Intelligent Agents

Heuristic Search Planning

Ute Schmid

Cognitive Systems, Applied Computer Science, Bamberg University

last change: June 29, 2010

Planning as Search

- Planning is an abstract search problem

Abstract-search(u)

- if *Terminal(u)* then *return(u)*
- $u \leftarrow Refine(u)$
- $B \leftarrow Branch(u)$
- $C \leftarrow Prune(B)$
- if $C = \emptyset$ then *return(failure)*
- (nondeterministically) choose $v \in C$
- *return(Abstract-search(v))*

STRIPS Planning as Search

- In state-based planning (STRIPS) u is a sequence of actions
  - Every solution reachable from u contains this sequence as prefix (forward search) or as suffix (backward search)
  - u is a partial plan

Graphplan Planning as Search

- In Graphplan u is a subgraph of a planning graph, that is a sequence of *sets of* actions together with constraints (for preconditions, effects, mutex)
  - Each solution reachable from u contains actions in u corresponding to the solved levels and at least one action from each level of the subgraph has not yet been solved in u.
  - Not every action in u will appear in the solution plan (several actions may achieve a goal but perhaps only one of them will be needed in the solution plan)

Abstract-Search

- Refinement: Modifying the collection of actions or constraints in u
- Branch: generating children of u
- Prune: removing some nodes which seem unpromising for search (e.g. an already visited node, or a domain-specific reason)
- Choose: instead of nondemerministic selection often depth-first search with some *Select*-function is used

Selection of Nodes

- *Select* is realized using a heuristic function!
- Heuristic: ranking nodes in order of their relative desirability
- in $A*$: hand-crafted, in planning automatically derived from planning problem
- $\hookrightarrow$ design principle: relaxation
- Since heuristics: no guarantee to be the best choice

Relaxation

- $Select(C) = argmin\{h(u) | u \in C\}$
- Relaxation: simplifying assumptions, relaxing constraints
- Obtaining h by solving the relaxed problem
- The closer the relaxed problem is to the real one
  - the better is the heuristic
  - the more effort it takes to calculate the heuristics
- For search for optimal solutions: admissible heuristics necessary

State Reachability Relaxation

- Asses how close an action may bring us to the goal
- $Res(s, a) = s \setminus DEL(a) \cup ADD(a)$ if $PRE(a) \subseteq s$
- Relaxation: neglect $DEL(a)$
- Simplified $Res(s, a)$: monotonic increase in number of propositions from s to $Res(s, a)$

- Let $s \in S$ be a state, p a proposition, and g a set of propositions
- The minimum distance from s to p, $\Delta^*(s, p)$ is the minimum number of actions to reach from s a state containing p.
- The minimum distance from s to g, $\Delta^*(s, g)$ is the minimum number of actions to reach from s a state containing all propositions g.

Ignoring DEL-Effects

- Estimate Δ_0: ignoring DEL, estimate distance to g as sum of the distances to all propositions in g
- $\Delta_0(s, p) = 0$ if $p \in s$
- $\Delta_0(s, p) = \infty$ if $\forall a \in A, p \notin ADD(a)$
- $\Delta_0(s, g) = 0$ if $g \subseteq s$
- otherwise
  - $\Delta_0(s, p) = min_a\{1 + \Delta_0(s, PRE(a)) | p \in ADD(a)\}$
  - $\Delta_0(s, g) = \sum_{p \in g} \Delta_0(s, p)$

Heuristic function: $h_0(s) = \Delta_0(s, g)$ (where g is the set of top-level goals)

Computing the Heuristic

Delta(s)

- for each p do: if $p \in s$ then $\Delta_0(s, p) \leftarrow 0$ else $\Delta_0(s, p) \leftarrow \infty$
- $U \leftarrow \{s\}$
- iterate
  - for each a such that $\exists u \in U$ with $PRE(a) \subseteq u$ do
    - ★ $U \leftarrow \{u\} \cup ADD(a)$
    - ★ for each $p \in ADD(a)$ do
      $\Delta_0(s, p) \leftarrow min\{\Delta_0(s, p), 1 + \sum_{q \in PRE(a)} \Delta_0(s, q)\}$
- until no change occurs in the updates

Computing the Heuristic

- Computes a value for each p
- Similar to minimum-distance (single-source) graph-search algorithm
- Starting from s_0 it procedes through each action whose preconditions are reached, until a fixed point is reached
- Action selection: $a \leftarrow argmin\{\Delta_0(Res(s, a), g)\}$
- Algorithm is polynomial in the number of propositions and actions
- For actions with different costs: replace 1 by the cost value of a
- Realized in the planner HSP (Geffner and colleagues)

Admissibility

- Heuristic h_0 is not admissible
- Example: $PRE(a) \in s_0$, $ADD(a) = g$, $s_0 \cap g = \emptyset$
  true distance to goal is 1
  $\Delta_0(s_0, g) = \sum_{p \in g} \Delta_0(s_0, p) = |g|$
- Modification of Δ_0: instead of sum of the distances of the elements of g, take maximum of the distances
- Problem: not as informative as Δ_0 (considering only a single goal proposition)
- Further modifications: look at the maximum of reaching k-tuples of propositions of g

Planning in Real-World Domains

- Incomplete Information
  - Conformant planning: Creat plans that work for all cases
  - Conditional planning: sense world during execution and decide which branch of the plan to follow
- Incorrect Information
  - Execution monitoring: check for unsatisfied preconditions
  - Re-planning
- Continuous planning: create new goals during acting in real time
- Multiagent planning

Including Knowledge

Using knowledge about the structure of the domain

- Hierarchical Planning (decomposition rules)
  cf. problem solving with AND-OR trees
- Domain axioms
- Domain specific search strategies

$\hookrightarrow$ larger plans become feasible (necessary for many real world problems, e.g. Mars Mobile)

Alternative to knowledge engineering: Learning of planning strategies!

Further Topics

- Interleaving plan construction and plan execution
- Plan revision
- Planning with temporal/resource constraints
- Non-deterministic planning
- ...