# Reserch Review - Planning

**Frederick Chyan**

*Aug 17, 2017*

## Intractability

Planning is an important and foundamental problem in AI, because it covers two essential study **logic** and **search**. To phrase it in another way, *planning is about controlling combinatorial explosion*. Therefore, before diving into the history of development of various planning algorithm and solvers, a section on computational complexity is provided for reference. To begin with, P problems contains problems that can be solved in polynomial time and verify in polynominal time. NP problem contains problems that can be verified in polynomial time but unknown if a polynomial solution exist. Therefore P is a subset of NP. Examples of NP-Complete problems include set cover, set packing, 3SAT etc. Co-NP problem is the complementary of NP, i.e. when the "no" instance (complement of NP's decision problem) can be verified in polynomial time. **Planning problems** are even harder (at least as hard as) than NP-complete problems because a solution might not be verified in polynomial time. This is due to the plan itself might be up to 2^n long. There are algorithms, however, can solve planning problems by using only polynomial space by throwing away intermediate data (recomputation necessary), so planning problem belong in PSPACE. It can be shown all other problems in PSPACE are polynomial reducible to planning problem, making planning problem PSPACE-Hard.

## Hierarchical Task Network

Hierarchical task network was one of the early development in planning. The algorithm works as follow.[1]

```
1. Input a planning problem P.
2. If P contains only primitive tasks, then resolve the conflicts in P
   and return the result. If the conflicts cannot be resolved,
   return failure.
3. Choose a non-primitive task t in P.
4. Choose an expansion for t.
5. Replace t with the expansion.
6. Use critics to find the interactions among tasks in P,
   and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.
```

The intuition of the algorithm can be described in 4 steps. 1. Decompose tasks into subtasks 2. Handle constraints 3. Resolve interaction. 4. Backtrack and try other decompositions. Due to HTN's expressive syntax, its complexity might be harder than PSPACE because some problem might be undecidable. The HTN is closely related to partial-order planning such as UCPOP[2] which allows expressive syntax (ADL) and yet achieve sound and complete plan without restrictive syntax (STRIP) or linear planning.

## Planning Graph

Planning Graph was introduced as part of GRAPHPLAN algorithm in 1997 [3]. Planning graph works by creating alternating literal and action states recursively. When new possible literals are added, it makes new actions available (or not available) and vice versa. This repeats until the planning graph has leveled off (no plan), that is, when no new information are generated. GraphPlan always returns optimal plan. Planning Graph opens a wide variety of possibilities in the development in planners in the area of **Reachability Heuristics**.[4] The simplest reachability analysis is whether the goal can be reached from the initial state. Without using planning graph, the search tree will need to be explored using forward chaining, but this takes exponential time as it basically brute-forced the solution. Planning graph can be used to derive several heuristics to improve the search such as level based and cost based heuristics. There are also heuristics to be used in regression as well. Mutexes used in constructing the planning graph are relaxed or ignored in some heuristics, this is useful for stochastic and temporal planning too.

## Ordered Binary Decision Diagram

Binary decision diagram are used in the field of computer aided design, FPGA and logic synthesis rely heavily on optimization/equivalence checking provided by BDDs. It enables formal verification via **model checking**. It works by describing the domain of interest by a semantic model. Then describe the desired property by a logic formula. And finally check to see if the domain satisfy the desired property by checking whether the formula is true in the model. It not hard to draw similarity of formal verification of dynamic system to planning problems. Planning can be solved via symbolic model checking. In planning, the main problem is large state spaces. The idea is to utilize work on symbolic model checking based on OBDD's. It provides two main benefits: 1. canonical form for propositional formulas 2. efficient boolean operations in polynomial time.[5] However, OBDD does require human heuristic specifying ordering on problem input. This cannot be done automatically because choosing the correct ordering to minimize the graph size is itself a co-NP problem.

## Comparison

Partial-order planning dominated the research space for a long time until planning graph was introduced. However, with the heuristics borrowed from planning graphs, partial order planning can achieve better performance as its more parallelizable. In terms of tackling intractibility, partial-order planning or planning graph are both constraint-based approaches. It is perhaps no surprising, because the only way to achieve optimal solution in PSPACE-Complete problems in the worse case is to either 1. constraint on the input or 2. achieve non-optimal plan. If the state-space is large, then a search based algorithm with feasible result will be better. OBDD's is a different planning system with its roots from hardware design. Its usage are in dynamic control systems where model checker has additional information based on the specific problem domain.

---

1. D. S. Nau, Y. Cao, A. Lotem, and H. Muoz-Avila. SHOP: Simple hierarchical ordered planner. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 968-973. Morgan Kaufmann Publishers, Jul 31-August 6 1999 ↩

2. J.S. Penberthy and D. Weld "UCPOP: A Sound, Complete, Partial-Order Planner for ADL," Proceedings of KR-92, 103-114, Cambridge, MA, October 1992 ↩

3. A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis", Artificial Intelligence, 90:281--300 (1997) ↩

4. Daniel Bryce and Subbarao Kambhampati. (2007) Tutorial on Planning Graph–Based Reachability Heuristics. AI Magazine Spring 2007 ↩

5. Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, Vol. C-35, No. 8, August, 1986, pp. 677-691. ↩