Nicholas Beshouri
AIND project 3 research review

Development 1: Stanford Research Institute Problem Solver (STRIPS)

Very early planners tried to define both the state of the world and the actions that could cause that state to change purely in terms of formal logic. While some of these systems were successful, they struggled with the frame problem, and could only solve trivial problems (Fikes & Nilsson, 1971). STRIPS retained a logic based state representation for the states themselves, but defined actions as "operators" that could only add or remove fluents from the current state. Any fluent not in an action's add or delete list is assumed not to have changed, side stepping the frame problem.

STRIPS was limited in the problems that it could solve, and its actual planning algorithm (an early form of backward chaining that checked states using a theorem prover) was relatively inefficient, but it simplified the problem of planning enough for progress to be made (Fikes & Nilsson, 1993). The formal language it used to encode problems was also highly influential and provided a foundation for the Planning Domain Definition Language (PDDL), which is still used today (Russell & Norvig 2010).

Development 2: GraphPlan

By the mid 70's the focus of the field had shifted to partial order planners, which break problems up into groups of actions that can be completed in parallel. These dominated the field for the next two decades, culminating in the development of GraphPlan in 1997 (Russell & Norvig, 2010).

GraphPlan works in two phases. First, it constructs a Planning Graph, which is a sort of linearized approximation of how a world state might evolve over time. The first level of the graph, S_0, is just the set of fluents in the initial state of the problem. S_0 is followed by A_0, which contains all the actions whose preconditions are met in S_0 as well as "no-op" actions that carry the fluents in S_0 forward in time unchanged. S_1 contains all the fluents that are the results of actions in A_0, and so on. As each level of the graph is added, any pairs of mutually exclusive elements are recorded. This continues until a level S_n is found that contains all the goal fluents and doesn't mark any of them mutually exclusive (Blum & First, 1997).

In the second phase, the algorithm backtracks from S_n, searching A_{n-1} for a set of non-mutex actions that could have produced the goal states in S_n. This continues recursively for each pair of levels until the algorithm either reaches the initial state, meaning it's found a valid group of actions to perform at each time step, or fails to, in which case another pair of action and state levels are added to the planning graph and the process starts over.

One advantage of this approach is that, if there are lots of independent actions in a problem, GraphPlan can group them together in a few action levels and solve a large problem quickly using a short graph. It also guarantees an optimal plan. However, GraphPlan can perform poorly when the actions in a domain aren't independent (forcing longer graphs and deeper searches) or when the mutex relationships of nodes don't capture the important constraints of a problem

(Blum & First, 1997). Despite these limitations, GraphPlan was orders of magnitude faster than other planners of the time and reenergized the field (Russell & Norvig, 2010). At the AIPS-98 planning competition, held a year after GraphPlan's introduction, four of the five contestants used it in some form (Long, 2000).

Development 3: Heuristic Search Planner (HSP)

HSP, the fifth contestant at AIPS-98, was the first planner to demonstrate that forward state space search could be a competitive strategy in domain independent planning (Russell & Norvig, 2010). HSP works by conducting a greedy hill climbing search of the state space (Bonet & Geffner, 2001). To assign a value of a successor state, HSP generates a heuristic that estimates the cost of solving a relaxed version of the problem from that state. While this approach has disadvantages (incomplete, sub-optimal, can't find parallel sub-plans), its greedy bias is actually an advantage when speed is more important than optimality. At the AIPS-98 competition, HSP found solutions to more problems than any other planner. An improved version, HSP2, solved some of HSP's problems by replacing hill climbing with best-first search. HSP2 maintained HSP's greedy bias by weighting the heuristic component of each node's cost by 5.0 (a weight of 1.0 being equivalent to A* search), but generally found shorter plans than HSP.

The lasting influence of HSP is the idea of automatically generating heuristics from PDDL's factored problem representation, which others continued to improve. By the time of the AIPs-'00 competition (won by Fast-Forward, another non-optimal total order planner inspired by HSP) partial order planners like GraphPlan had ceased to be competitive (Russell & Norvig, 2010).

References

Blum, A. L., & Furst, M. L. (1997). Fast planning through planning graph analysis. Artificial Intelligence, 90(1-2), 281-300.

Bacchus, F. (2001). AIPS'00 Planning Competition. AI Magazine, 22(3).

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. Artificial Intelligence, 129(1-2), 5-33.

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3-4), 189-208.

Fikes, R. E., & Nilsson, N. J. (1993). STRIPS, a retrospective. Artificial Intelligence, 59(1-2), 227-232.

Long, D. (2000). The AIPS-98 Planning Competition. AI Magazine, 21(2).

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach (3rd ed.). Upper Saddle River: Prentice-Hall.