Planning with Affordances

Abstract

Current methods for exactly solving decision-making under uncertainty require exhaustive enumeration of all possible states and actions, leading to exponential run times, leading to the well-known "curse of dimensionality." Approaches to address this problem by providing the system with formally encoded knowledge such as options or macro-actions still fail to prevent the system from considering many actions which seem obviously irrelevant for a human partner. To address this issue, we introduce a novel approach to representing knowledge about how to plan in terms of *affordances* [1]. Our affordance formalism and associated planning framework allows an agent to efficiently prune its action space based on domain knowledge. 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 on several planning and building tasks, showing a significant increase in speed and reduction in state-space exploration compared to subgoal planning, options, and macro-actions.

1 INTRODUCTION

As robots move out of the lab and into the real world, planning algorithms need to be able to scale to domains of increased noise, size, and complexity. A classic formalization of this issue is the sequential decision making problem, where increases in problem size and complexity directly correspond to an explosion in the state-action space. Current approaches to solving sequential decision making problems cannot tackle these problems as the state-action space becomes large [**?**].

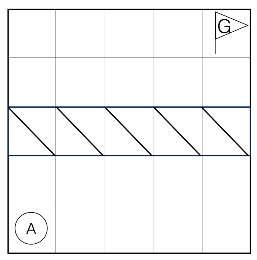
There is a strong need for a generalizable form of knowledge that, when coupled with a planner, is capable of solving problems in these massive domains. Humans provide an excellent existence proof for such planning, as we are capable of searching over an immense number of possible actions when presented with a goal. One approach to explaining how humans solve this planning problem is by focusing on problem-specific aspects of the environment which focus the search toward the most relevant and useful parts of the state-action space. Formally, Gibson [1] proposed to define this intuition as an *affordance*, "what [the environment] offers [an] animal, what [the environment] provides or furnishes, either for good or ill." Additionally, roboticists have recently become interested in leveraging affordances for perception and prediction of humans [3, 4]. In this paper we will formalize the notion of an affordance as a piece of planning knowledge provided to an agent operating in a Markov Decision Process (MDP) [2]. We demonstrate that, like an option or macro-action, an affordance provides additional information to the agent, enabling more efficient planning. However, unlike previous approaches, an affordance enables more significant speedups by reducing the size and branching-factor of the search space, enabling an agent to focus its search on the most relevant part of the problem at hand. This approach means that a *single* set of affordances provides general domain knowledge, becoming relevant just when the agent reasons that it needs to pursue a particular subgoal. Furthermore, Affordances are not specific to a particular state-space nor problem-type, and thus, provide the agent with transferrable knowledge that is effective in a wide variety of domains and problems, unlike other approaches.

2 BACKGROUND

2.1 OOMDP

The Object Oriented Markov Decision Process (OO-MDP) is an extension of the classic Markov Decision

BRIDGEWORLD



$$\Delta_1 := < onPlane, reachGoal > \Longrightarrow \alpha = \{\leftrightarrow\updownarrow\}$$
$$\Delta_2 := < nearTrench, reachGoal > \Longrightarrow \alpha = \{\leftrightarrow\updownarrow, \square\}$$
$$\Delta_3 := < nearWall, reachGoal > \Longrightarrow \alpha = \{\leftrightarrow\updownarrow, \boxtimes\}$$

Affordance Knowledge Base for BRIDGEWORLD

Figure 1: Figure demonstrating the intuition of the approach. Maybe a minecraft scene annotated with the affordances used to build a bridge?

Process (MDP). Recall that a finite 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 $s' \in \mathcal{S}$, and $\mathcal{R}(s, a)$ denotes the reward at s when action a is applied, and γ is a discount factor. The OO-MDP changes the underlying representation of the state space S.

2.2 SUBGOALS

Subgoal planning leverages the intuition that certain goals in planning domains may only be brought about if certain preconditions are first satisfied. For instance, in the Minecraft domain, one must be in possession of grain in order to bake bread. In Branavan et. al, they explore learning subgoals and applying them in order to plan through a variety of problems in Minecraft.

Formally, in subgoal planning, the agent is given a pair of predicates:

$$< x_k, x_l >$$

where x_l is the effect of some action sequence performed on a state in which x_k is true. Thus, subgoal planning requires that we perform high-level planning in subgoal space, and low-level planning to get from subgoal to subgoal.

