AI Planning & Search: Historical Developments

Rohit Mehra
contactrohitmehra@gmail.com

I. Introduction: What was the need of developing AI Planning & Search?

Traditional search algorithms searched for a plan to the goal in the state search space and then executed it. This approach had a major drawback of not using the observations from environment on the go or in other words requirement to map all the real states with the actions before hand. Hence, a new approach which would interweave planning and executing was required. This new approach deals with finding a solution in a plan search space, where instead of dealing with real world states, agent deals with the hypothetical scenarios (belief states) of taking actions, observing the effects and drawing conclusions. In traditional search, the states were represented by real values (provided by humans), and everything regarding the problem was defined around these values. But with AI planning, we require agents to deal with logic as base and logical information as knowledge, hence a requirement of representing this in a language, that agents could use to "understand" and "solve" the problem. Also, the techniques to solve these problems and find a plan to achieve the goal had to be developed, though most revolved around traditional search, the were required to perform better.

II. Languages: How AI Planning languages evolved?

STRIPS (STanford Research Institute Problem Solver), was the planning component in software used to control Shakey, a robot developed at the Stanford Research Institute (SRI). What became more popular than Shakey itself was the representational language used by STRIPS. It used the concepts of states, goals and a set of actions to represent problems. These same concepts became a common ground to develop upon, for other representational languages such as ADL and PDDL.

- **Action Description Language (ADL):** It extended STRIPS by removing some of its constraints to represent more realistic problems. ADL does not follow the same assumption of unmentioned literals being False. It supports negative literals in general with quantified variables, conditional effects and disjunction logic in goals, which were not allowed in STRIPS.
- **Planning Domain Definition Language (PDDL):** With an aim to standardize planning languages PDDL was developed, with both STRIPS and ADL as its valuable predecessors. In other words PDDL contains STRIPS and ADL, as well as many more other representational languages.

III. Techniques: How did AI planning techniques evolved from traditional search techniques?

Searching is the core technique used in planning. As mentioned earlier, AI search is done over plan search space and these plans are represented using the planning languages mentioned above. The three main techniques wrapped around search algorithms are as follows:

- **Linear Planning:** Using this technique, plan for each subgoal was searched separately and then these actions were concatenated to complete the task. It is able to do its job efficiently if goals do not interact much, but there was no interleaving of goal and achievement. LP was shown to be incomplete and a better technique called goal-regression planning [Waldinger, 1975] was introduced. Here, the main idea was to reorder actions so that the plan for subgoals don’t conflict with each other whenever possible to do so.
- **Non-Linear or Partial-Order Planning:** Basic idea here was to use the goal set instead of goal stack (used in Linear Planning) and to include all possible subgoal orderings in the search space, which handled goal interactions by interleaving. This approach is sound, complete and may be optimal with respect to the plan length found.
- **Graphplan Algorithm:** Research on AI planning was concentrated on non-linear or partial-order planning algorithms until the introduction of the Graphplan algorithm in 1995 by Blum and Furst(1997). This algorithm had two characteristics that separated it from earlier ones: it finds plans of a fixed length (that is incrementally increased until a plan is found), and it uses reachability information for pruning the search tree. These differences brought the performance of Graphplan to a level not seen in connection with earlier planners.[Rintanen and Hoffmann]