Supplementary Material for Hierarchy Through Composition with Multitask LMDPs

Andrew M. Saxe [1] Adam C. Earle [2] Benjamin Rosman [2 3]

1. Generality of the LMDP

We demonstrate here how more general non-navigation tasks can be modeled as LMDPs. The LMDP is defined by a three-tuple $L = \langle S, P, R \rangle$, where S is a set of states, P is a passive transition probability distribution $P : S \times S \rightarrow [0, 1]$, and R is an expected instantaneous reward function $R : S \rightarrow \mathbb{R}$.

The specific form of the LMDP can seem to limit the applicability of the framework. Yet as shown in a variety of recent work (Jonsson & Gómez, 2016), and in the examples given here, many standard domains can be translated to the MDP framework; and there exists a general procedure for embedding traditional MDPs in the LMDP framework (Todorov, 2009).

A key difference between the LMDP and standard MDPs is the fact that the cost function must include the KL term penalizing deviation from the 'passive dynamics.' While this may seem limiting, most domains of interest have some notion of 'efficient' actions, making a control cost a reasonably natural and universal phenomenon. Indeed, we suggest that for many real-world domains the standard MDP formulation is overly general, discarding useful structure in most real world domains–namely, a preference for efficient actions. Standard MDP formulations commonly place small negative rewards on each action to instantiate this efficiency goal, but they retain the flexibility to, for instance, prefer energetically inefficient trajectories by placing positive rewards on each action. The drawback of this flexibility is the unstructured maximization over actions in the standard Bellman equation, which prevents compositionality. In this sense, by accepting additional structure which is broadly present in real-world domains, the LMDP yields more explicit solutions. Here we show how several common domains may be modeled as LMDPs.

[1]Center for Brain Science, Harvard University [2]School of Computer Science and Applied Mathematics, University of the Witwatersrand [3]Council for Scientific and Industrial Research, South Africa. Correspondence to: Andrew M. Saxe <asaxe@fas.harvard.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

1.1. Tower of Hanoi

The passive dynamics approach is most natural when a domain has literal physical dynamics that occur in the absence of control inputs, such as the dynamics of a robot arm acted on by gravity or nearby positions in a spatial navigation task. However, the approach can also be used to model non-physical situations. The Tower of Hanoi puzzle (Simsek, 2008) demonstrates how an LMDP can model 'conceptual' tasks that do not possess physical dynamics. The puzzle consists of blocks of different sizes stacked on three pegs (Fig. 1A). The rules of the game are that a larger block may not be stacked on a smaller block. Fig. 1B shows how this setting may be instantiated and solved as an LMDP, by identifying a state with each unique configuration of blocks, and defining the passive dynamics such that the only nonzero transitions correspond to legal moves. The resulting LMDP can be solved to yield the optimal trajectories (Fig. 1C). With the MLMDP, new tasks like "place the medium size block on the middle peg" can be represented as a new pattern of boundary rewards over states (Fig. 1D), with the optimal solution flowing immediately from the compositionality property (Fig. 1E).

1.2. Taxi

The TAXI domain (Dietterich, 2000) is a common benchmark problem in hierarchical reinforcement learning, in which a taxi must navigate a $m \times n$ grid-world domain in which a subset of the states are demarcated as pick-up or drop-off locations (l) for one or more passengers (p) (see Fig. 2). The taxi may typically move in the four cardinal directions, remain at it's current state, execute a pick-up action when at one of the demarcated locations with the passenger, or execute the drop-off action when at one of the demarcated locations with the passenger on board. Here the agent must operate in the product space of the base domain $m \times n$, and the possible location-passenger configurations, l choose p, for a complete state-space of $|S| = (m \times n) \times$ (l choose p).

