# Planning Algorithms in AI

Duc Vu

**1. The Universal Multi-agent Obdd-based Planne (UMOP)** :

UMOP [Jensen Veloso,2000] is a universal planning system for multi-Agent and non-Deterministic domains. As input UMOP takes a planning problem described in the Non-deterministic Agent Domain Language (NADL). The NADL problem is translated to a Kripke structure search problem. The Kripke structure is represented by a conjunctive partitioned transition relation encoded symbolically as a set of Ordered Binary Decision Diagrams (OBDDs, Bryant 1986).

All of these algorithms iteratively compute a BDD representing a universal plan during a breadth-first search backward from the goal states to the initial states. The search algorithms rely on efficient techniques developed in symbolic model checking for computing components of the plan and searching the state space. The set of states is the possible initial states, while is the goal states. In each iteration, a pre-component of the plan is generated from the current set of covered states

**2. Graphplan** :

Graphplan [Blum and Furst, 1995] is based on Planning Graph Analysis which explicitly constructs a compact structure called a Planning Graph. A Planning Graph encodes the planning problem in such a way that many useful constraints inherent in the problem become explicitly available to reduce the amount of search needed. Furthermore, Planning Graphs can be built in polynomial time and provide a quite substantial improvement in running time.

Planning Graphs are closer in spirit to the Problem Space Graphs (PSGs) of Etzioni [1990], though unlike PSGs, Planning Graphs are based not only on domain information, but also the goals and initial conditions of a problem and an explicit notion of time. Planning Graphs offer a means of organizing and maintaining search information that is reminiscent of the efficient solutions to Dynamic Programming problems. Planning Graph Analysis appears to have significant practical value in solving planning problems even though the inherent complexity of STRIPS planning, which is at least PSPACE-hard (e.g., see Bylander [1994]), is much greater than the complexity of standard Dynamic Programming problems.

The Graphplan planner uses the Planning Graph that it creates to guide its search for a plan. The search that it performs combines aspects of both total-order and partial-order planners. Like traditional total-order planners, Graphplan makes strong commitments in its search.

**3. Partial-order planning** :

Partial-order planning is an approach to automated planning that leaves decisions about the ordering of actions as open as possible . To better understand that, it might be helpful to know what planning is, and then what ordered planning entails. To that end, planning is ”the task of coming up with a sequence of actions that will achieve a goal” . There are several types of algorithms that allow us to construct a plan, such as regression planning and progression planning, which Norvig taught in the videos, and partial-order is also one of the kind. But what distinguishes partial-order planning from the other two - it is not totally-ordered as we see in progression and regression planning. Instead, partial-order planning enables us to ”take advantage of problem decomposition.” The algorithm ”works on several subgoals independently, solves them with several sub-plans, then combines the sub-plans”. In addition, he notes that, ”such an approach also has the advantage of flexibility in the order in which it constructs the plan. That is, the planner can work on ’obvious’ or ’important’ decisions first, rather than being forced to work on steps in chronological order.” In the 1980s and 90s, partial-order planning was seen as the best way to handle planning problems with independent subproblemsafter all, it was the only approach that explicitly represents independent branches of a plan. On the other hand, it has the disadvantage of not having an explicit representation of states in the state-transition model. That makes some computations cumbersome. By 2000, forward-search planners had developed excellent heuristics that allowed them to efficiently discover the independent subproblems that partial- order planning was designed for. As a result, partial-order planners are not competitive on fully automated classical planning problems.

## REFERENCES

[1] Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach 3rd Edition, P. 391.*

[2] R.M. Jensen and M. M. Veloso, *OBDD-based Deterministic Planning using the UMOP Planning Framework.* AIPS-00 Workshop on Model-Theoretic Approaches to Planning, 2000