask whose set of terminal states is the *union* of the sets of terminal states of subtasks in the class.

Formally, we define a set of equivalence classes $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_C\}$, $C \leq L$, i.e. a partition of the set of subtasks $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$ such that all subtasks in a given partition are equivalent. We represent a single subtask $\mathcal{L}_j = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j \rangle$ per equivalence class $\mathcal{C}_j \in \mathcal{C}$. The components $\mathcal{S}_j, \mathcal{P}_j, \mathcal{R}_j$ are shared by all subtasks in the equivalence class, while the set of terminal states is $\mathcal{T}_j = \bigcup_{\mathcal{L}_i \in \mathcal{C}_j} \mathcal{T}_i$, where the union is taken w.r.t. the bijection f relating all equivalent subtasks. As before, the reward $\mathcal{J}_j$ of terminal states is parameterized on a given value estimate $\hat{v}$. We assume that each non-terminal state $s \in \mathcal{S}$ can be easily mapped to its subtask $\mathcal{L}_i$ and equivalence class $\mathcal{C}_j$.

Example 1: Figure 1a) shows an example 4-room LMDP with a single terminal state marked F, separate from the room but reachable in one step from the highlighted location. The rooms are only connected via a single doorway; hence if we partition the states by room, the subtask corresponding to each room has two terminal states in other rooms, plus the terminal state F for the top right room. The 9 exit states in $\mathcal{E}$ are highlighted and correspond to states next to doorways, plus F. Figure 1b) shows a single subtask that is equivalent to all four room subtasks, since dynamics is shared inside rooms and the set of terminal states is the union of those of the subtasks. Hence the number of equivalent subtasks is $C = 1$, the number of non-terminal and terminal states of subtasks is $K = 25$ and $N = 5$, respectively, and the number of exit states is $E = 9$.

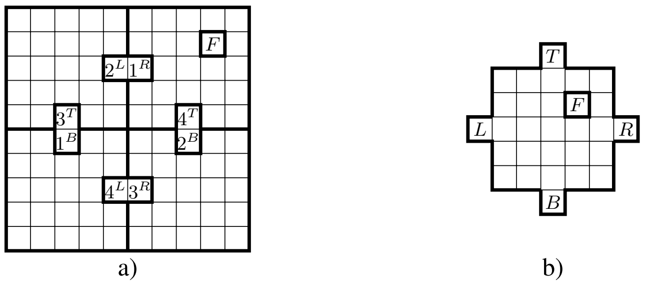

Figure 1: a) A 4-room LMDP, with a terminal state F and 8 other exit states; b) a single subtask with 5 terminal states F, L, R, T, B that is equivalent to all 4 room subtasks. Rooms are numbered 1 through 4, left-to-right, then top-to-bottom, and exit state 1^B refers to the exit B of room 1, etc.

Subtask Compositionality

During learning, the value estimate $\hat{v}$ changes frequently, and it is inefficient to solve all subtasks after each change. Instead, our approach is to use compositionality to obtain solutions to the subtasks without learning. The idea is to introduce several base LMDPs for each subtask $\mathcal{L}_j$ such that *any* reward function $\mathcal{J}_j$ can be expressed as a combination of the reward functions of the base LMDPs.

Given a subtask $\mathcal{L}_j = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j \rangle$ as defined above, assume that the set $\mathcal{T}_j$ contains n states, i.e. $\mathcal{T}_j = \{\tau_1, \ldots, \tau_n\}$. We define n base LMDPs $\mathcal{L}_j^1, \ldots, \mathcal{L}_j^n$, where each base LMDP is given by $\mathcal{L}_j^k = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j^k \rangle$. Hence the base LMDPs only differ in the reward of terminal states. Concretely, we define the exponentiated reward as $z_j^k(\tau) = 1$ if $\tau = \tau_k$, and $z_j^k(\tau) = 0$ otherwise. This corresponds to an actual reward of $\mathcal{J}_j^k(\tau) = 0$ for $\tau = \tau_k$, and $\mathcal{J}_j^k(\tau) = -\infty$ otherwise.