The passive transition dynamics P required by the LMDP framework are typically taken to be the resulting Markov chain under a uniformly random policy over actions. Explicitly we might begin by defining the transition dynamics

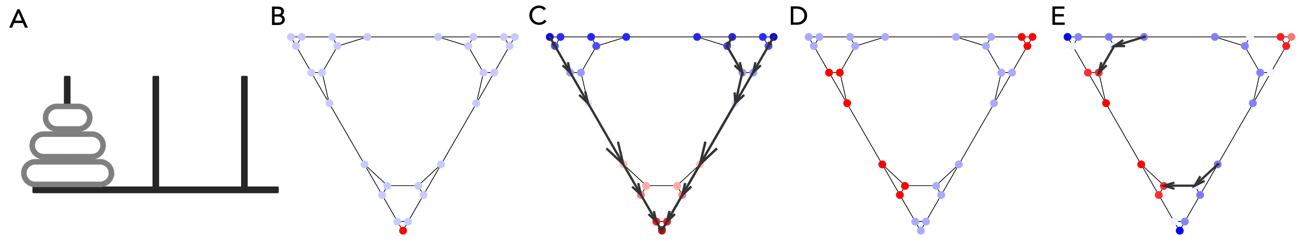

Figure 1. Tower of Hanoi. (A) The Tower of Hanoi puzzle contains blocks of different sizes and three pegs. A larger block may not be placed on a smaller block. The goal is to reform the stack of blocks on the rightmost peg. (B) State graph representing this task as an LMDP. Circles denote block configurations, edges denote transitions between states effected by valid moves, and the color of the circle indicates the boundary reward for reaching each state (red is higher reward). (C) Cost-to-go solution found via LMDP. Arrows depict trajectories, which all converge to the target configuration. (D) A different task in the same domain corresponds to a different reward structure across each state. Here the task is "place the medium size block on the middle peg". (E) Cost-to-go solution for this new task, found through composition using the MLMDP. Arrows denote trajectories.

for the base $m \times n$ grid-world domain which, under a uniformly random policy, results in the expected normalized state-frequency visits under a random walk. This is simply the banded matrix obtained by executing the up, down, left, right and stay actions.

We then note that transitions between the states factored by the l choose p possible passenger configurations can only be realized by executing either the pick-up or drop-off action. By way of example executing the pick-up action in state $(m_{l_i,P_j}, n_{l_i,P_j})$ transitions the state to $(m_{l_i,P_t}, n_{l_i,P_t})$ indicating that the agent is still in the same base domain state, but that the passenger is now in the taxi. The pick-up and drop-off actions, must be evaluated at all l choose p realizations of the base domain in order to map between all the factored states.

Boundary rewards are simply placed at the appropriate base domain state in the factor corresponding to the desired final location-passenger configuration.

Moreover, standard hierarchical task decompositions may be realized in our hierarchical framework. In the TAXI domain it is common to define a set of subtasks corresponding to the macro-actions of navigating the taxi to one of the specified locations (or the slightly more abstract GetPassenger action), see Fig. 2for a simple task-graph description of this. The full policy of getting a passenger at location A to location B may then we described by first executing the appropriate macro-action to navigate the taxi to A, then executing the pick-up action, then executing the macro-action to navigate the taxi to B, and finally executing the drop-off action. This hierarchy can be directly codified in our scheme.

Firstly we must define the subtask policy corresponding to navigating the taxi to location A (regardless of the passengers position). This task is realized by placing boundary rewards at the l choose p states of the full state-space corresponding to all of the states in which the taxi is at location A (one for each possible passenger location configuration). A similar procedure is followed to defined the subtask polices corresponding to navigating the taxi to the other locations. The pick-up and drop-off actions are already suitably defined. Solving the full MDP by solving this structured task decomposition mirrors the MAXQ approaches taken by other authors (Jonsson & Gómez, 2016). Critical to realizing these task decompositions in our framework is the fact that multiple states may be linked to a single subtask.

1.3. Robot visual servoing and warehouse scheduling

Finally, we note that several interesting domains have been formulated as LMDPs, including the Mountain Car (Todorov, 2006), warehouse scheduling (Jonsson & Gómez, 2016), and visual servoing on a robot (Kinjo et al., 2013).

2. Relationship to other hierarchical approaches

Compared to standard HRL schemes, our approach differs by relying on a parallel, distributed combination of tasks enabled by the composition property of the LMDP: A mid-level option in standard HRL chooses one of a discrete set of subtasks to execute, whereas a mid-level layer in our HRL scheme chooses a weighted task blend which drives behavior.

