Affordance-Aware Planning

David Abel & Gabriel Barth-Maron, James MacGlashan, Stefanie Tellex
Department of Computer Science, Brown University
{dabel,gabrielbm,jmacglashan,stefie10}@cs.brown.edu

***Abstract*—Planning algorithms for non-deterministic domains are often intractable in large state spaces due to the well-known "curse of dimensionality." Existing approaches to address this problem fail to prevent autonomous agents from considering many actions which would be obviously irrelevant to a human solving the same problem. We formalize the notion of affordances as knowledge added to an MDP that prunes actions in a state- and reward- general way. This pruning significantly reduces the number of state-action pairs the agent needs to evaluate in order to act optimally. We demonstrate our approach in the Minecraft domain as a model for robotic tasks, showing significant increase in speed and reduction in state-space exploration during planning. Further, we provide a learning framework that enables an agent to learn affordances through experience, opening the door for agents to learn to adapt and plan through new situations. We provide preliminary results indicating that the learning process effectively produces affordances that help solve an MDP faster, suggesting that affordances serve as an effective, transferable piece of knowledge for planning agents in large state spaces. We compare our approach with several existing knowledge engineering methods and deploy an affordance-aware planner on ReThink's Baxter as part of a cooperative cooking task.**

I. Introduction

As robots move out of the lab and into the real world, planning algorithms need to scale to domains of increased noise, size, and complexity. A classic formalization of this problem is a stochastic sequential decision making problem in which the agent must find a policy (a mapping from states to actions) for some subset of the state space that enables the agent to achieve a goal from some initial state, while minimizing any costs along the way. Increases in planning problem size and complexity directly correspond to an explosion in the state-action space, restricting extensions to large state-spaces such as robotic applications. Current approaches to solving sequential decision making problems in the face of uncertainty cannot tackle these problems as the state-action space becomes too large [8].

To address this state-space explosion, prior work has explored adding knowledge to the planner to solve problems in these massive domains, such as options [20] and macroactions [4, 15]. However, these approaches add knowledge in the form of additional high-level actions to the agent, which *increases* the size of the state-action space (while also allowing the agent to search more deeply within the space). The resulting augmented space is even larger, which can have the paradoxical effect of increasing the search time for a good policy [11]. Other approaches fall short of learning useful, transferable knowledge, either due to complexity or lack of generalizability.

Instead, we propose a formalization of *affordances* [7] for Markov Decision Processes (MDPs) that specifies which actions an agent should consider in different kinds of states to achieve a certain kind of goal. Our approach enables an agent to focus on aspects of the environment that are most relevant toward solving its current goal and avoids exploration of irrelevant parts of the state-action space, which leads to dramatic speedups in planning.

Further, we propose a learning process that enables agents to autonomously learn affordances through experience, lessening the agent's dependence on expert knowledge. Affordances are not specific to a particular reward function or state space, and provide the agent with transferable knowledge that is effective in a wide variety of problems. We call any planner that uses affordances an *affordance-aware* planner.

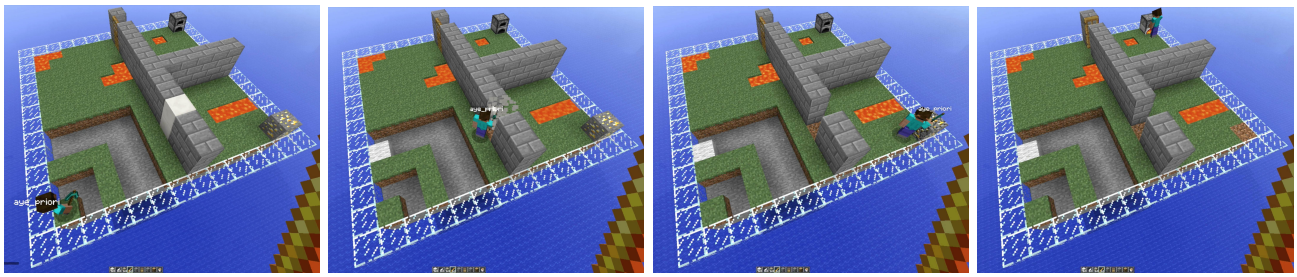
Because affordances define the *kind* of goals for which actions are useful, affordances also enable high-level reasoning that can be combined with approaches like subgoal planning for even greater performance gains. In our current model, ideal subgoals are sometimes given directly to planning agents by an expert - however, we are interested in automatically discovering subgoals in an online way, a problem which has already enjoyed some success [**? ?**].

