Research Review of the Papers:

A. Blum & M. Furst H. A. Kautz & B. Selman Jorg Hoffmann

Field of AI planning and search

Important historical developments in the field of AI planning and search.

Submitted by: Vadym Serpak, v.serpak@gmail.com

Course: Artificial Intelligence Nanodegree, Udacity

Date: 07/19/2017

Developments in Artificial Intelligence

Artificial intelligence (AI) has been one of the most controversial domains of inquiry in computer science since it was first proposed in the 1950s. Defined as the part of computer science concerned with designing systems that exhibit the characteristics associated with human intelligence—understanding language, learning, reasoning, solving problems, and so on (Barr and Feigenbaum, 1981)—the field has attracted researchers because of its ambitious goals and enormous underlying intellectual challenges. The field has been controversial because of its social, ethical, and philosophical implications. [1].

1. Planning

Planning combines the two major areas of AI: search and logic. A planner can be seen either as a program that searches for a solution or as one that (constructively) proves the existence of a solution. The cross-fertilization of ideas from the two areas has led both to improvements in performance amounting to several orders of magnitude in the last decade and to an increased use of planners in industrial applications[2].

1.1. Graphplan

From Wikipedia, the free encyclopedia: **"Graphplan** is an algorithm for automated planning developed by Avrim Blum and Merrick Furst in 1995. Graphplan takes as input a planning problem expressed in STRIPS and produces, if one is possible, a sequence of operations for reaching a goal state. The name *graph*plan is due to the use of a novel *planning graph*, to reduce the amount of search needed to find the solution from straightforward exploration of the *state space 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)
```

[3]

Graphplan takes the initial conditions and operator definitions and uses them to construct a leveled graph. The initial conditions are the first level of the graph. The next level consists of actions that might be performed at time 1. The level after that consists of facts that might be true at time 2. The level after that is actions that might be performed at time 2, etc. In the animation, the initial conditions are in a column at the left edge of the page and the levels go left to right. When we create the graph, we also propagate exclusion relations left to right. These are properties of the graph that are used to limit search. This part is all fairly quick and in any case is polynomial time.

In the searching phase, Graphplan performs a fairly standard sort of backward-chaining search, but using the information propagated when creating the graph. This limits the amount of searching performed. Notice in the animations that the searches get cut off pretty quickly.

One thing to mention: depending on your machine, the time to view the animation is likely a lot longer than the actual time the program takes to construct a plan. For instance, even on a slow DEC2100, it takes less than 1.5 seconds to solve the fixit problem.[4].

1.2. Satplan

From Wikipedia, the free encyclopedia: **"Satplan** (better known as Planning as Satisfiability) is a method for automated planning. It converts the planning problem instance into an instance of the Boolean satisfiability problem, which is then solved using a method for establishing satisfiability such as the DPLL algorithm or WalkSAT.

Given a problem instance in planning, with a given initial state, a given set of actions, a goal, and a horizon length, a formula is generated so that the formula is satisfiable if and only if there is a plan with the given horizon length. This is similar to simulation of Turing machines with the satisfiability problem in the proof of Cook's theorem. A plan can be found by testing the satisfiability of the formulas for different horizon lengths. The simplest way of doing this is to go through horizon lengths sequentially, 0, 1, 2, and so on."

SATPlan pseudocode

```
function SATPLAN(init, actions, goal, T_max) returns solution or failure
  inputs: init, actions, goal, constitute a description of the problem
          T_max, an upper limit for plan length
  for t = 0 to T_max do
      cnf ← TRANSLATE-TO-SAT(init, actions, goal, t)
      model ← SAT-SOLVER(cnf)
      if model is not null then
          return EXTRACT-SOLUTION(model)
  return failure
```

[3]

Reduction to SAT is a very successful approach to solving hard combinatorial problems in Artificial Intelligence and computer science in general. Most commonly, problem instances reduced to SAT are solved with a general-purpose SAT solver. Although there is the obvious possibility of improving the SAT solving process with application-specific heuristics, this has rarely been done successfully. [5]

In this work authors propose a planning-specific variable selection strategy for SAT solving. The strategy is based on generic principles about properties of plans, and its performance with standard planning benchmarks often substantially improves on generic variable selection heuristics, such as VSIDS, and often lifts it to the same level with other search methods such as explicit state-space search with heuristic search algorithms.

1.3. The Fast-Forward Planning System

FAST-FORWARD (FF) was the most successful automatic planner in the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00) planning systems competition. Like the well-known HSP system, FF relies on forward search in the state space, guided by a heuristic that estimates goal distances by ignoring delete lists. It differs from HSF in a number of important details. This article describes the algorithmic techniques used in FF in comparison to HsP and evaluates their benefits in terms of run-time and solution-length behavior.

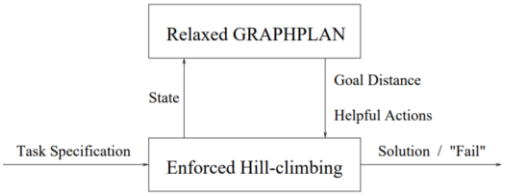

Figure 1: FF's base system architecture. [6]

FF can be seen as an advanced successor of the HSP system, which differs from its predecessor in a number of important details.

1. A more sophisticated method for heuristic evaluation, taking into account positive interactions between facts.
2. A novel kind of local search strategy, employing systematic search for escaping plateaus and local minima.
3. A method that identifies those successors of a search node that seem to be|and usually are| most helpful in getting to the goal[7]

2. Conclusions

Planners such as GRAPHPLAN, SATPLAN, and FF have moved the field of planning forward, by raising the level of performance of planning systems, by clarifying the representational and combinatorial issues involved, and by the development of useful heuristics.[2].

References:

1. Funding a Revolution: Government Support for Computing Research National Academy Press (http://www.nap.edu) 2101 Constitution Ave., NW, Box 285 Washington, D.C. 20055 800-624-6242 202-334-3313 (in the Washington metropolitan area)
2. Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition). Pearson Education, Inc., Upper Saddle River, New Jersey 07458.
3. http://www.cs.nott.ac.uk/~psznza/G52PAS/lecture12.pdf
4. A. Blum and M. Furst (1997). Fast planning through planning graph analysis. Artificial intelligence. 90:281-300. https://www.cs.cmu.edu/~avrim/graphplan.html
5. H. A. Kautz and B. Selman (1992). Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359-363
6. https://www.cs.nmsu.edu/~sto/PlanningPapers/FF%20Planning%20System.pdf
7. FF: The Fast-Forward Planning System. Institute for CS Albert Ludwigs University Georges-K□hler-Allee, Geb. 52 79110 Freiburg, Germany homann@informatik.uni-freiburg.de **http://www.cs.toronto.edu/~sheila/2542/w06/readings/ffplan01.pdf**