Learning Dirichlet Priors for Affordance Aware Planning

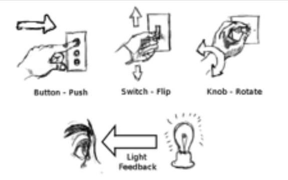
David Abel & Gabriel Barth-Maron, James MacGlashan, Stefanie Tellex
Dept. of Computer Science, Brown University

Goal

Enable autonomous agents to learn how to plan efficiently in massive stochastic state spaces.

Background

Affordances: Direct agent toward relevant action possibilities.



"What [the environment] offers [an] animal, what [the environment] provides or furnishes, either for good or ill"

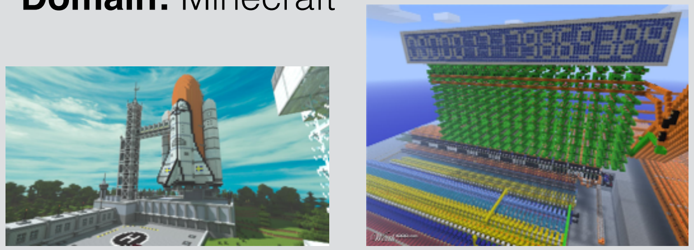
- J.J. Gibson, 1977

Formalism:

$$\Delta = \langle p, g \rangle \longmapsto \mathcal{A}'$$

p = predicate on states
g = lifted goal description
$\mathcal{A}'$ = subset of MDP Actions

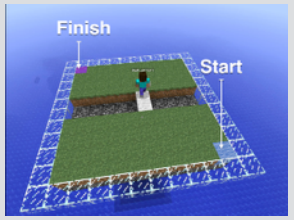
Domain: Minecraft



≈ *Turing Complete Legos*

Affordance Example

$\Delta_1 = \langle nearPlane, atLoc \rangle \longmapsto \{move\}$
$\Delta_2 = \langle nearTrench, atLoc \rangle \longmapsto \{place\}$



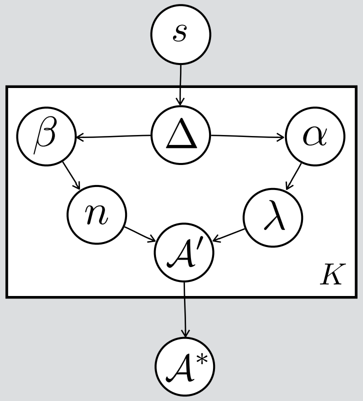
If Δ's predicate is true and Δ's goal type matches the current goal, use Δ's actions

Learning

Goal: For a given state, for each affordance, learn which actions are most relevant:

$$\Pr(\mathcal{A}^* \mid s, \Delta_1 \ldots \Delta_K)$$

Graphical Model:



s = OO-MDP State
Δ = Affordance
α = Action Counts
β = Action Set Size Counts
λ = Distribution on Actions
n = Distribution on Action Set Size
$\mathcal{A}'$ = *One* Affordance's Action Set
$\mathcal{A}^* = \bigcup_{i=1}^{K} \mathcal{A}'_i$

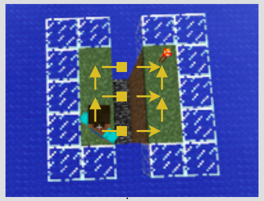
Where:

$$\Pr(\lambda \mid \alpha) = DirMult(\alpha)$$
$$\Pr(n \mid \beta) = Dir(\beta)$$

Learning Example

1) For each activated affordance, count:

→ = *move*
■ = *place*



$\alpha =$ number of worlds in which each action was used
$\beta =$ number of unique actions used in each world

$\Delta_1 = \langle ✓\ nearTrench, ✓\ atGoal \rangle$

$\Delta_1.\alpha.moveRight{+}{+},\ \Delta_1.\alpha.moveForward{+}{+},\ \Delta_1.\alpha.placeRight{+}{+}$
$\Delta_1.\beta.3{+}{+}$

2) When solving the MDP on a new state space, in each state *s*:

$$\mathcal{A}^* = \bigcup_{i=1}^{K} (\Delta_i.getActions(s))$$

3) Where

$\Delta_i.getActions(s)$:

```
λ ← DirMult(Δ_i.α)
n ← Dir(Δ_i.β)
A' ←_n λ
return:  A'
```

Results

Avg. # Bellman Updates Per Converged Policy