The work of Jonsson & Gómez (2016) develops a MAXQ hierarchy within the LMDP formalism, enabling powerful hierarchical decomposition. This approach differs from ours in the role compositionality plays. The MAXQ LMDP makes use of compositionality to let a subtask have more than one goal state by statically combining several single-goal subtasks. Our scheme uses compositionality as an

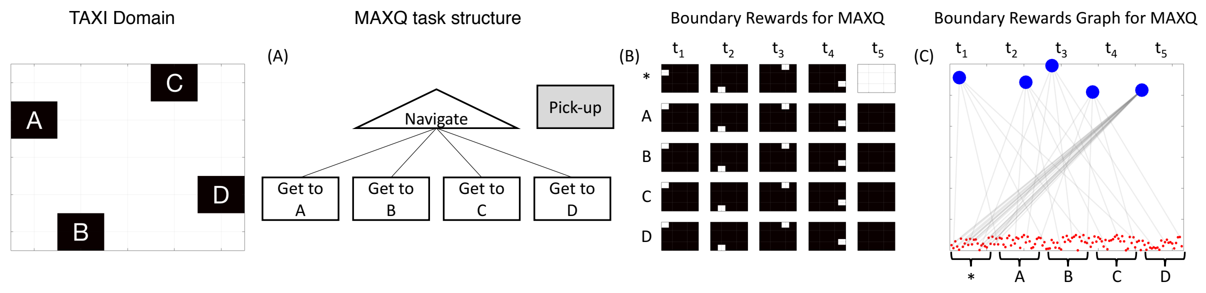

Figure 2. Defining MAXQ subtasks by their boundary reward structures. [A] The task-graph for the MAXQ decomposition. [B] Representation of the boundary reward structure for subtasks that implement a MAXQ-like task structure. Each small panel depicts the 5×5 grid of spatial locations. Each row corresponds to a different location of the passenger (there are five possible passenger locations: the location $*$ corresponds to the passenger being in the taxi, and locations A-D are the pick-up/drop-off points depicted in the taxi domain at left). Each column depicts the boundary reward for a specific subtask $\{t_1, \cdots, t_5\}$. White indicates high reward. By way of example, the subtask 'get to location A' in-paints boundary rewards at each of the states corresponding to the taxi being at location A in the base domain, regardless of where the passenger is located. Subtask t_5 corresponds to the pick-up action by in-painting boundary rewards into all states corresponding to the passenger being in the taxi. [C] An alternative view: the boundary reward graph for the subtasks. In this representation it is clear that subtasks $t_1, \cdots, t_4$ in-paint boundary rewards to exactly five states and collapse across passenger location, while t_5 in-paints rewards to all states in which the passenger is on board, collapsing across spatial location.

integral part of the execution model to enable a dynamic, changing blend of tasks, such that every subtask is always executing with different weightings. Future work might explore the potential of combining these schemes.

3. Computational complexity advantages of hierarchy

When solving multiple tasks, hierarchy can qualitatively improve the efficiency of the solution. Fig. 3A shows a simple domain consisting of N states arranged on a ring. The agent wishes to perform N tasks corresponding to navigating to each particular state. The transition dynamics are local, meaning that there is nonzero probability of transitioning only to adjacent states or remaining still. As noted in the main text, the hierarchical solution requires only $O(N \log N)$ iterations of z-learning, and requires only $O(N \log N)$ nonzero values to represent the value function for all tasks, compared to $O(N^2)$ for the flat case. In particular, following the formulation given in the main text, z-iteration requires

$$\begin{aligned} \sum_{l=1}^{D} M\left(\frac{N}{M^l} + \frac{N}{M^{l-1}}\right) &= N(1+M)\frac{1-(1/M)^D}{1-1/M} \\ &\leq N(1+M) \\ &\approx O(N \log N) \end{aligned}$$

iterations to propagate a value function signal to every state. Moreover, when the desirability functions are initialized to zero, K iterations of z-learning can only yield $O(K)$ nonzero elements in the desirability function, due to the local transition dynamics. Hence by the same analysis, the memory requirements are also $O(N \log N)$.