II. Background

We use Minecraft as our planning and evaluation domain. Minecraft is a 3-D blocks world game in which the user can place and destroy blocks of different types. It serves as a model for a variety of robotic tasks involving assembly and construction. Minecraft's physics and action space are expressive enough to allow very complex systems to be created by users, including logic gates and functional scientific graphing calculators[1]; simple scenes from a Minecraft world appear in Figure 1 - a video demonstration of an early iteration of an affordance-aware planner solving this task may be seen online[2]. Minecraft serves as a model for robotic tasks such as cooking assistance, assembling items in a factory, and object retrieval. As in these tasks, the agent operates in a very large state-action space in an uncertain environment.

Minecraft is also an effective parallel for the actual world, both in terms of approximating the complexity and scope of planning problems, as well as modeling the uncertainty and noise presented to a robotic agent. For instance, robotic agents are prone to uncertainty throughout their system, including

[1] https://www.youtube.com/watch?v=wgJfVRhotlQ
[2] Watch at: https://vimeo.com/88689171



(a) Start (b) Destroy Wall (c) Collect Ore (d) Smelt Ore

Fig. 1. Affordance-aware RTDP tasked with a gold-smelting task with a variety of obstacles (only solved by an affordance-aware planner)

noise in their sensors (cameras, LIDAR, microphones, etc.), odometry, control, and actuation. In order to accurately capture some of the inherent difficulties of planning under uncertainty, the Minecraft agent's actions were modified to have stochastic outcomes. These stochastic outcomes may require important changes in the optimal policy in contrast to deterministic actions, such as keeping the agent's distance from high cost areas of the state-space, such as lava or cliffs.

We chose to give the Minecraft agent perfect sensor data about the Minecraft world. However, affordances typically relate to the agent's immediate surroundings, so limiting the perceptual scope should not impede the performance gains of affordances. We have considered extensions to Partially Observable domains, though at a distance solving a POMDP is effectively unchanged by the presence of affordances (beyond the performance gains provided by pruning actions).

A. OO-MDPs

We define affordances in terms of propositional functions on states. Our definition builds on the Object-Oriented Markov Decision Process (OO-MDP) [6]. OO-MDPs are an extension of the classic Markov Decision Process (MDP). A classic MDP is a five-tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a state-space; $\mathcal{A}$ is the agent's set of actions; $\mathcal{T}$ denotes $\mathcal{T}(s' \mid s, a)$, the transition probability of an agent applying action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and arriving in $s' \in \mathcal{S}$; $\mathcal{R}(s, a, s')$ denotes the reward received by the agent for applying action a in state s and and transitioning to state s'; and $\gamma \in [0, 1)$ is a discount factor that defines how much the agent prefers immediate rewards over distant rewards (the agent more greatly prefers to maximize more immediate rewards as γ decreases).

A classic way to provide a factored representation of an MDP state is to represent each MDP state as a single feature vector. By contrast, an OO-MDP represents the state space as a collection of objects, $O = \{o_1, \ldots, o_o\}$. Each object o_i belongs to a class $c_j \in \{c_1, \ldots, c_c\}$. Every class has a set of attributes $Att(c) = \{c.a_1, \ldots, c.a_a\}$, each of which has a domain $Dom(c.a)$ of possible values. Upon instantiation of an object class, its attributes are given a state $o.state$ (an assignment of values to its attributes). The underlying MDP state is the set of all the object states: $s \in \mathcal{S} = \cup_{i=1}^{o} \{o_i.state\}$.

Our motivation for using an OO-MDP lies in the ability to formulate predicates over classes of objects. That is, the OO-MDP definition also includes a set of predicates $\mathcal{P}$ that operate on the state of objects to provide additional high-level information about the MDP state.

While an OO-MDP reduces the size of the Minecraft state space by a significant factor, the resulting state space is still far too large to solve with any existing (OO)-MDP solver. This is the primary motivator for incorporating affordances - to reduce the amount of the state space that an OO-MDP agent will have to explore.

The Brown UMBC Reinforcement Learning And Planning framework (BURLAP[3]) is working toward integrating planning and reinforcement learning algorithms with a variety of planning domains represented as an OO-MDP, including ROS. In this way, transferable knowledge like affordances can be quickly deployed to domains like Mountain Car [**?**] and Minecraft, but also to a variety of Robots that utilize ROS. Our group is also working to deploy affordances as a means of knowledge representation and reasoning for collaborative cooking with ReThink's Baxter.

