Historical developments in the field of AI planning and search

"The planning problem in Artificial Intelligence is about the decision making performed by intelligent creatures like robots, humans, or computer programs when trying to achieve some goal. It involves choosing a sequence of actions that will (with a high likelihood) transform the state of the world, step by step, so that it will satisfy the goal." [2] "AI planning arose from investigations into state-space search, theorem proving, and control theory and from the practical needs of robotics, scheduling, and other domains." [1]

The Standford Research Institute Problem Solver (STRIPS) is an automated planning system developed by Stanford Research Institute in 1971 by Fikes, Richard E., and Nils J. Nilsson. It was the first major planning system and it was designed as the planning component of the software for the Shakey robot project at SRI. Its overall control structure was modeled on that of a a state-space search system called the General Problem Solver created in 1959 by Herbert A. Simon, J.C. Shaw, and Allen Newell. [1]

In STRIPS, the state variables have the domain {0,1}, and an action consists of three sets of state variables, the PRECONDITION, the ADD list, and the DELETE list (it is assumed that ADD and DELETE don't intersect.)

In STRIPS planning, a goal is usually expressed as a set of state variables. A state is a goal state if all of the goals have the value 1 in it. All the assignments are instantaneous and take place simultaneously. It is possible to execute an action in a state if all the variables in PRECONDITION have the value 1. [2]

The STRIPS language has some limitations as it cannot express:

•Hierarchical Plans

•Complex Conditions

•Time

•Resources [3]

The representation language used by STRIPS has been far more influential than its algorithmic approach.

The Action Description Language, or ADL, relaxed some of the STRIPS restrictions and made it possible to encode more realistic problems. Then, the Problem Domain Description Language, or PDDL, was introduced as a computer-parsable, standardized syntax for representing planning problems. [1]

The differences of PDDL in comparison to STRIPS are:

•The PRECONDITION may be an arbitrary Boolean combination of atomic facts about the state variables.

- •Instead of the unconditional assignments represented by ADD and DELETE, the effects may be conditional. This means that the effects are of the form, IF condition THEN a := v where the condition is a Boolean combination of facts.
- •Goals may be Boolean combinations of atomic facts (formulas).

STRIPS corresponds to PDDL with trivial conditions that are always true. [3]

Planners in the early 1970s generally considered totally ordered action sequences. Problem decomposition was achieved by computing a subplan for each subgoal and then stringing the subplans together in some order. This approach, called linear planning by Sacerdoti (1975), was soon discovered to be incomplete. [1]

A complete planner must allow for interleaving of actions from different subplans within a single sequence. The idea of a partial-order planner is to have a partial ordering between actions and only commit to an ordering between actions when forced.[2]

A partial ordering is a less-than relation that is transitive and asymmetric. A partial-order plan is a set of actions together with a partial ordering, representing a "before" relation on actions, such that any total ordering of the actions, consistent with the partial ordering, will solve the goal from the initial state. [4]

The ideas underlying partial-order planning include the detection of conflicts and the protection of achieved conditions from interference. The construction of partially ordered plans (then called task networks) was pioneered by the NOAH planner and by the NONLIN system. [1]

Explicit state-space search, meaning the generation of states reachable from the initial state one by one, is the earliest and most straightforward method for solving some of the most important problems about transition systems including planning.

All the current main techniques for it were fully developed by 1990s, including symmetry and partial order methods, informed search algorithms, and optimal search algorithms. It is relatively easy to implement efficiently, but, when the number of states is high, its applicability is limited by the necessity to do the search only one state at a time. However, when the number of states is less than some tens of millions, this approach is efficient and can give guarantees of finding solutions in a limited amount of time. [2]

The resurgence of interest in state-space planning was pioneered by Drew McDermott’s UNPOP program (1996), which was the first to suggest the ignore-delete-list heuristic. Then Bonet and Geffner’s Heuristic Search Planner (HSP) and its later derivatives were the first to make state-space search practical for large planning problems.[1]

Many transition systems have too many states to consider them explicitly one by one, and then factored representations that allow representing large numbers of states and state sequences may be a more efficient alternative. Such representations and search methods are known as symbolic or factored. The currently most scalable method (since the late 1990s) is based on reduction to the propositional satisfiability problem SAT. [2]

Constraint-based approaches such as GRAPHPLAN and SATPLAN are best for NP-hard domains, while search-based approaches do better in domains where feasible solutions can be found without backtracking. GRAPHPLAN and SATPLAN have trouble in domains with many objects because that means they must create many actions. In some cases the problem can be delayed or avoided by generating the propositionalized actions dynamically, only as needed, rather than instantiating them all before the search begins. [1]

Avrim Blum and Merrick Furst (1995, 1997) revitalized the field of planning with their GRAPHPLAN system, which was orders of magnitude faster than the partial-order planners of the time. [1]

GRAPHPLAN is a general-purpose planner for STRIPS style domains, based on ideas used in graph algorithms. GRAPHPLAN uses a novel planning graph, to reduce the amount of search needed to find the solution from straightforward exploration of the state space graph. In this approach, both nodes are actions and atomic facts, arranged into alternate levels. This way, a planning graph makes explicit many of the constraints inherent in the problem to reduce the amount of search needed [5].

Most recently, there has been interest in the representation of plans as binary decision diagrams, compact data structures for Boolean expressions widely studied in the hardware verification community. There are techniques for proving properties of binary decision diagrams, including the property of being a solution to a planning problem.[1]

References

1.Norvig, P. and Russel, S. (2010). Artificial Intelligence: A Modern Approach. Third Edition.
2.Rintanen, J. (2010) A brief overview of AI planning. Recovered from https://users.ics.aalto.fi/rintanen/planning.html
3.Regli, W. Planning. Recovered from http://edge.cs.drexel.edu/regli/AI/slides/mcs380-590_planning.PDF
4.Poole, D. and Mackworth, (2010) A. Artificial Intelligence, Partial Order planning. Recovered from http://artint.info/html/ArtInt_209.html
5.Blum, A. and Furst, M. (1997) "Fast planning through planning graph analysis". Artificial intelligence. 90:281-300.