Even though the exit reward $\mathcal{J}_j^k(\tau)$ equals negative infinity for terminal states different from τ_k, this does not cause computational issues in the exponentiated space, since the value $z_j^k(\tau) = 0$ is well-defined in (2) and (3). Moreover, there are two good reasons for defining the rewards in this way. The first is that the rewards form a convenient basis that allows us to express *any* value estimate on the terminal states in $\mathcal{T}_j$ as a linear combination of $z_j^1, \ldots, z_j^n$. The second is that a value estimate $\hat{z}(\tau) = 0$ can be used to *turn off* terminal state τ, since the definition of the optimal policy in (3) assigns probability 0 to any transition that leads to a state τ with $\hat{z}(\tau) = 0$. This is the reason that we do not need the sets of terminal states to be equal for equivalent subtasks.

Now assume that we solve the base LMDPs to obtain the optimal value functions $z_j^1, \ldots, z_j^n$. Also assume a given value estimate $\hat{v}$ for the terminal states in $\mathcal{T}_j$, i.e. $\mathcal{J}_j(\tau) = \hat{v}(\tau)$ for each $\tau \in \mathcal{T}_j$. Then we can write the exponentiated reward $\hat{z}(\tau) = e^{\hat{v}(\tau)/\lambda}$ of each terminal state as

$$\hat{z}(\tau) = \sum_{k=1}^{n} w_k z_j^k(\tau) = \sum_{k=1}^{n} \hat{z}(\tau_k) z_j^k(\tau), \tag{5}$$

where each weight is simply given by $w_k = \hat{z}(\tau_k)$. This is because for a given terminal state $\tau_\ell \in \mathcal{T}_j$, the value $z_j^k(\tau_\ell)$ equals 0 for $k \neq \ell$, so the weighted sum simplifies to $w_\ell z_j^\ell(\tau_\ell) = w_\ell \cdot 1 = \hat{z}(\tau_\ell)$.

Due to compositionality, we can now write the estimated value of each non-terminal state $s \in \mathcal{S}_i$ as

$$\hat{z}(s) = \sum_{k=1}^{n} \hat{z}(\tau_k) z_j^k(s) \;\; \forall s \in \mathcal{S}_i, \forall \mathcal{L}_i \in \mathcal{C}_j. \qquad (6)$$

Here, the terminal states $\tau_1, \ldots, \tau_n$ are by definition exit states in $\mathcal{E}$. If we have access to a value estimate $\hat{z}_\mathcal{E} : \mathcal{E} \rightarrow \mathbb{R}$ on exit states, as well as the value functions $z_j^1, \ldots, z_j^n$ of all base LMDPs, we can thus use (6) to express the value estimate of each other state without learning. Hence (6) is a form of value function decomposition, allowing us to express the values of arbitrary states in $\mathcal{S}$ in terms of value functions with smaller domains. Concretely, there are $O(CN)$ base LMDPs, each with $O(M)$ values, so in total we need $O(CMN + E)$ values for the decomposition.

Example 1: In the 4-room example, there are five base LMDPs with value functions z^F, z^L, z^R, z^T and z^B, respectively. Given an initial value estimate $\hat{z}_\mathcal{E}$ for each exit state in $\mathcal{E}$, a value estimate of any state in the top left room is given by $\hat{z}(s) = \hat{z}_\mathcal{E}(1^B)z^B(s) + \hat{z}_\mathcal{E}(1^R)z^R(s)$, where we use $\hat{z}_\mathcal{E}(F) = \hat{z}_\mathcal{E}(L) = \hat{z}_\mathcal{E}(T) = 0$ to indicate that the terminal states F, L and T are not present in the top left room. We need $CMN = 125$ values to store the value functions of the 5 base LMDPs, and $E = 9$ values to store the value estimates of all exit states. Although this is more than the 100 states of the original LMDP, if we increase the number of rooms to $X \times Y$, the term CMN is a constant as long as all rooms have equivalent dynamics, and the number of exit states is $E = (2X - 1)(2Y - 1)$, which is much smaller than the $25XY$ total states. For 10×10 rooms, the value function decomposition requires 486 values to represent the values of 2,500 states.