III. Related Work

In this section, we discuss the differences between affordance-aware planning and other forms of knowledge that have been used to accelerate planning.

A. Temporarily Extended Actions

Temporally extended actions are actions that the agent can select like any other action of the domain, except executing them results in multiple primitive actions being executed in succession. Two common forms of temporally extended actions are *macro-actions* [10] and *options* [20]. Macro-actions are actions that always execute the same sequence of primitive actions. Options are defined with high-level policies that accomplish specific sub tasks. For instance, when an agent is near a door, the agent can engage the 'door-opening-option-policy', which switches from the standard high-level planner to running a policy that is hand crafted to open doors.

Although the classic options framework is not generalizable to different state spaces, creating *portable* options is a topic of active research [14, 12, 17, 5, 1, 13].

[3]http://burlap.cs.brown.edu/

Given the potential for unhelpful temporally extended actions to negatively impact planning time [11], we believe combing affordances with temporally extended actions may be especially valuable because it will restrict the set of temporally extended actions to those useful for a task. We conducted a set of experiments to investigate this intuition.

B. Hierarchical Task Networks

D: I think we should have a shoutout to Branavan's Learning High Level Plans from Text paper in this section (and include subgoal planning as part of this section

C. Action Pruning

Sherstov and Stone [19] considered MDPs with a very large action set and for which the action set of the optimal policy of a source task could be transferred to a new, but similar, target task to reduce the learning time required to find the optimal policy in the target task. The main difference between our affordance-based action set pruning and this action transfer work is that affordances prune away actions on a state by state basis, where as the learned action pruning is on per task level. Further, with lifted goal descriptions, affordances may be attached to subgoal planning for a significant benefit in planning tasks where complete subgoal knowledge is known.

Rosman and Ramamoorthy [18] provide a method for learning action priors over a set of related tasks. Specifically, they compute a Dirichlet distribution over actions by extracting the frequency that each action was optimal in each state for each previously solved task.

There are a few limitations of the actions priors work that affordance-aware planning does not possess: (1) the action priors can only be used with planning/learning algorithms that work well with an ϵ-greedy rollout policy; (2) the priors are only utilized for fraction ϵ of the time steps, which is typically quite small; and (3) as variance in tasks explored increases, the priors will become more uniform. In contrast, affordance-aware planning can be used in a wide range of planning algorithms, benefits from the pruned action set in every time step, and the affordance defined lifted goal-description enables higher-level reasoning such as subgoal planning.

D. Temporal Logic

Bacchus and Kabanza [2, 3] provided planners with domain dependent knowledge in the form of a first-order version of linear temporal logic (LTL), which they used for control of a forward-chaining planner. With this methodology, STRIPS style planner may be guided through the search space by checking whether candidate plans do not falsify a given knowledge base of LTL formulas, often achieving polynomial time planning in exponential space.

The primary difference between this body of work and affordance-aware planning is that affordances may be learned (increasing autonomy of the agent), while LTL formulas are far too complicated to learn effectively, placing dependence on an expert.

E. Heuristics

Heuristics in MDPs are used to convey information about the value of a given state-action pair with respect to the task being solved and typically take the form of either *value function initialization*, or *reward shaping*. Initializing the value function to an admissible close approximation of the optimal value function has been shown to be effective for LAO* and RTDP [9].

Reward shaping is an alternative approach to providing heuristics. The planning algorithm uses a modified version of the reward function that returns larger rewards for state-action pairs that are expected to be useful, but does not guarantee convergence to an optimal policy unless certain properties of the shaped reward are satisfied [16].

A critical difference between heuristics and affordances is that heuristics are highly dependent on the reward function and state space of the task being solved, whereas affordances are state space independent and transferable between different reward functions. However, if a heuristic can be provided, the combination of heuristics and affordances may even more greatly accelerate planning algorithms than either approach alone.

IV. AFFORDANCES

A. Learning Affordances

V. EXPERIMENTS

VI. RESULTS

A. HTN/TLPlan Comparison

State Space	JSHOP2	Affordances
10 blocks	-	-
100 blocks	-	-
200 blocks	-	-
300 blocks	-	-
400 blocks	-	-
500 blocks	-	-

TABLE I
LEARNED AFFORDANCE RESULTS: AVG. NUMBER OF BELLMAN UPDATES PER CONVERGED POLICY

B. Baxter

Fig. 2. Placeholder for baxter results/image

C. Options

State Space	None	Options	Affordances	Both
4rooms	-	-	-	-
Doors	-	-	-	-
Small	-	-	-	-
Medium	-	-	-	-
Large	-	-	-	-

