Planning as Heuristic Forward Search

Brian C. Williams
March 31, 2004
16.412J/6.834J
Slides with help from:
Prof. Maria Fox

Readings in Planning as Forward Heuristic Search

Outline

- Introduction to FF
- FF Search Algorithm
- FF Heuristic Fn
- FF Example
- Appendix: HSP

Example: Polish

Move from room x to room y
- pre: robot is in x, door open
- add: robot is in y
- del: robot is in x

Open door
- pre: door is closed
- add: door is open
- del: door is closed

Close door
- pre: door is open
- add: door is closed
- del: door is open

Polish from room x to room y
- pre: door open
- add: floor polished

Initial State
- ln(A)
- Closed

Final State
- ln(B)
- Closed
- Polished
Planning as Forward Heuristic Search

- Planning can be seen as a state space search, for a path from the initial state to a goal state.
- Planning research has largely not been concerned with finding optimal solutions.
  - Although heuristic preference to shorter plans.
- Planning research has largely used incomplete or uninformed search methods.
  - Breadth first search
  - Meta search rules

The size of most state spaces requires informative heuristics to guide the search.

Review: Search Strategies

- Breadth first search (Uninformed)
  - systematic search of state space in layers.
- A* search (Informed)
  - Expands search node with best estimated cost.
  - Estimated cost = cost-so-far + optimistic-cost-to-go
- Greedy search
  - Expands search node closest to the goal according to a heuristic function.
- Hill-climbing search
  - Move towards goal by random selection from the best children.

To apply informed search to planning need heuristic fn
Fast Forward (FF)

- Forward-chaining heuristic search planner
- Basic principle: Hill-climb through the space of problem states, starting at the initial state.
  - Each child state results from apply a single plan operator.
  - Always moves to the first child state found that is closer to the goal.
  - Records the operators applied along the path.

The operators leading to the goal constitute a plan.

Outline

- Introduction to FF
- FF Search Algorithm
- FF Heuristic Fn
- FF Example
- Appendix: HSP
Planning Problem and State Space

- A planning problem is a tuple \( <P, A, I, G> \):
  - Propositions \( P \)
  - Ground actions \( A \) are instantiated operators
  - Initial state \( I \) is a subset of \( P \), and
  - Goal state \( G \) is a subset of \( P \).
- The state space of a problem consists of all subsets of propositions \( P \).
- A transition between two states is any valid application of an action, that is, its preconditions are satisfied.

FF Search Strategy

FF uses a strategy called enforced hill-climbing:
- Obtain heuristic estimate of the value of the current state.
- Find action(s) transitioning to a better state.
- Move to the better state.
- Append actions to plan head.
- \( \Rightarrow \) Never backtrack over any choice.
Maximize Utility $h$

$\text{h(S1)} < \text{h(S4)} < \text{h(init)} < \text{h(S2)} < \text{h(S3)} < \text{h(S5)} = \text{h(S6)}$

Plan Head: A, B

Finding a better state: Plateaus

$\text{h(S7)} < \text{h(S6)} = \text{h(S8)} \ldots = \text{h(S10)} < \text{h(S11)} < \text{h(S12)}$

Perform breadth first search from the current state, to states reachable by action applications, stopping as soon as a strictly better one is found.
Enforced Hill-Climbing (cont.)

- The success of this strategy depends on how informative the heuristic is.
  - FF uses a heuristic found to be informative in a large class of benchmark planning domains.
- The strategy is not complete.
  - Never backtracking means that some parts of the search space are lost.
- If FF fails to find a solution using this strategy it switches to standard best-first search.
  - (e.g., Greedy or A* search).

