Algorithms for Planning as State-Space Search

Section 10.2

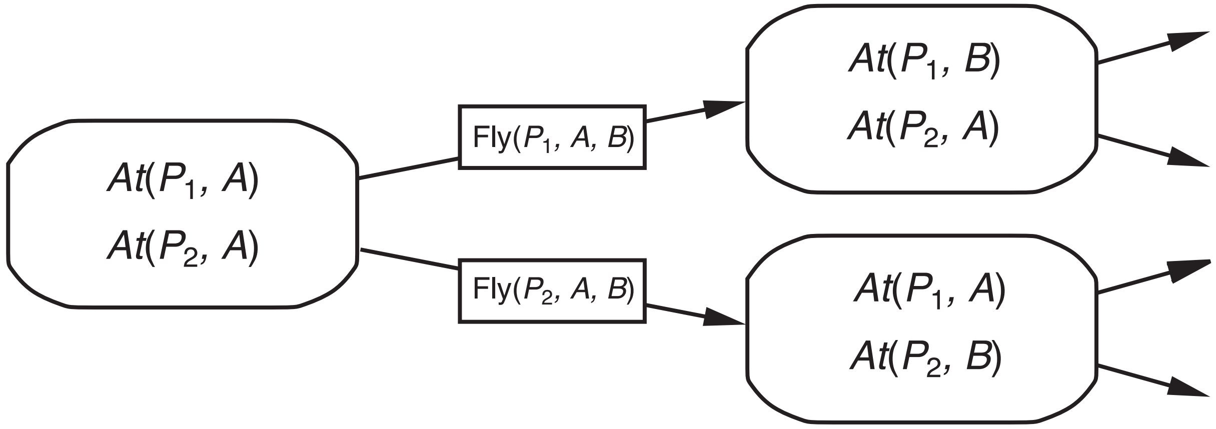
Outline

- Forward (progression) state-space search
- Backward (regression) relevant-states search
- The Fast Forward (FF) system

Additional references used for the slides:
Hoffmann, Jörg (2001). FF: The Fast-Forward Planning System. *AI Magazine*, 22(3), 57-62.
Yoon, Soon; **Fern**, Alan; **Givan**, Robert (2008). Learning Control Knowledge for Forward Search Planning. *Journal of Machine Learning Research*, 9, 683-718.

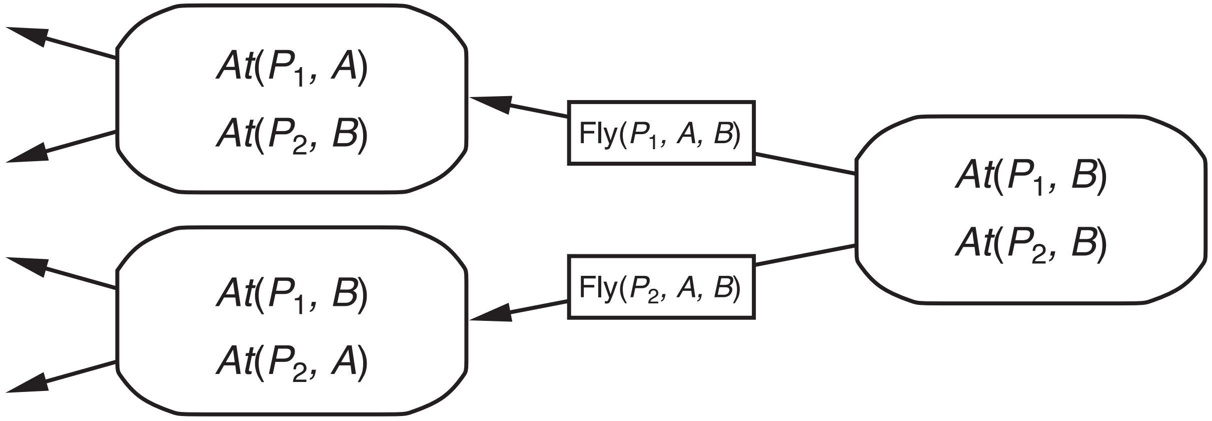
Forward vs. backward search

(a)



$At(P_1, A)$ $At(P_2, A)$
$Fly(P_1, A, B)$
$Fly(P_2, A, B)$
$At(P_1, B)$ $At(P_2, A)$
$At(P_1, A)$ $At(P_2, B)$

(b)



$At(P_1, A)$ $At(P_2, B)$
$At(P_1, B)$ $At(P_2, A)$
$Fly(P_1, A, B)$
$Fly(P_2, A, B)$
$At(P_1, B)$ $At(P_2, B)$

Forward search

- Works similar to regular search: start with the initial state, expand the graph by computing successors
- The successors are computed by using the applicable actions and finding the resulting states

Properties of forward search for planning

- There will be a lot of *irrelevant actions*, i.e., actions that will not contribute to the final plan
- The state space is large: e.g., air cargo problem with 10 airports, and 5 planes and 20 pieces of cargo at each airport:
  at each state there is a mimimum of 450 actions (when all packages are at airports with no planes, each of the 50 planes can fly to one of the 9 airports) and a maximum of 10,450 actions (when all packages and planes are at the same airport, each one of the 200 package can be loaded to one of the 50 planes, or each of the 50 planes can fly to one of the 9 airports.

