Graphplan[1]

Graphplan is initialised with a layer consisting of the initial state S_0 of the problem and a layer with the possible actions A_0 from that state. It then recursively creates new layers of states and actions. The set of states S_i is composed of the union between S_{i-1} and the result of any of the possible actions in S_{i-1}. The set of actions is updated with the actions for which all preconditions are met. The key improvement of Graphplan is that it propagates mutually exclusive states and actions and therefore significantly reduces the search space (i.e. the graph expands monotonically). Mutually exclusive actions are such that no plan could contain both, while two propositions are mutually exclusive if no valid plan could make them both true. Mutually exclusive actions can be detected through *inference* (if either action deletes a precondition or adds an effect to the other), or if there exist mutually exclusive preconditions for two actions.

The forward expansion phase of the algorithm halts if Graphplan identifies a proposition state in which all goal propositions exist and are not mutually exclusive of one another. Then, it uses backward search in order to find a set of actions that reach the goal. For some goal set found at time t, it selects an action at t-1 that is not exclusive of any actions that have been selected and which achieves the goal. This is applied recursively until it reaches the initial state. In essence, the second part of Graphplan is a search algorithm that recursively looks at previous state layers (i.e. sub-goals to the goal) and then outputs the actions that form the path from the goal to the initial state. During the forward expansion phase, the algorithm checks that there is no solution of length t-1 before expanding to plans of length t and therefore it is guaranteed to reach an optimal solution.

Graphplan is guaranteed to terminate if no solution exists. The algorithm terminates if upon expansion from a set S_i to a set S_{i+1}, the subgoal set (i.e. the proposition level) remains constant. A proposition level S_n is the iterative union of $\forall S_i, 0 \leq i \leq n$. Because of this, it is guaranteed that if a proposition appears in some level, it will appear in all future levels. It follows that if $\exists$ some proposition level S_n such that $S_n = S_{n+1}$, then $S_n = S_m, \forall m > n$. Such a state is bound to happen because STRIPS operators produce only a finite set of propositions. Hence, if the solution does not exist at level n, there exists no solution to the plan and the algorithm terminates.

SATplan[2]

It is an algorithm that is in many ways similar to Graphplan. The underlying concept is that it converts planning problems into Boolean satisfiability problems. In other words, SATplan takes as input the initial state, the set of successor state axioms and the set of goal propositions and creates a logical proposition that is the concatenation under "AND" of the above. If this proposition is satisfiable then SATplan extracts the solution by checking which of the actions have been assigned "True" values. If not, the time parameter is incrementally increased up to an upper bound T_{max}. This time parameter can be thought of in the same way as the layer structuring in GraphPlan. T_0 represents

[1] Avrim L. Blum and Merrick L. Furst (1995), *Fast planning through planning graph analysis*
[2] Henry Kautz and Bart Selman (1992), *Planning as Satisfiability*

the initial state, T_1 is the state after one action, and so on. One of the differences between GraphPlan and SATplan is that SATplan requires additional input with regard to the upper bound to which it expands the search. In other words, GraphPlan terminated the expansion if the proposition subsets were identical for consecutive iterations, while SATplan expands up to an arbitrary bound T_{max}. However, it is possible to estimate the true diameter of the graph before expanding it, and this allows for a reasonable estimation of $T_{max.}$

BlackBox[3]

The Blackbox algorithm attempts to unify the different planning frameworks, and uses features of both Graphplan and SATplan. SATplan fully initiates a problem instance before passing it through a logic simplifier, while Graphplan mixes graph representation and simplification through computation of mutually exclusive actions and propositions. BlackBox computes mutexes just like Graphplan and then converts the graph to a CNF wff (i.e. a well-formed formula that is a made of an AND of ORs), and a SAT engine then solves the problem. In other words, BlackBox combines the mutex computation used in Graphplan with the conversion from a planning problem to an instance of Boolean satisfiability problem used in SATplan.

3 Henry Kautz and Bart Selman (1998), *Unifying SAT-based and Graph-based planning*