Research Review

Diego Serrano

For this review, I will focus on summarizing the recent advances of AI Planning. Particularly, I will focus on the STRIPS Representation, partial order planning and GraphPlan.

In AI Planning, the objective is to build algorithms that enable an agent to create a list of actions to achieve its goals. The following 3 fields have attracted attention as recent developments over the last 2, have revolutionized the field:

- The STRIPS Representation.
- Partial Order Planning.
- The GraphPlan planning algorithm.

These approaches have much in common and both are improved by recent progress in constraint satisfaction and search algorithms technology. With recent developments, the performance is impressive, as many planners are able to solve problems which are orders of magnitude harder that problems from years ago.

As an example, the *Blackbox Planner* (a planning system that works by converting problems specified in STRIPS notation into Boolean satisfiability problems, and then solving the problems with a variety of state-of-the-art satisfiability engines) requires only 6 minutes to find a 105 action logistics plan in a search space with 10^{16} possible states.

The STRIPS Representation

STRIPS (standing for Stanford Research Institute Problem Solver) is an alternative representation for the feature-based representation of actions of an AI problem that is action-centric. For each action, specifies the effect of the action.

Given the fact that features describe the states of the problem, a feature-based representation of actions is needed. Then, those features must be divided into primitive and derived features. Definite clauses are used to determine the value of derived features from the values of the primitive features in any given state.

The STRIPS representation for an action consists of:

- The precondition which is a set of assignments of values to features that must be true for the action to occur
- The effect which is a set of resulting assignments of values to those primitive features that change as the result of the action

Partial Order Planning

The objective of Partial Order Planning is to have partial ordering between actions and only commit to an ordering between actions when forced. 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 will solve the goal from the initial state.

$$act_0 < act_1 \; if \; action \; act_0 \; before \; action \; act_1 \; in \, partial \; order$$

This means that act_0 must occur before act_1.

The elements of a partial order plan would be:

- Start: an action that achieves the relations that are true in the initial state
- Finish: an action whose precondition is the goal to be solved
- Goal: an atom that must be true in the goal state
- P: Precondition on an action act_1 in a plan that will have an action act_0 associated such that act_0 occurs before action act_1
- A: Any other action that makes P false. Must either be before act_0 or after act_1

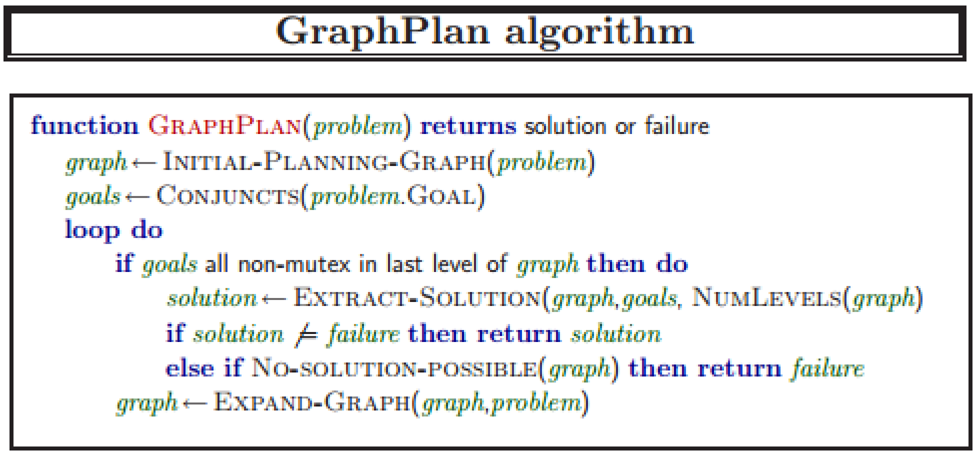
Informally, a partial order planning works as follows: begin with the actions *start* and *finish* and the partial order *start < finish.* The planner maintains an agenda that is a set of *{P, A*} pairs. Initially the agenda contains pairs *{Goal,finish}.*

At each stage in the planning process, a pair *{Goal,* act_1*}* is selected from the agenda, where *P* is a precondition for action act_1. Then, an action act_0 is chosen to achieve *P*. That action is either already in the plan, or is a new action that is added to the plan. An action act_0 must happen before act_1 in the partial order. It adds a causal link that records that act_0 achieves *P* for action act_1. Either action that deletes *P* must happen either before act_0 or after act_1. If act_0 is a new action, its preconditions are added to the agenda, and the process continues until the agenda is empty.

GraphPlan

GraphPlan is an algorithm based on planning graph. The algorithm takes as input a planning problem expressed in STRIPS and produces – if possible – a sequence of operations to reach a goal state. It uses a novel planning graph to reduce the amount of search needed to find the solution from straightforward exploration of the state pace graph.

Graphplan alternates between two phases: graph expansion and solution extraction. The graph expansion phase extends a panning graph forward in “time” until it has achieved a necessary (but insufficient) condition for plan existence. The solution extraction phase then performs a backward-chaining in the graph looking for a plan that solves the problem. If no solution is found, the cycle repeats by further expanding the planning graph.



GraphPlan algorithm

```
function GRAPHPLAN(problem) returns solution or failure
    graph ← INITIAL-PLANNING-GRAPH(problem)
    goals ← CONJUNCTS(problem.GOAL)
    loop do
        if goals all non-mutex in last level of graph then do
            solution ← EXTRACT-SOLUTION(graph, goals, NUMLEVELS(graph))
            if solution ≠ failure then return solution
            else if NO-SOLUTION-POSSIBLE(graph) then return failure
        graph ← EXPAND-GRAPH(graph, problem)
```

Figure 1 GraphPlan pseudocode taken from http://www.cs.nott.ac.uk/~psznza/G52PAS/lecture12.pdf

Sources

http://artint.info/html/ArtInt_209.html

http://homes.cs.washington.edu/~weld/papers/pi2.pdf

https://en.wikipedia.org/wiki/Graphplan

http://www.cs.nott.ac.uk/~psznza/G52PAS/lecture12.pdf

https://www.cs.rochester.edu/u/kautz/satplan/blackbox/