Artificial Intelligence

Search-Based Planning

Russell and Norvig - Chapter 11

see also

Planning as Heuristic Search, Bonet and Geffner

http://www.ldc.usb.ve/~hector/reports/hsp-aij.ps

AIPS-98 Planning Competition

Round	Planner	Av. Time	Solved	Shortest
Round 1	BLACKBOX	1.49	63	55
	HSP	**35.48**	**82**	**61**
	IPP	7.40	63	49
	STAN	55.41	64	47
Round 2	BLACKBOX	2.46	8	6
	HSP	**25.87**	**9**	**5**
	IPP	17.37	11	8
	STAN	1.33	7	4

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

initial state	goal state
At(Home)	At(Home)
Sells(HWS, Drill)	Have(Drill)
Sells(SM, Milk)	Have(Milk)
Sells(SM, Bananas)	Have(Bananas)

operators

Go(here, there)
Precond: At(here)
Effect: At(there) AND NOT At(here)

Buy(x, store)
Precond: At(store) and Sells(store,x)
Effect: Have(x)

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

- Step 1: Instantiate all Operators (can delete all unnecessary ones)

Go(Home, HWS)
Go(Home, SM)
Go(SM, HWS)
Go(SM, Home)
Go(HWS, Home)
Go(HWS, SM)
Buy(Milk, SM)
Buy(Bananas, SM)
Buy(Drill, HWS)

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

- Step 2: Relax the Planning Problem by Removing all Negative Predicates from the Effects List

Go(here, there)
Precond: At(here)
Effect: At(there) ~~AND NOT At(here)~~
Buy(x, store)
Precond: At(store) and Sells(store,x)
Effect: Have(x)

Unfortunately, the planning problem remains NP hard.

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

- Step 3: Simplify the Planning Problem Further by Not Taking Interactions of the Positive Predicates from the Effects List into Account

Consider the original planning problem.
Calculate the heuristic of state s as h_s(goal), with:

h_s(set of predicates s') := $\max_{\text{predicate p in s'}} h_s(s')$. [estimated # of actions to achieve all predicates in s' from s]

h_s(predicate p) :=

0	if p in s
$1 + \min_{\text{operator o with p in add list}} h_s(\text{precondition list of o})$	otherwise

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

For our example:

h_{start}(goal) = max(h_{start}(At(Home)),
h_{start}(Have(Drill)),
h_{start}(Have(Milk)),
h_{start}(Have(Bananas)))
= max (0, 2, 2, 2) = 2.

h_{start}(At(Home)) = 0.

h_{start}(Have(Drill)) = 1 + h_{start}({At(HWS), Sells(HWS, Drill)}) = 1+1 = 2.

h_{start}({At(HWS), Sells(HWS, Drill)}) =
max(h_{start}(At(HWS)), h_{start}(Sells(HWS, Drill))) = max(1,0) = 1.

h_{start}(At(HWS)) = 1 + min(h_{start}({At(Home)}), h_{start}({At(SM)})) =
1 + min(0, ...) = 1.

and so on...

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

- Step 4: Give up Admissibility to Get the Estimated F-Values Closer to the Correct Ones

Trick 1: When calculating the heuristic values, use
h_s(set of predicates s') := $\sum_{\text{predicate p in s'}} h_s(s')$.

Trick 2: During the A* search, use
f(state) = g(state) + 5 h(state).
This results in a mix of Greedy BFS and A*.

Planning as Search

perform a regular A* search from the start state

the only problem is to come up with a good heuristic function

For our example:

h_{start}(goal) = h_{start}(At(Home)) +
h_{start}(Have(Drill)) +
h_{start}(Have(Milk)) +
h_{start}(Have(Bananas))
= 0 + 2 + 2 + 2 = 6.

h_{start}(At(Home)) = 0.

h_{start}(Have(Drill)) = 1 + h_{start}({At(HWS), Sells(HWS, Drill)}) = 1+1 = 2.

h_{start}({At(HWS), Sells(HWS, Drill)}) =
h_{start}(At(HWS)) + h_{start}(Sells(HWS, Drill)) = 1 + 0 = 1.

h_{start}(At(HWS)) = 1 + min(h_{start}({At(Home)}), h_{start}({At(SM)})) =
1 + min(0, ...) = 1.

and so on...