To verify these theoretical results numerically, we implement this domain as follows: transitions to adjacent states occur with probability $p = .2$, and the probability of remaining still is $1 - 2p$. The interior rewards for each state are $-.1$, the reward for reaching the goal state is 1, and the KL penalty parameter $\lambda = 1$. Finally, for the hierarchical formulation, we choose the probability of subtask transitions $\alpha = .1$, and place next-level states $M = \log N$ apart. Fig. 3B shows the propagation of the value function over successive iterations of z-learning. As can be seen, the front progresses roughly linearly due to the local transition structure. Fig. 3C-E show the resulting number of iterations, wall clock time, and memory usage for the flat and hierarchical cases. The flat solver exhibits roughly quadratic scaling while the hierarchical version is linearithmic.

4. Pseudocode for Hierarchical MLMDP

The execution structure of the hierarchical MLMDP algorithm is listed in Algorithm 1. Briefly, sampling a transition to a subtask state at level l causes the next higher level $l+1$ to activate and begin its own sampling process. Higher layers then communicate a task weighting w to lower layers.

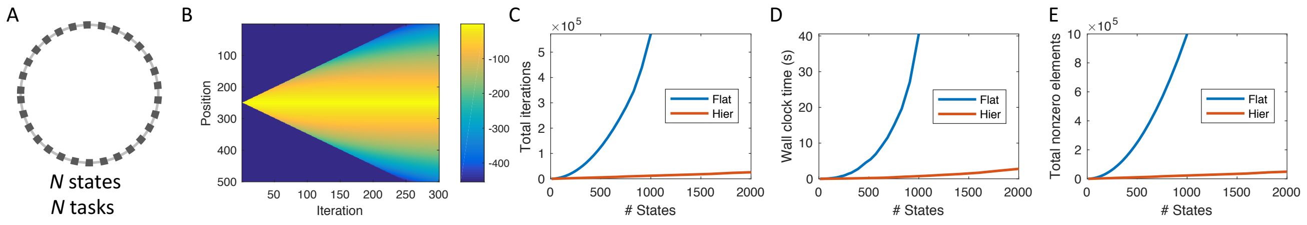

Figure 3. Computational complexity advantages of hierarchy with the Multitask LMDP. (A) A ring of N states in which transitions may only be made to adjacent states. The agent desires to perform N tasks corresponding to navigation to each position in the ring. (B) Cost-to-go function over successive z-iterations. Because of the local transition structure, the cost-to-go function progresses about one state on each iteration. Learning to reach a position K states away thus requires $O(K)$ iterations. (C) Total iterations required to learn all N tasks. (D) Wall clock time. (E) Total nonzero entries in cost-to-go representation (proportional to memory usage).

Algorithm 1 Hierarchical MLMDP

```
 1: Initialize variables $(z^1, \lambda, \cdots)$
 2: **while** Not at goal **do**
 3:     Compute controlled transition probability $P^1(z^1)$
 4:     Sample next state $\rightarrow s^1(t+1)$
 5:     **if** $s^1(t+1)$ is a base state **then**
 6:         Transition to next state
 7:         Update $z^1$ via Z-iteration
 8:     **else if** $s^1(t+1)$ is a higher layer state **then**
 9:         **for** $k = 2 \rightarrow L$ **do**
10:             Compute controlled transition probability $P^k(z^{k^*})$
11:             Sample next state $\rightarrow s^k(t+1)$
12:             **if** $s^k(t+1)$ is a subtask state **then**
13:                 Repeat
14:             **else**
15:                 Compute $r_b^{k-1}$ via Eqn.(10)
16:                 Compute $w^{k-1}$ via Eqn.(7)
17:                 Return
18:             **end if**
19:         **end for**
20:     **else**
21:         $s^1(t+1)$ *is a Terminal boundary state*
22:         Terminate episode
23:     **end if**
24: **end while**
```

References

Dietterich, T.G. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Jonsson, A. and Gómez, V. Hierarchical Linearly-Solvable Markov Decision Problems. In *ICAPS*, 2016.

Kinjo, K., Uchibe, E., and Doya, K. Evaluation of linearly solvable Markov decision process with dynamic model learning in a mobile robot navigation task. *Frontiers in neurorobotics*, 7 (April):7, 2013.

Simsek, O. *Behavioral building blocks for autonomous agents: Description, identification, and learning*. PhD thesis, University of Massachussetts, Amherst, 2008.

Todorov, E. Linearly-solvable Markov decision problems. In *NIPS*, 2006.

Todorov, E. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 7 2009.