6.882
Embodied Intelligence

Phillip Isola
Leslie Pack Kaelbling
Tomas Lozano-Perez

Classic Papers

- Shakey
  - Domain: Partially observable, stochastic, sequential, continuous, single agent
  - Agent: goal-based agent
- Brooks
  - Domain: Partially observable, stochastic, sequential, continuous, (mostly) single agent
  - Agent: simple reflex, reflex with state
- Sims (Evolution process)
  - Domain: Observable, deterministic?, episodic, continuous?, single agent
  - Agent: utility-based
- Sims (Evolved agents)
  - Domain: Partially observable, stochastic, sequential, continuous, (mostly) single agent
  - Agent: simple reflex, reflex with state
- Dyna
  - Domain: (World-state is observable, Dynamics is not), stochastic, sequential, discrete, single agent
  - Agent: utility-based

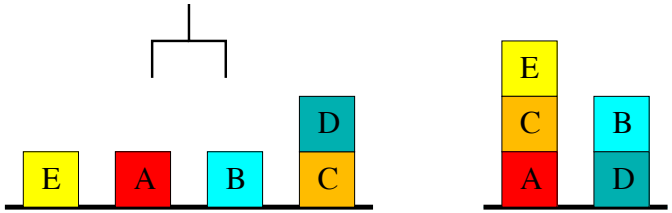
STRIPS planning model

A planning problem in Strips is represented by a tuple $P = \langle A, O, I, G \rangle$ where A is a set of atoms, O is a set of operators, and $I \subseteq A$ and $G \subseteq A$ encode the initial and goal situations. The operators $op \in O$ are all assumed grounded (i.e., with the variables replaced by constants). Each operator has a precondition, add, and delete lists denoted as $Prec(op)$, $Add(op)$, and $Del(op)$ respectively. They are all given by sets of atoms from A. A Strips problem $P = \langle A, O, I, G \rangle$ defines a state space $\mathcal{S}_P = \langle S, s_0, S_G, A(\cdot), f, c \rangle$ where

(S1) the states $s \in S$ are collections of atoms from A;
(S2) the initial state s_0 is I;
(S3) the goal states $s \in S_G$ are such that $G \subseteq s$;
(S4) the actions $a \in A(s)$ are the operators $op \in O$ such that $Prec(op) \subseteq s$;
(S5) the transition function f maps states s into states $s' = s - Del(a) + Add(a)$ for $a \in A(s)$;
(S6) all action costs $c(a, s)$ are 1.

STRIPS domain

(Oh no it's) The Blocksworld



- Facts: $on(x,y)$, $onTable(x)$, $clear(x)$, $holding(x)$, $armEmpty()$.
- Initial state: $\{onTable(E), clear(E), \ldots, onTable(C), on(D,C), clear(D), armEmpty()\}$.
- Goal: $\{on(E,C), on(C,A), on(B,D)\}$.
- Actions: $stack(x,y)$, $unstack(x,y)$, $putdown(x)$, $pickup(x)$.
- $stack(x,y)$? $pre: \{holding(x), clear(y)\}$
  $add: \{on(x,y), armEmpty()\}$
  $del: \{holding(x), clear(y)\}$.

Previous approaches to planning in STRIPS-like problems

(descriptions due to Jorg Hoffman)

- Logical theorem proving
  - From planning task description, generate formula in first-order logic that is satisfiable iff there exists a plan; use a theorem prover on that formula.
- Search in the space of plans (partial order planning)
  - Starting at goal, extend partially ordered set of actions by inserting achievers for open sub-goals, or by adding ordering constraints to avoid conflicts.
- Graphplan
  - In a forward phase, build a layered "planning graph" whose "time steps" capture which pairs of actions can achieve which pairs of facts; in a backward phase, search this graph starting at goals and excluding options proved to not be feasible.
- SATPlan
  - From planning task description, generate propositional CNF formula that is satisfiable iff there exists a plan with k steps; use a SAT solver on such formulas for different values of k

Planning as Heuristic Search: Bonet & Geffner

- Brute-force graph search, e.g. breadth-first, is hopeless in large state spaces. We need some leverage.
- Heuristics are one important form of leverage, but they may be difficult to invent.
- **Key contribution of this paper** – "delete relaxation" heuristics. A domain-independent strategy for deriving heuristics for planning in the STRIPS framework. Exploits the factored structure of the domain and rules.
  - h_max – roughly number of actions needed to add the "most difficult" goal fact (in the relaxed problem)
  - h_add – roughly the sum of the number of actions needed to add all the goal facts (in the relaxed problem)
- h_FF (by Hoffman) – size of plan to achieve the goals (in the relaxed problem)

Discussion

- When might you want to use a STRIPS-style planner as opposed to value iteration?
- Assume we represent state as some vector of (uninterpreted) numbers and we have a simulator s' = f(s,a) and a goal test g(s). What is the difference in how much information is "baked in" vs a STRIPS representation?
- Imagine that we want to use a STRIPS style planner to plan for a real blocks world – piles of blocks on a real 3d table with a real robot. Assume perfect perception/control for now. What challenges would you face?
- What might be other approaches to "discovering" heuristic information?

Complexity of Planning (from Helmert, Nebel)

Summary

- Planning using general first-order terms is undecidable.
- Planning using a function free language is EXPSPACE-complete.
- Planning with a propositional language (no schema variables) is PSPACE-complete.
- If we consider only "short" plans, the complexity comes down to NP-completeness.
- Domain-dependent planning can be easier.
- For Logistics, the existence problem is in P, while the optimization problem is NP-complete, which holds for many other domains as well.

Width and Serialization of Classical Planning Problems: Lipovetzky & Geffer

- Key contributions of this paper:
  - A novel notion of the complexity of a planning problem: width
  - A simple algorithm, IW(k), that runs in time exponential in width (k)
  - An empirical exploration that shows that many standard planning benchmarks have very low width
  - Combining width-based pruning with heuristics
- Discussion
  - Does IW(k) need a heuristic? What if we don't have STRIPS-style action descriptions, just a factored state and goal representation, and a simulator s'=f(s,a)?

Planning with Pixels in (Almost) Real Time: Bandres et al.

- Key contributions of this paper:
  - Planning in the space of screen features using IW(1)
  - Rollout IW(1) for good “anytime” behavior
- Discussion
  - Why does this work?
  - Compare and contrast with DQN
  - What if we used more meaningful object-based features?
  - Should we try to learn action descriptions instead?