Backward search

- It is a search in the reverse direction: start with the goal state, expand the graph by computing parents
- The parents are computed by *regressing actions*: given a ground goal description g and a ground action a, the regression from g over a is g': $g' = (g - \textsc{Add}\ (a)) \cup \textsc{Precond}\ (a)$.
- The regression represents the effects that
    - don’t have to be true in the previous step because they were added
    - have to be true in the previous step because they are the preconditions of the action

Properties of backward search for planning

- *Irrelevant actions* will be less of an issue because we are starting with the goal.
- The branching factor is low but regression gives a set of states rather than a single state. Thus, it is hard to develop heuristics (the situation is similar to partial order planners).

The Fast-Forward (FF) planning system

- Heuristic method: use *relaxed Graphplan*
- Search method: *enforced hill climbing*
- Ordering successors: *helpful actions*

Relaxed planning graph

- Ignore (remove) the delete lists of the actions.
- The first fact layer is identical to the starting state.
- The action layers contain the applicable actions.
- Expand the graph until a layer contains all the goals.
- Note that the graph will not contain any mutexes because the delete lists were removed.

Extracting a relaxed plan

- Start at the top graph layer m, work on all the goals.
- At each layer i,
  if a goal is present in layer $i-1$, then insert it to the goals to be achieved in layer $i-1$,
  else, select an action in layer $i-1$ that adds the goal, and insert the action's preconditions into the goals at $i-1$.
- Once all the goals at level i are worked on, continue with the goals at level $i-1$. Stop at the first level.
- The relaxed plan is a sequence of action sets:
  $< O_0, O_1, \ldots, O_{m-1} >$.
- Note that this is a backtrack-free procedure.

Computing the heuristic

- The estimated solution length from a state S is:

$$h_{FF}(S) := \sum_{i=0,\ldots,m-1} |O_i|$$

- This heuristic is computed in polynomial time.
- Note that this is an admissible heuristic because the preconditions and the goals are defined in terms of positive state facts, and it is easier to achieve the goal when the delete lists are removed.

Enforced hill climbing

- In standard hill climbing used by the HSP planner, a best successor to each state is chosen randomly, and restarts take place when a path becomes too long.
- FF evaluates all the successors, then
  - If no successor has a better heuristic value, performs a breadth-first search for a state with a strictly better evaluation
  - The path to the new state is added to the current plan, and the search continues from this state
- FF's method performs well because plateaus and local minima tend to be small in many benchmark planning problems

Helpful actions

- Restrict any state's successors to those generated by the first action set in its relaxed solution.
- For a state S, the set $H(S)$ of helpful actions is defined as

$$H(S) := \{o | pre(o) \subseteq S, add(o) \cap G_1 \neq \emptyset\}$$

- G_1 denotes the set of goals at the next level.

Performance evaluation

- Eight experiments were conducted by turning the three features of FF on or off.
- “Turning a feature off” yields HSP’s techniques (HSP: Heuristic Search Planner)
- The test suite included 20 domains where one alternative leads to significantly better performance than the other one

Experimental results

Distance Estimate	Hill Climbing				Enforced Hill Climbing			
	All Actions		**Helpful Actions**		**All Actions**		**Helpful Actions**	
	Time	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*	*Length*
HSP distance	2	2	1	2	2	0	1	0
FF distance	12	2	12	5	11	9	9	11

Search Strategy	All actions				Helpful Actions			
	HSP distance		**FF distance**		**HSP distance**		**FF distance**	
	Time	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*	*Length*
Hill Climbing	5	1	9	1	3	2	1	2
Enforced HC	9	8	8	10	16	6	16	9

Pruning Strategy	Hill Climbing				Enforced Hill Climbing			
	HSP distance		**FF distance**		**HSP distance**		**FF distance**	
	Time	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*	*Length*
All Actions	2	0	3	0	2	1	2	0
Helpful Actions	13	7	14	8	15	5	15	3

Performance evaluation

- FF's estimates improve run-time performance in about half of the domains across all switch alignments
- With enforced hill climbing in the background, FF's estimates have clear advantages in terms of solution length
- Enforced hill climbing often finds shorter plans because when its enters a plateau, it performs a complete search for an exit and adds the shortest path to this exit ot its current plan prefix.
- Helpful actions strategy performs better in domains where a significant number of actions can be cut. Solutions are shorter.

Hoffmann’s comments

- The simple structure of the benchmarks is the reason behind FF’s success
- FF was outperformed in problems using random SAT instances. The other planners (IPP and Blackbox) did better because they can rule out many partial truth assignments early.