Outline

- Introduction to FF
- FF Search Algorithm
- FF Heuristic Fn
- FF Example
- Appendix: HSP
**FF’s Heuristic Estimate**

- The value of a state is a measure of how close it is to a goal state.
- This cannot be determined exactly (too hard), but can be approximated.
- One way of approximating is to solve a relaxed problem.
  - Relaxation is achieved by ignoring the negative effects of the actions.
  - The relaxed action set, $A'$, is defined by:
    \[
    A' = \{<\text{pre}(a), \text{add}(a), 0> | a \text{ in } A\}
    \]

**Distance Estimate Extracted From A Relaxed Plan Graph**

- Current: $\text{In}(A)$, Closed
- Goal: $\text{In}(B)$

- Layers correspond to successive time points,
- # layers indicate minimum time to achieve goals.
**Building the Relaxed Plan Graph**

- Start at the initial state.
- Repeatedly apply all relaxed actions whose preconditions are satisfied.
  - Assert their (positive) effects in the next layer.
- If all actions are applied and the goals are not all present in the final graph layer, **Then** the problem is unsolvable.

**Distance Estimate Extracted From A Relaxed Plan Graph**

- Current: In(A), Closed  
  Goal: In(B)

- Layers correspond to successive time points,  
- # layers indicate minimum time to achieve goals.
Extracting a Relaxed Soln

- When a layer containing all of the goals is reached, FF searches *backwards* for a plan.
- The first possible achiever found is always used to achieve each goal.
  - This maximizes the possibility for exploiting actions in the relaxed plan.
- The relaxed plan might contain many actions happening concurrently at a layer.
- The number of actions in the relaxed plan is an estimate of the true cost of achieving the goals.

Distance Estimate Extracted From A Relaxed Plan Graph

- Current: In(A), Closed
- Goal: In(B)

- Layers correspond to successive time points,
- # layers indicate minimum time to achieve goals.
How FF Uses the Heuristic

- FF uses the heuristic to estimate how close each state is to a goal state
  - any state satisfying the goal propositions.

- The actions in the relaxed plan are used as a guide to which actions to explore when extending the plan.
  - All actions in the relaxed plan at the 1st layer that achieve at least one of the (sub)goals required at the 2nd layer are considered helpful.
  - FF restricts attention to the helpful actions when searching forward from a state.

Distance Estimate Extracted From A Relaxed Plan Graph

- Current: In(A), Closed
- Goal: In(B)

Useful actions: Open
Properties of the Heuristic

- The relaxed plan that is extracted is not guaranteed to be the optimal relaxed plan.
  - the heuristic is not admissible.
    - FF can produce non-optimal solutions.

Getting Out of Deadends

- Because FF does not backtrack, FF can get stuck in dead-ends.
  - This arises when an action cannot be reversed, thus, having entered a bad state there is no way to improve.

- When no search progress can be made, FF switches to Best First Search from the initial state.
  - Detecting a dead-end can be expensive if the plateau is large.
Outline

- Introduction to FF
- FF Search Algorithm
- FF Heuristic Fn
- FF Example
- Appendix: HSP

Example: Polish

**Move** from room x to room y
- pre: robot is in x, door open
- add: robot is in y
- del: robot is in x

**Open** door
- pre: door is closed
- add: door is open
- del: door is closed

**Close** door
- pre: door is open
- add: door is closed
- del: door is open

**Polish** from room x to room y
- pre: door open
- add: floor polished

**Initial State**
- In(A)
- Closed

**Final State**
- In(B)
- Closed
- Polished
Relaxed Graph

Estimate: 3
- # actions in plan
- Useful Actions: Open
  - actions used in 1st layer, used to create children

Relaxed Graph

Estimate: 3 = Plateau
- perform BFS
- Useful Actions:
  - Move, Close, Polish
Goal: In(B), Closed, Polished

Estimate 3

In(A) Closed

Open

Node 2

In(A) Opened

Move(A,B)

Close

Polish

Node 3

In(B) Opened

Estimate ?