The 4-room example is limited in the sense that changing the configuration and size of the rooms may break the assumption of equivalence, which in turn makes the hierarchical approach less powerful. However, the notion of equivalence is naturally associated with factored (L)MDPs, in which the state is factored into a set of variables $\mathcal{V} = \{v_1, \ldots, v_m\}$, i.e. $\mathcal{S} = \mathcal{D}(v_1) \times \cdots \times \mathcal{D}(v_m)$, where $D(v_i)$ is the domain of variable v_i, $1 \leq i \leq m$. Concretely, if there is a subset of variables $\mathcal{U} \subset \mathcal{V}$ such that the transitions among $\mathcal{U}$ are independent of the variables in $\mathcal{V} \setminus \mathcal{U}$, then it is natural to partition the states based on their assignment to the variables in $\mathcal{V} \setminus \mathcal{U}$. Consequently, there is a single equivalent subtask whose set of states is $\times_{v \in \mathcal{U}} D(v)$, i.e. all partial states on the variables in $\mathcal{U}$.

Example 2: The Taxi domain (Dietterich 2000) is described by three variables: the location of the taxi (v_1), and the location and destination of the passenger (v_2 and v_3). Since the location of the taxi is independent of the other two, it is natural to partition the states according to the location and destination of the passenger. Each partition consists of the possible locations of the taxi, defining a unique equivalent subtask whose terminal states are the locations at which the taxi can pick up or drop off passengers. Since there are 16 valid combinations of passenger location and destination, there are 16 such equivalent subtasks. Dietterich (2000) calls this condition *max node irrelevance*, where "max node" refers to a given subtask.

Eigenvector Approach

If the dynamics $\mathcal{P}$ and the state costs $\mathcal{R}, \mathcal{J}$ are known, we can use the power method to solve the original LMDP $\mathcal{L}$ by composing individual solutions of the subtask LMDPs $\mathcal{L}_i$. In this case, we define Bellman equations in (2) to solve the base LMDPs of all equivalence classes. To compute the values of the original LMDP $\mathcal{L}$ for the exit states in $\mathcal{E}$, the compositionality relation in (6) provides us with an additional system of linear equations, one for each non-terminal exit state. We can reformulate this additional system of equations in matrix form defined for the exit states $\mathbf{z}_\mathcal{E}$:

$$\mathbf{z}_\mathcal{E} = G\mathbf{z}_\mathcal{E}. \qquad (7)$$

Here, the matrix G contains the values of the base LMDPs according to (6). We can thus use the power method on this system of linear equations to obtain the values of all exit states in $\mathcal{E}$.

Example 1: In the 4-room example, the row in G corresponding to $\hat{z}_\mathcal{E}(2^L)$ contains the element $z^B(2^L)$ in the column for $\hat{z}_\mathcal{E}(1^B)$, and the element $z^R(2^L)$ in the column for $\hat{z}_\mathcal{E}(1^R)$, while all other elements equal 0. While the flat approach requires one run of the power method on a large matrix, our hierachical approach needs five runs of the power method on significantly reduced matrices (these runs can be parallelized), and one additional run on a 8×8 matrix, corresponding to (7).

We remark that we do not explicitly represent the values of states in $\mathcal{S} \setminus \mathcal{E}$ since they are given by (6). Since we can now obtain the value $z(s)$ of each state $s \in \mathcal{S}$, we can define the optimal policy directly in terms of the values z and (3). Hence unlike most approaches to hierarchical reinforcement learning, the policy does not select among subtasks, but instead depends directly on the decomposed value estimates.

Online and Intra-task Learning

In the online learning case, we need to maintain estimates $\hat{z}_j^1, \ldots, \hat{z}_j^n$ of the value functions of the base LMDPs associated with each equivalent subtask $\mathcal{L}_j$. These estimates can be updated using the Z-learning rule (4) after each transition. But to make learning more efficient, we can use a single transition (s_t, r_t, s_{t+1}) with $s_t \in \mathcal{S}_j$ to update the values of *all* base LMDPs associated with $\mathcal{L}_j$ simultaneously. This is known in the literature as intra-task learning (Kaelbling 1993; Jonsson and Gómez 2016).

