
Planning, Execution & Learning: Heuristic Search Planning

Reid Simmons

Heuristic Search Planning

- Basic Idea
 - *Automatically Analyze Domain/Problems to Derive Heuristic Estimates to Guide Search*
- Decisions
 - How to evaluate search states
 - How to use the evaluations to guide search
 - How to choose which part of plan to work on next
- *Resurgence in Total-Order, State-Space Planners*
 - Best such planner (FF) dominates other types
 - Still a hot topic for research

Search Heuristics

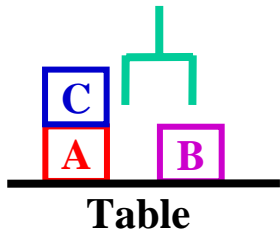
- Admissible
 - *What?*
 - *Why Important?*
- Informed
 - *What?*
 - *Why Important?*

Evaluating Search States

- Basic Idea
 - *Solve a **Relaxed** Form of the Problem;
Use as Estimate for Original Problem*
- Approaches
 - Assume **complete** subgoal independence
 - Assume no **negative** interactions
 - Assume **limited** negative interactions

HSP (Bonet & Geffner, 1997)

- Heuristic State-Space Planner
 - Can do either *progression* or *regression*
- Relax Problem by Eliminating “Delete” Lists
 - Essentially compute transitive closure of actions, starting at initial state
 - “Reachability” analysis
 - Cost of literal is stage/level at which first appears
 - Continue until no new literals are added
 - Similar to *GraphPlan’s* forward search



Computing Costs of Literals

⁰On(C, A) ⁰On(A, Table) ⁰On(B, Table) ⁰Handempty ⁰Clear(C) ⁰Clear(B)

Pick(C, A)

PickT(B)

¹Holding(C) ¹Holding(B) ¹Clear(A)

PutT(C) Put(C, A) Put(C, B) PutT(B) Put(B, A) Put(B, C) PickT(A)

²On(C, Table) ²On(C, B) ²On(B, A) ²On(B, C) ²Holding(A)

PickT(C) Pick(C, B) Pick(B, A) Pick(B, C) Put(A, B) Put(A, C)

³On(A, B) ³On(A, C)

On(A, B) & On(B, C) **Estimate:** 5 **Actually:** 6

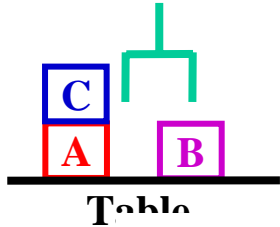
On(A, C) & On(C, B) **Estimate:** 5 **Actually:** 4

HSP Heuristics

- **Max**
 - Cost of action is *maximum* over costs of preconditions
 - Admissible, but not very informed
- **Sum**
 - Cost of action is *sum* of precondition costs
 - Informed, but not admissible
- **H²**
 - Solve for *pairs* of literals
 - Take maximum cost over all pairs
 - Informed, and claimed to be admissible

FF (Hoffmann, 2000)

- FF (Fast Forward) Refines HSP Heuristic
- Takes *positive* interactions into account
 - Avoids double-counting of actions
- Similar to *GraphPlan's* forward search combined with a *relaxed* version of its backward search
 - Ignores negative interactions
- Admissible and Informed



FF State Evaluation Heuristic

On(A, C) On(A, Table) On(B, Table) Handempty Clear(C) Clear(B)

Pick(C, A) **PickT(B)**

On(C, A) On(A, Table) On(B, Table) Handempty Clear(C) Clear(B)

Holding(C) Holding(B) Clear(A)

Pick(C, A) **PickT(B)**

PutT(C) **Put(C, A)** **Put(C, B)** **PutT(B)** **Put(B, A)** **Put(B, C)** **PickT(A)**

On(C, A) On(A, Table) On(B, Table) Handempty Clear(C) Clear(B)

Holding(C) Holding(B) Clear(A)

On(C, Table) **On(C, B)** On(B, A) On(B, C) **Holding(A)**

Pick(C,A) **PickT(B)**

PutT(C) **Put(C, A)** **Put(C, B)** **PutT(B)** **Put(B, A)** **Put(B, C)** **PickT(A)**

PickT(C) **Pick(C, B)** **Pick(B, A)** **Pick(B, C)** **Put(A, B)** **Put(A, C)**

On(C, A) On(A, Table) On(B, Table) Handempty Clear(C) Clear(B)

Holding(C) Holding(B) Clear(A)

On(C, Table) **On(C, B)** On(B, A) On(B, C) Holding(A) On(A, B) **On(A, C)**

On(A, C) & On(C, B)

Discussion

- Progression: Need to Calculate Heuristic Every Step
- Regression: Just Calculate Heuristic Once
- Heuristic Search Using Progression Generally More Robust
- HSP and FF Heuristics Outperform Partial-Order Planners
 - Ground actions seem to be the big difference
 - Easier to estimate cost without variables
 - Forward search provides reachability analysis
- Similar Techniques Applicable to Partial-Order Planners
 - REPOP (Nguyen & Kambhampati, 2001)
 - VHPOP (Younes & Simmons, 2001)

Heuristic Search Strategies

- **Best-First**
- **A***
- **Weighted A***
 - $H(s) = \text{cost-so-far}(s) + W * \text{estimated-cost}(s)$
 - Not admissible, but tends to perform much better than A*
- **Hill-Climbing**
 - Rationale: Heuristics tend to be better discriminators amongst local alternatives than as global (absolute) estimate
 - Random “restarts” when stuck

“Enforced” Hill Climbing

- Used to Avoid “*Wandering*” on “Plateaus” or in Local Minima
 - Perform breadth-first search until find *some* descendant state whose heuristic value is less than the current state
- Shown to be Very Effective
 - Especially when search space is pruned to eliminate actions that are “unlikely” to lead to goal achievement
- Used by FF

Flaw Selection

- Answers the Question:
 - *Which subgoal (or threat) should be worked on next?*
- **LIFO**
 - + Empirically, continuing to work on a given subproblem, all else being equal, tends to perform well (“coherence”)
 - Uninformed
- **Least-Cost**
 - Use heuristic estimate of subgoal cost to choose “easiest”
 - + Prefers forced choices, which reduces branching factor
 - Can be myopic

Flaw Selection

- **Delay Separable Threats**
 - Put off handling potential threats that may be solved by adding binding constraints
 - + Other choices often force separation to occur naturally (or converts to non-separable threat)
 - If separation is forced, could lead to earlier detection of dead end
- **Forced Choice**
 - Choose flaws for which only one possible choice exists (better: “at most one”)
 - + Adds no additional branching
 - May delay the inevitable dead-end detection

VHPOP (Younes)

- Plan Selection Uses A*
 - “Additive” heuristic estimates number of actions to achieve a condition
 - Similar to FF’s heuristic, but for partial-order plans
 - With positive interactions (admissible) or without
 - “Estimated effort” (used as tie-breaker)
- New Flaw Selection Strategies
 - “Static First”
 - “Most Cost First”
 - “Least Cost First”
 - “Local Flaw Selection”
 - combines heuristic and LIFO