Relaxed Graph

In(B)

Opened

noop

Move(B,A)

Close

Polish

noop

Closed

Opened

Polished

In(B)

Opened

Estimate: 2 = Off Plateau

Useful Actions:

Close, Polish

Relaxed Graph

Estimate: 2

No improvement

In(B) Closed

Closed

Open

In(A) Opened

Move(A,B)

Close

Polish

In(B) Opened

Estimate 2

In(B)

Closed

Opened

Polished

noop

Move(B,A)

noop

Close

Polish

In(B)

Closed

Opened

Polished
Plan: Open, Move(A,B), Polish, Close

Relaxed Graph

Estimate: 0

Useful Actions:
Close

Goal: In(B), Closed, Polished

Node 1
In(A) Closed
Open

Node 2
In(A) Opened

Node 3
In(B) Opened
Close
Polish

Node 4
In(B) Closed
Estimate 2

Node 5
In(B) Opened Polished

In(B) Closed

In(B) Opened

Estimate ?
Fast Forward (FF)

- Forward-chaining heuristic search planner
- Basic principle: Hill-climb through the space of problem states, starting at the initial state.
  - Each child state applies a single plan operator.
  - Always moves to the first child state found that is closer to the goal.
  - Record the transitions applied along the path.
  - The transitions leading to the goal constitute a plan.

Outline

- Introduction to FF
- FF Search Algorithm
- FF Heuristic Fn
- FF Example
- Appendix: HSP
Other Distance Estimates

- Distance to the goal can be estimated without building a relaxed reachability analysis, and then extracting a relaxed plan.

- An alternative is to estimate the cost of achieving a goal, as the cost of achieving the preconditions of a suitable action, plus one.

- The cost of achieving any goal, not already true, can be initialized to infinity, and then updated in every state.

HSP

- Developed by Blai Bonet and Hector Geffner at Simon Bolivar University, Venezuela.
- Hill-climbing search based on random selection from the set of the best successor states.
- Heuristic evaluation - Add: estimate the relaxed distance to the goal set using the following equation:

  \[ g_s + (C) = \sum_{q \in C} g_s(q) \]

- That is: the cost of achieving a collection of atoms, C, from state s, is the sum of the cost of achieving each of those atoms from state s.
Search Strategy of HSP

- Starting at initial state, move forward by hill-climbing using the additive heuristic measure to evaluate states.
- Next state visited is one strictly closer to the goal - ties broken randomly.
- A threshold is used to limit time spent exploring plateaus and search is restarted if threshold is exceeded.

Properties of Heuristics

- The add heuristic $g_s(C)$ pessimistically assumes that the individual atoms have to be achieved independently.
- This means $g_s(C)$ is not admissible because it will often over-estimate distance to a goal.
- An alternative, admissible heuristic is:
  
  $$g_s(C) = \max(q \in C) g_s(q)$$

- This is admissible but uninformative
  - greatly under-estimates distances
  - little use in practice.
HSP Heuristic Estimate

- The cost of achieving a single atom from a state \( s \) is given by evaluating:

\[
g_s(q) = \min(g_s(q), 1 + g_s(Prec(op))
\]

- That is: the minimum of the currently known cost (if \( q \) is already asserted in \( s \)) and the cost of achieving the preconditions of an action, \( op \), that would assert \( q \) in the next state.

- \( g_s(q) \) gets initialized to infinity (or 0 if it is true in the initial state) and is updated in every state.

Differences between FF and HSP

- FF uses a heuristic evaluation based on the number of actions in an explicit relaxed plan.
- HSP uses weight values which approximate (but over-estimate) the length of a relaxed plan.
- FF uses the relaxed plan to prune action choices.
- FF terminates BFS for a successor state as soon as an improvement is found. HSP selects successors randomly from the set of best states.
- FF defaults to complete BFS from initial state if the enforced hill-climbing strategy fails.
Conclusions

- FF and HSP are forward chaining planners that use a hill-climbing strategy based on relaxed distance estimates.
- FF uses a number of pruning heuristics that can be very powerful (especially in the simply-structured propositional benchmark domains).
- HSP has been extended in a number of ways to exploit alternative search strategies.
- HSP is better suited to some problems than FF, but FF has overall better performance.