Given the estimates $\hat{z}_j^1, \ldots, \hat{z}_j^n$, we could then formulate and solve the same system of linear equations in (6) to obtain the value estimates of exit states. However, it is impractical to solve this system of equations every time we update $\hat{z}_j^1, \ldots, \hat{z}_j^n$. Instead, we explicitly maintain estimates $\hat{z}_\mathcal{E}$ of the values of exit states in the set $\mathcal{E}$, and update these values incrementally. For that, we turn (6) into an update rule:

$$\hat{z}_{\mathcal{E}}(s) \leftarrow (1-\alpha_\ell)\hat{z}_{\mathcal{E}}(s) + \alpha_\ell \sum_{k=1}^{n} \hat{z}_j^k(s)\hat{z}_{\mathcal{E}}(\tau_k). \quad (8)$$

The question is when to update the value of an exit state. We propose several alternatives:

V_1: Update the value of an exit state $s \in \mathcal{E}_i$ each time we take a transition from s.

V_2: When we reach a terminal state of the subtask $\mathcal{L}_i$, update the values of all exit states in $\mathcal{E}_i$.

V_3: When we reach a terminal state of the subtask $\mathcal{L}_i$, update the values of all exit states in $\mathcal{E}_i$ and all exit states of subtasks in the equivalence class $\mathcal{C}_j$ of $\mathcal{L}_i$.

Again, the estimated policy π is defined directly by the value estimates $\hat{z}$ and (3), and thus does not select among subtasks. Below is the pseudo-code of the proposed algorithm.

Algorithm Online and Intra-Task Learning Algorithm

```
1: Input: An LMDP $\mathcal{L} = \langle \mathcal{S}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \mathcal{J} \rangle$ and a partition $\{\mathcal{S}_i\}_{i=1}^{L}$ of $\mathcal{S}$
   A set $\{\mathcal{C}_1, \dots, \mathcal{C}_C\}$ of equivalent subtasks and related base LMDPs $\mathcal{L}_j^k = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j^k \rangle$
2: Initialization:
   $\hat{z}_{\mathcal{E}}(s) := 1 \quad (\forall s \in \mathcal{E})$ {high-level Z function approximation}
   $\hat{z}_j^k(s) := 1$ {base LMDPs $1 \dots |\mathcal{T}_j|$ for each equivalent subtask $\mathcal{L}_j$}
3: while termination condition is not met do
4:     observe transition $s_t, r_t, s_{t+1} \sim \hat{\pi}(\cdot|s_t)$, where $s_t \in \mathcal{S}_i$ and $\mathcal{L}_i \in \mathcal{C}_j$
5:     update lower-level estimations $\hat{z}_j^k(s_t)$ using (4)
6:     if $s_t \in \mathcal{E}$ or $s_{t+1} \in \mathcal{T}_j$ then {$s_t$ is an exit or $s_{t+1}$ is terminal for current subtask $\mathcal{L}_j$}
7:         apply (8) to update $\hat{z}_{\mathcal{E}}$ using variant $V_1$, $V_2$ or $V_3$
8:     end if
9: end while
```

Analysis

Let $\mathcal{L} = \langle \mathcal{S}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \mathcal{J} \rangle$ be an LMDP, and let $\mathcal{L}_i = \langle \mathcal{S}_i, \mathcal{T}_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{J}_i \rangle$ be a subtask associated with the partition $\mathcal{S}_i \subseteq \mathcal{S}$. Let z denote the optimal value of $\mathcal{L}$, and let z_i denote the optimal value of $\mathcal{L}_i$.

Lemma 4.2 *If the reward of each terminal state $\tau \in \mathcal{T}_i$ equals its optimal value in $\mathcal{L}$, i.e. $z_i(\tau) = z(\tau)$, the optimal value of each non-terminal state $s \in \mathcal{S}_i$ equals its optimal value in $\mathcal{L}$, i.e. $z_i(s) = z(s)$.*

Proof Since $\mathcal{P}_i$ and $\mathcal{R}_i$ are the restriction of $\mathcal{P}$ and $\mathcal{R}$ onto $\mathcal{S}_i$, for each $s \in \mathcal{S}_i$ we have

$$\begin{aligned} z_i(s) &= e^{\mathcal{R}_i(s)/\lambda} \sum_{s'} \mathcal{P}_i(s'|s) z_i(s') \\ &= e^{\mathcal{R}(s)/\lambda} \sum_{s'} \mathcal{P}(s'|s) z_i(s'), \end{aligned}$$

which is the same Bellman equation as for $z(s)$. Since $z_i(\tau) = z(\tau)$ for each terminal state $\tau \in \mathcal{T}_i$, we immediately obtain $z_i(s) = z(s)$ for each non-terminal state $s \in \mathcal{S}_i$.