Running Example

Problem 1: Loss of generality One important thing to note about subgoals that *are* general enough to enhance an agent's planning abilities in a wide variety of state spaces is that they propose *necessary* claims about the domain that the agent occupies. If the subgoals are *contingent* (i.e. true in some state spaces of the domain but not in others), then they can be shown to completely lose their scalability. For instance, consider the task in `BRIDGEWORLD`, in which the agent must place a block in the trench that separates the agent from the goal. The subgoal $< blockInTrench(), agentAtGoal() >$ might be a perfectly useful goal in *this* planning scenario, but an adversary could easily come up with thousands of worlds in which such a subgoal would completely derail the agent's planner. In order for subgoals to be useful, they must be necessary claims about the domain, otherwise, one can always come up with a counter world (by definition of necessary).

Problem 2: Granular Planning However, this poses a serious problem for subgoal planning, as the vast majority of tasks in mine craft are not so easily broken into useful, necessary subgoals. Movement for instance is particularly difficult, as the subgoals for movement must be necessary claims (i.e. $< agentOneAwayFromGoal(), agentAtGoal() >$), otherwise, they can be broken. Unfortunately, coming up with such subgoals is not an easy task, and often the best we can do is to plan at such a low level that we lose any benefit of planning over subgoals to begin with

2.3 OPTIONS

The options framework proposes incorporating high-level policies to 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. An option o is defined as follows:

$o \ = < \pi_0, I_0, \beta_0 >$, where:

$$\pi_0 : (s, a) \rightarrow [0, 1]$$
$$I_0 : s \rightarrow \{0, 1\}$$
$$\beta_0 : s \rightarrow [0, 1]$$

Here, π_0 represents the *option policy*, I_0 represents a precondition, under which the option policy may initiate, and β_0 represent the post condition, which determines which states terminate the execution of the option policy.

As Konidaris and Barto point out, the classic options framework is not generalizable, as it does not enable an agent to transfer knowledge from one state space to another. Recently, Konidaris and Barto's expand on the classic options framework and allow for a more portable implementation of options. Still, though, planning with options requires either that we plan in a mixed space of actions *and* options (which blows up the size of the search space), or requires that we plan entirely in the space of options. Additionally, providing an agent with an option policy is a difficult task

$$\boxed{\begin{aligned} \leftrightarrow\!\!\!\!\!\!\updownarrow \; &= \{\uparrow, \leftarrow, \downarrow, \rightarrow\} \\ \square &= \{\uparrow \square, \leftarrow \square, \downarrow \square, \rightarrow \square\} \\ \boxtimes &= \{\uparrow \boxtimes, \leftarrow \boxtimes, \downarrow \boxtimes, \rightarrow \boxtimes\} \\ \\ \mathcal{A} &= \{\leftrightarrow\!\!\!\!\!\!\updownarrow, \square, \boxtimes\} \end{aligned}}$$

Figure 2: The agent's set of actions in the Minecraft domain ($\square$ is a placement action, $\boxtimes$ is a destruction action)

for a human designer (especially if we want an optimal policy, which we do).

2.4 MACROACTIONS

Running Example

3 AFFORDANCES

Formally, an Affordance is defined as
$Aff = < p, g > \longrightarrow \alpha$, where:

$$\alpha \subseteq \mathcal{A}$$

$$p : s \longrightarrow \{0, 1\}$$

$$g : s \longrightarrow \{0, 1\}$$

Where α is a subset of the agent's given set of actions $\mathcal{A}$, p is a *precondition* that is a predicate over states, and g is a *goal* or *subgoal* that is also a predicate over states.

The constituents that make up an Affordance parallel those of the other planning approaches discussed in the background section.

Running Example

The Affordance formalism introduced above and expanded on in this paper resolves the weaknesses of these other frameworks by limiting the complexity of the seed knowledge required of the designer, providing enough knowledge to limit the search space, and still maintains scalability.

We should be able to prove that given a "good" set of subgoals the agent will be able to reach one after the other with high probability. Therefore, in the case that the agent cannot reach a subgoal there is likely a better one. The agent should then prompt for a more specific subgoal that will better allow it to reach the next one.

	Affordances	Options	Subgoals	VI
T1	1	1	0	0
T2	1	1	0	1
T3	1	1	1	0
T4	1	1	0	0

Figure 3: Results of running each algorithm N times on a variety of planning tasks (see Appendix I for list of tasks). Units are in wall-clock time.

4 EXPERIMENTS

Task List, apply each planning system (Aff, O, SG) to all tasks.

5 RESULTS

We ran each planning algorithm on 5 distinct tasks, each with its own state space and goal-type. Options, Subgoals, and Affordances were all given optimal knowledge, in each scenario.

6 CONCLUSION

References

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

[2] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2), 1998.

[3] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Robotics: Science and Systems (RSS)*, 2013.

[4] Hema S Koppula and Ashutosh Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *International Conference on Machine Learning (ICML)*, 2013.