A short summary of key ideas in AI Planning

Felix Brei

April 18, 2017

1 Introduction

Automated planning and scheduling, or briefly called *Planning* [1], is one of the main applications of AI. Its goal is to create *plans* (sequences of actions)that lead from an initial state to a desired *goal* state upon execution.

Planning is a very complex task for real world problems and there are a lot of approaches that try to deal with this issue by trying to express complex actions as simple action schemas (PDDL) and stay at a high level, while others try to formalize the way humans think about planning (plan graphs knowledge representation).

In the following sections, I will give a brief explanation of three ideas that had a great impact on planning and have shown very good results in specific domains.

2 STRIPS

STRIPS, which is short for Stanford Research Institute Problem Solver, was the first influential problem solving algorithm [2] in the field of planning. It was developed by R. Fikes and N. Nilsson in 1971 [3] and consists of four different sets:

- P - a set of propositional variables to describe the domain
- S_0 - the initial state
- G - the goal state and
- O - the set of operations that can be applied in the given domain

The initial state and the goal state are basically propositional sentences. The operations on the other hand are a data structure themselves. They are usually described as a triplet $(pre(O), add(O), del(O))$, where $pre(O)$ is the set of preconditions (a sentence in the language of P) that have to be true to let O be applicable in a given situation, $add(O)$ is the set of variables that become true after applying O and $del(O)$ is the set of variables that become false.

The domain in which we want find a plan to get us from one situation to another, can be seen as a graph, where nodes represent states of the world and edges are operations that let us move between states of this world. Finding a plan becomes equivalent to finding a route from one node to another in said graph. This can be achieved in two ways: starting from the initial state (forward planning) is what most people would do intuitively. Or we can start from the goal state and consider actions that got us there. Then we continue this method with the goal's predecessors and so on (backwards planning). Both approaches have proven to be effective.

STRIPS does have its limitations though. It relies on a properly defined domain with named actions and known effects. It also has to rely on the world being deterministic and static, that is, all actions that have to be executed actually do get executed and there is no interference from other agents. These limitations have led to the development of another popular approach in planing, so called *partial order plans*.

3 Partial Order Planning

In 1975, Gerald Sussman proved that STRIPS struggles at solving problems with non-independent sub-goals. The so called Sussman anomaly consists of a simple arrangement of blocks on a table that have to be stacked [4]. This problem cannot be divided into sub goals that can be solved independently, which is why STRIPS and other so called *total order planners* fail at this task. Partial Order Planners on the other hand are able to solve this anomaly quickly. Their goal is to generate not an absolute sequence of actions, but rather a loosely ordered list of actions that have to be performed in order to reach a goal. For example, suppose you want to take the train to the next big city, but the ride is quite long and you want to buy something that you can eat during the ride. A total order planner would try to generate sequences of actions that fulfill each sub goal, for example: $buy(snack) \rightarrow buy(ticket) \rightarrow ride(train)$. This sequence does indeed fulfill the goal of reaching the destination without starving, but it is pretty much set in stone. A partial order planner could generate something like this: $\{buy(snack), buy(ticket)\} \rightarrow \{ride(train)\}$, which means that it does not matter if we buy the ticket first and then the snack or vice versa. It only matters that we do all the shopping first before boarding the train. This representation leaves more freedom in executing the actions. It is up to the agent to either serialize the actions if necessary, or maybe execute them in parallel if possible. It also allows us to quickly change the order of actions because of unknown influences without recalculating the whole plan.

Calculating plans in such a way follows the *Principle of Least Commitment* [5], meaning that we only constrain the order of actions as much as we really have to, leaving a lot of freedom to the agent to decide what to do next. This of course implies that the agent has to have the ressources to make quick decision on the fly because planning happens now only partially offline. This in turn results in a higher energy consumption and higher costs for robots want to execute such kind of plans.

4 GraphPlan

For a very long time researchers focussed on partial order planners. In 1997, Avrim L. Blum and Merrick L. Furst took a radically different approach and revolutionized the world of planning [6]. Instead of jumping right into the state space search like other planning algorithms did before, they first construct a data structure that is called Planning Graph. This graph consists of two different layers, each divided again into levels, that are connected in a certain way.

The first layer contains nodes with information about states, the second layer contains actions that rely upon and change these states. The first level inside the state layer consists of everything that is true in the initial state. The first level of the action layer consists of all actions that are possible, given the initial state. The second level of the state layer in turn contains all the information of the layer before, plus all effects of all actions from first action layer. This concept is repeated, until there are two identical levels. In addition to keeping track of all the actions that could be done and all propositions that could be true at a certain level, a planning graph also has to include links between nodes that state that these two nodes cannot be made true at the same time, or interfere with each other. These links are called *mutexes* (mutual exclusions). Keeping track of these mutexes has to be done directly after creating a level in the action or state level, the original paper describes thoroughly how to achieve this.

Because a planning graph generates a lot of (mutual exclusive) information that cannot be true at the same time, it is also often referred to as *belief state.*

The popularity of planning graphs has several reasons. At first, they are fast to calculate. But the main reason is their rich information content. A planning graph contains information about when a statement can be made true for the very first time and is therefore an excellent source for heuristics. Calculating the maximum or the sum of these levels for a given goal state gives a great estimate of how many actions are necessary to achieve all sub goals. By serializing the actions of a planning graph, it is also possible to quickly generate plans that achieve an arbitrary goal state within a reasonable time. If sub goals are independent, these plans are optimal. If not, they are at least a good estimate.

It is also easy to see if a goal can even be achieved at all. If any sub goal is not part of the final state level, than a given goal is impossible to achieve. This is because a planning graph also contains no-op actions whose sole purpose is to carry a state node from one level to another. This means, that as soon as a state node is part of a state level, it is automatically part of all following state levels.

5 Summary

This paper could only give a short introduction to some of the great ideas that people had while trying to solve the problem of planning in AI. A lot of these ideas were at first based on relaxed problems of real world examples, but also tried to improve upon past results. Much progress was made in the field of planning because people tried to find areas where state-of-the-art planners would fail and then tried to fix this by either improving the given algorithms or coming up with new ones. Algorithms like GraphPlan show how important it is to sometimes rethink the way we try to solve planning problems and have made it possible to solve even large problems today.

References

[1] Wikipedia *https://en.wikipedia.org/wiki/Automated_planning_and_scheduling*

[2] Artificial Intelligence, A Modern Approach *Peter Norvig, Stuart Russel*

[3] Wikipedia *https://en.wikipedia.org/wiki/STRIPS*

[4] Wikipedia *https://en.wikipedia.org/wiki/Sussman_Anomaly*

[5] An Introduction to Least Commitment Planning *Daniel S. Weld*

[6] Fast Planning Through Planning Graph Analysis *Avrim L. Blum, Merrick L. Furst*