As an consequence of Lemma 4.2, assigning the optimal value $z(\tau)$ to each exit state $\tau \in \mathcal{E}$ yields a solution to (7), which is thus guaranteed to have a solution with eigenvalue 1. Lemma 4.2 also guarantees that we can use (6) to compute the optimal value of any arbitrary state given optimal values of the base LMDPs and the exit states. The only necessary conditions needed for convergence to the optimal value function is that (i) $\{\mathcal{S}_i\}_{i=1}^{L}$ is a proper partition of the state space; and (ii) the set of terminal states $\mathcal{T}_i$ of each subtask $\mathcal{L}_i$ includes all states reachable in one step from $\mathcal{S}_i$.

Lemma 4.3 *The solution to* (7) *is unique.*

Proof By contradiction. Assume that there exists a solution $\mathbf{z}'_{\mathcal{E}}$ which is different from the optimal values $\mathbf{z}_{\mathcal{E}}$. We can extend $\mathbf{z}$ and $\mathbf{z}'$ to all states in $\mathcal{S}$ by applying (6). Due to the same argument as in the proof of Lemma 4.2, the solution $\mathbf{z}'$ satisfies the Bellman optimality equation of all states in $\mathcal{S}$. Hence $\mathbf{z}'$ is an optimal value function for the original LMDP $\mathcal{L}$, which contradicts that $\mathbf{z}'$ is different from $\mathbf{z}$ since the Bellman optimality equations have a unique solution.

Lemma 4.4 *For each subtask $\mathcal{L}_i$ and state $s \in \mathcal{S}_i^+$, it holds that $z_i^1(s) + \cdots + z_i^n(s) \leq 1$.*

Proof By induction. The base case is given by terminal states $t_\ell \in \mathcal{T}_i$, in which case $z_i^1(t_\ell) + \cdots + z_i^n(t_\ell) = z_i^\ell(t_\ell) = 1$. For $s \in \mathcal{S}_i$, the Bellman equation for each base LMDP yields

$$\sum_{k=1}^{n} z_i^k(s) = e^{\mathcal{R}_i(s)/\lambda} \sum_{s'} \mathcal{P}(s'|s) \sum_{k=1}^{n} z_i^k(s').$$

Since $\mathcal{R}_i(s) = \mathcal{R}(s) < 0$ holds by assumption, and since $z_i^1(s') + \cdots + z_i^n(s') \leq 1$ holds for each s' by hypothesis of induction, it follows that $z_i^1(s) + \cdots + z_i^n(s) \leq 1$.

As a consequence, just like the matrix RP in (2), the matrix G in (7) has spectral radius at most 1, and hence the power method is guaranteed to converge to the unique solution with largest eigenvalue 1, corresponding to the optimal values of the exit states.

The convergence rate of the power method is exponential in $\gamma < 1$, the eigenvalue of RP or G with second largest value and independent of the state space. The average running time scales linearly with the number of non-zero elements in RP or G (Todorov 2006), which is drastically reduced compared to the non-hierarchical approach. More precisely, given an upper bound B on the support of $\mathcal{P}$ and a sparse representation, the matrix multiplication in (2) has complexity $\mathcal{O}(BS)$. In comparison, the matrix multiplication of the $\mathcal{O}(CN)$ base LMDPs has complexity $\mathcal{O}(BK)$, while the matrix multiplication in (7) has complexity $\mathcal{O}(NE)$. Hence the hierarchical approach is competitive whenever $\mathcal{O}(CNBK + NE)$ is smaller than $\mathcal{O}(BS)$. In a 10×10 room example, $CNBK + NE = 500 + 1{,}805 = 2{,}305$, while $BS = 10{,}000$.

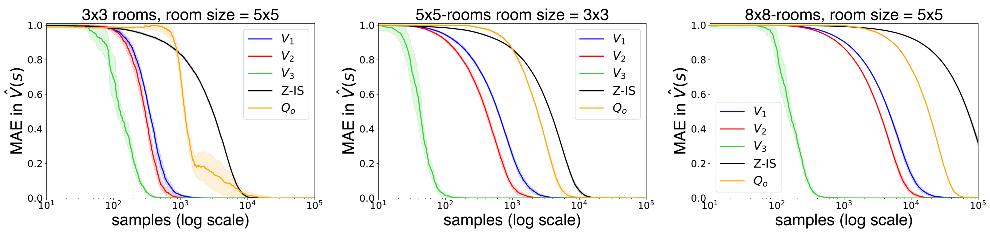

Figure 2: Results for 3×3 rooms of size 5×5 (left); 5×5 rooms of size 3×3 (center); 8×8 rooms of size 5×5 (right).