TABLE II
LEARNED AFFORDANCE RESULTS: CPU TIME PER CONVERGED POLICY

D. Minecraft: Expert vs Learned vs None

D: We may want to make this a bar chart to include error bars more naturally

State Space	No Affordances	Learned	Expert
Trench	-	-	-
Mining	-	-	-
Smelting	-	-	-
Wall	-	-	-
Tower	-	-	-

TABLE III
LEARNED AFFORDANCE RESULTS: AVG. NUMBER OF BELLMAN UPDATES PER CONVERGED POLICY

State Space	No Affordances	Learned	Expert
Trench	-	-	-
Mining	-	-	-
Smelting	-	-	-
Wall	-	-	-
Tower	-	-	-

TABLE IV
LEARNED AFFORDANCE RESULTS: CPU TIME PER CONVERGED POLICY

E. Minecraft: Learning rate

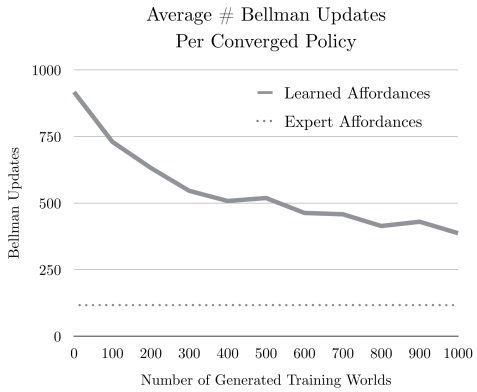


Fig. 3. Placeholder - will recollect this data given recent updates

VII. CONCLUSION

REFERENCES

[1] D. Andre and S.J. Russell. State abstraction for programmable reinforcement learning agents. In *Eighteenth national conference on Artificial intelligence*, pages 119–125. American Association for Artificial Intelligence, 2002.

[2] Fahiem Bacchus and Froduald Kabanza. Using temporal logic to control search in a forward chaining planner. In *In Proceedings of the 3rd European Workshop on Planning*, pages 141–153. Press, 1995.

[3] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116:2000, 1999.

[4] Adi Botea, Markus Enzenberger, Martin Müller, and Jonathan Schaeffer. Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24:581–621, 2005.

[5] T. Croonenborghs, K. Driessens, and M. Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. *Inductive Logic Programming*, pages 88–97, 2008.

[6] C. Diuk, A. Cohen, and M.L. Littman. An object-oriented representation for efcient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, 2008.

[7] JJ Gibson. The concept of affordances. *Perceiving, acting, and knowing*, pages 67–82, 1977.

[8] Matthew Grounds and Daniel Kudenko. Combining reinforcement learning with symbolic planning. In *Proceedings of the 5th, 6th and 7th European conference on Adaptive and learning agents and multi-agent systems: adaptation and multi-agent learning*, ALAS '05, 2005.

[9] Eric A Hansen and Shlomo Zilberstein. Solving markov decision problems using heuristic search. In *Proceedings of AAAI Spring Symposium on Search Techniques from Problem Solving under Uncertainty and Incomplete Information*, 1999.

[10] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 220–229. Morgan Kaufmann Publishers Inc., 1998.

[11] Nicholas K. Jong. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008.

[12] G. Konidaris and A. Barto. Efficient skill learning using abstraction selection. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, pages 1107–1112, 2009.

[13] G. Konidaris, I. Scheidwasser, and A. Barto. Transfer in reinforcement learning via shared features. *The Journal of Machine Learning Research*, 98888:1333–1371, 2012.

[14] George Konidaris and Andrew Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, IJCAI '07, pages 895–900, January 2007.

[15] M Newton, John Levine, and Maria Fox. Genetically evolved macro-actions in ai planning problems. *Proceedings of the 24th UK Planning and Scheduling SIG*, pages 163–172, 2005.

[16] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[17] Balaraman Ravindran and Andrew Barto. An algebraic approach to abstraction in reinforcement learning. In *Twelfth Yale Workshop on Adaptive and Learning Systems*, pages 109–144, 2003.

[18] Benjamin Rosman and Subramanian Ramamoorthy. What good are actions? accelerating learning using learned action priors. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–6. IEEE, 2012.

[19] A.A. Sherstov and P. Stone. Improving action selection in mdp's via knowledge transfer. In *Proceedings of the 20th national conference on Artificial Intelligence*, pages 1024–1029. AAAI Press, 2005.

[20] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.