5 Experiments

We now evaluate the proposed learning algorithm in the two previous examples.[1] The objective of this evaluation is to analyze empirically the different update alternatives (V_1, V_2, and V_3), and to compare against a flat approach which exploits the benefits of LMDPs without the hierarchy (Z-IS), and the hierarchical approach based on options (Q_o) (Sutton, Precup, and Singh 1999). Our main objective is to empirically show that our approach is more sample efficient than the other algorithms. We run each algorithm with four different random seeds to analyze the average MAE (mean absolute error) against the optimal value function (computed separately) and its standard deviation over the number of samples. Since the value functions are different for Q-learning and LMDP methods, we present the self-normalized MAE (Figures 2 and 3) for different configurations and domains. Further, for a fair comparison between approaches, we only use the exit set for calculating the MAE.

In all experiments, the learning rates for each abstraction level is $\alpha_\ell(t) = c_\ell/(c_\ell + n)$ where n represents the episode each sample t belongs to. We empirically optimize the constant c_ℓ for each domain. For LMDPs, we use a temperature $\lambda = 1$, which provides good results. Q_o solves an equivalent MDP with *deterministic* actions, which should actually give it an advantage. For fairness, Q_o obtains the same per-step negative reward, exploits the same equivalence classes, learns the same subtasks (i.e. reach a terminal state), and has knowledge of which options are available in each state.

Rooms Domain. We analyze the performance for different room sizes and number of rooms (Figure 2). In all configurations the proposed hierarchical approach outperfoms Z-IS and Q_o. Concretely, Q_o suffers from non-stationarity: initial option executions will incur more negative reward than later executions, which causes high-level Q-learning updates to be *incorrect*, and it takes the learner significant time to recover from this.

Figure 2 (left) shows results for 3×3 rooms of size 5×5 and Figure 2 (center) shows results for 5×5 rooms of size 3×3. Both scenarios have 225 interior states. The difference between variants V_1, V_2 and V_3 is more pronounced in the second case, when the number of subtasks increases (more rooms) and the partition for each subtask is smaller (smaller rooms). Figure 2 (right) shows how the method scales with the number of rooms of size 5×5. Again, variant V_3 has the best performance, in this case by a larger margin than before.

[1]Code available at https://github.com/guillermoim/HRL_LMDP

Taxi Domain. To allow comparison between all the methods, we adapted the Taxi domain as follows: when the taxi is at the correct pickup location, it can transition to a state with the passenger in the taxi. In a wrong pickup location, it can instead transition to a terminal state with large negative reward (simulating an unsuccessful pick-up). When the passenger is in the taxi, it can be dropped off at any pickup location, successfully completing the task whenever dropped at the correct destination.

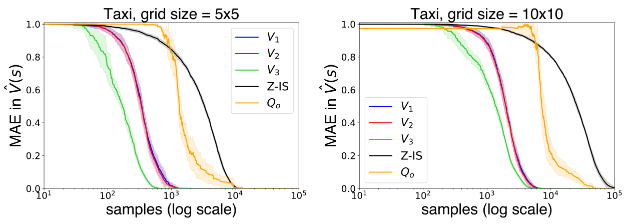

Figure 3: Results for 5×5 (left) and 10×10 (right) grids of Taxi domain.

Figure 3 shows results in two instances of size 5×5 (408 states) and 10×10 (1608 states). Again, the proposed hierarchical approach outperforms Z-IS and Q_o. In this case, the difference between V_1, V_2 and V_3 is less pronounced, even when the grid size increases. One possible explanation is the small number of exit states in this problem.

6 Discussion and Conclusion

In this paper we have introduced a novel approach to hierarchical reinforcement learning that focuses on the class of linearly-solvable Markov decision processes. Using subtask compositionality, we can decompose the value function and derive algorithms that converge to the optimal value function. To the best of our knowledge, our approach is the first to exploit both the concurrent compositionality enabled by LMDPs together with hierarchies and intra-task learning to obtain globally optimal policies efficiently.

The proposed hierarchical decomposition leads to a new form of zero-shot learning that allows to incorporate subtasks that belong to an existing equivalent class without additional learning effort. For example, adding new rooms in our example. This is in contrast with existing methods that only exploit linear compositionality of tasks.

Our approach is limited to OR compositionality of subtasks, but there is no fundamental limitation that prevents arbitrary compositions. The benefits of hierarchies can be combined for example, with the extended value functions proposed in Nangue Tasse, James, and Rosman (2020).

References

Barreto, A.; Borsa, D.; Hou, S.; Comanici, G.; Aygün, E.; Hamel, P.; Toyama, D.; Hunt, J.; Mourad, S.; Silver, D.; and Precup, D. 2019. The Option Keyboard: Combining Skills in Reinforcement Learning. In *Advances in Neural Information Processing Systems 32*, 13031–13041.

Barto, A. G.; and Mahadevan, S. 2003. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 13(1–2): 41–77.

Bas-Serrano, J.; Curi, S.; Krause, A.; and Neu, G. 2021. Logistic Q-Learning . In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, 3610–3618. PMLR.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1995. Exploiting Structure in Policy Construction. In *Proceedings of The 14th International Joint Conference on Artificial Intelligence*.

da Silva, M.; Durand, F.; and Popović, J. 2009. Linear Bellman Combination for Control of Character Animation. 28(3).

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res.*, 13: 227–303.

Dvijotham, K.; and Todorov, E. 2013. Linearly Solvable Optimal Control. In Lewis, F. L.; and Liu, D., eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, chapter 6, 119–141. John Wiley & Sons.

Haarnoja, T.; Pong, V.; Zhou, A.; Dalal, M.; Abbeel, P.; and Levine, S. 2018a. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, 6244–6251. IEEE.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018b. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 1861–1870. PMLR.

Hunt, J.; Barreto, A.; Lillicrap, T.; and Heess, N. 2019. Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, 2911–2920. PMLR.

Jonsson, A.; and Gómez, V. 2016. Hierarchical Linearly-Solvable Markov Decision Problems. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*.

Kaelbling, L. P. 1993. Learning to Achieve Goals. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1094–1099.

Kappen, H. J. 2005. Linear Theory for Control of Nonlinear Stochastic Systems. *Phys. Rev. Lett.*, 95: 200–201.

Kappen, H. J. 2013. Optimal control theory and the linear Bellman equation. In D. Barber, S. C., A. Taylan, ed., *Bayesian Time Series Models*, chapter 17, 363–387. Cambridge University Press.

Kappen, H. J.; Gómez, V.; and Opper, M. 2012. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2): 159–182.

Koller, D.; and Parr, R. 2000. Policy Iteration for Factored MDPs. In *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, 326–334.

Kolobov, A.; Mausam; and Weld, D. S. 2012. Discovering hidden structure in factored MDPs. *Artificial Intelligence*, 189: 19–47.

Levine, S. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, 1928–1937. PMLR.

Nangue Tasse, G.; James, S.; and Rosman, B. 2020. A Boolean Task Algebra for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, 9497–9507.

Saxe, A. M.; Earle, A. C.; and Rosman, B. 2017. Hierarchy through composition with multitask LMDPs. In *International Conference on Machine Learning*, 3017–3026. PMLR.

Strehl, A. L.; Diuk, C.; and Littman, M. L. 2007. Efficient Structure Learning in Factored-State MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 645–650.

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

Todorov, E. 2006. Linearly-solvable Markov decision problems. *Advances in Neural Information Processing Systems (NIPS)*, 1369–1376.

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

Van Niekerk, B.; James, S.; Earle, A.; and Rosman, B. 2019. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, 6401–6409. PMLR.

Vieillard, N.; Pietquin, O.; and Geist, M. 2020. Munchausen Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, 4235–4246.

Wen, Z.; Precup, D.; Ibrahimi, M.; Barreto, A.; Van Roy, B.; and Singh, S. 2020. On Efficiency in Hierarchical Reinforcement Learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*.

Ziebart, B. D. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. thesis, USA.