Caio Moreira Gomes

Recife, February 12, 2018

Important Advances in AI Planning and Search

INTRODUCTION

In order of accomplish the task proposed by Udacity Artificial Intelligence Nanodegree a review in three of the historic remarks over the AI Planning and Search.

It's important to mention the importance of the studies in this field, which led the development of some of the most important advances in human science, including the usage of these automated planning techniques by NASA for increasing the autonomy of the mars hover robot[1].

1. PDDL (Planning Domain Definition Language)

The PDDL development by Drew McDermott was a huge advance in the field of AI Planning, because it allowed people to develop standard description of planning problems, which didn't exists before. This language uses propositional logic to describe the problem situation and allows you to solve these problems using it. It was developed using as reference STRIPS and ADL, and have been used in the International Planning Competition, which aimed to bring development in this study field. For explain how this description standard works suppose that we have the universe as we have a cake and we may eat it or we may make a cake, but we can only eat a cake if there already exists some cake, and we have only one plate to put the cake, so we can only make a cake if there's no made cake yet. Once we suppose that exists a delicious cake ready to be eaten and we didn't eat any cake yet but we want to achieve the goal state from having a ready cake and already have eaten a cake, so we can describe it in PDDL as follow:

Init *(Have(Cake) ∧ ¬ Eaten(Cake))*

Goal *(Have(Cake) ∧ Eaten(Cake))*

[1] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20050157091.pdf

Action *(Eat (Cake)*

Precond : Have(Cake)

Effect : ¬ Have(Cake) ∧ Eaten(Cake))

Action *(Bake(Cake)*

Precond : ¬ Have(Cake)

Effect : Have(Cake))

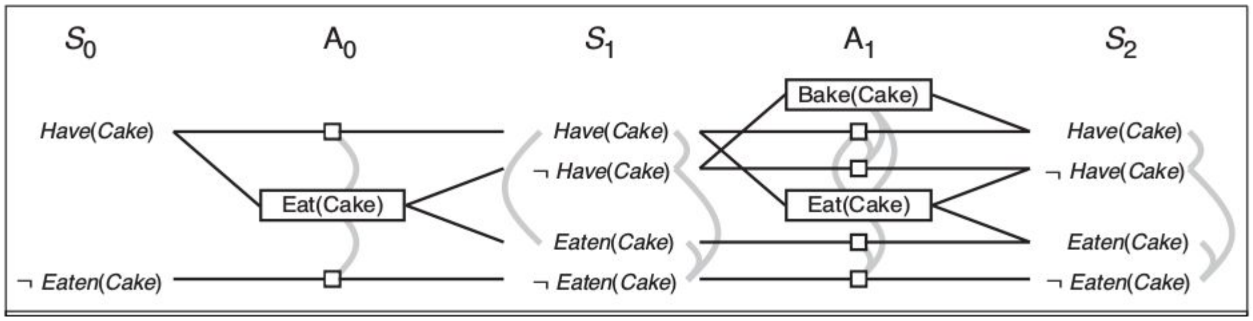
With this method it's possible to describe any situation and apply some algorithm to solve the described problems and also has the advantage from being a standard which allow any one to read it without doubt once get any problem plan.

2. GRAPHPLAN

It's almost impossible to talk about the AI automated planning without talking about GraphPlan[2], which is one of the most importants developments in this field, because this algorithm improved the efficiency of the process by far against the algorithms which existed in the same time as him.

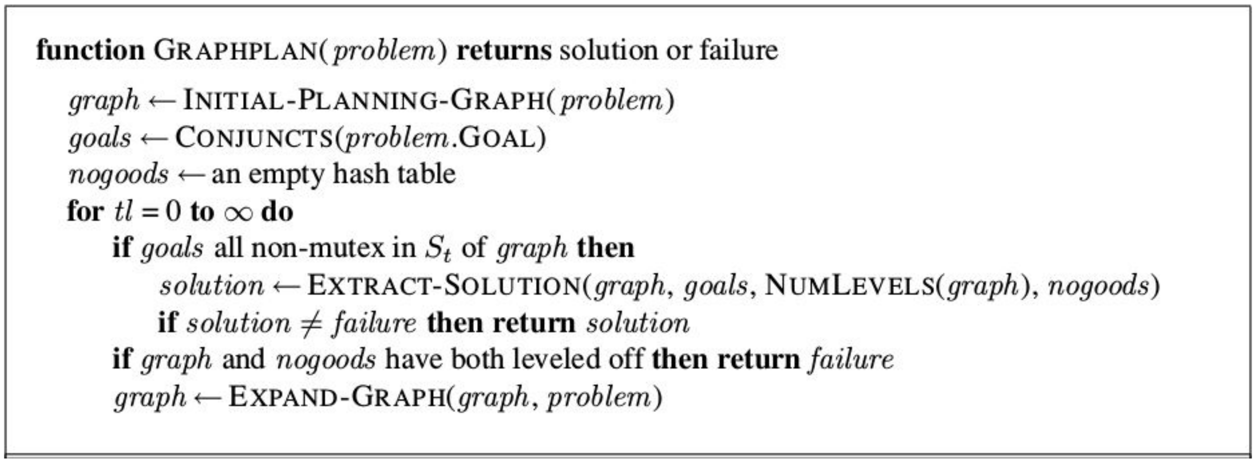
The Graphplan algorithms run in an instance of the planning graph data structure which is a graph organized in levels where the application of actions generate states and the combination of those state them, it builds the next possible actions and the graph is built such as a tree, but there exists some edges between nodes in the same level, since they are not allowed to exist in the same state, these links are called mutex. The mutex links allow the problem to not stop in a state where the goal state is achieved but with the usage of two non co-existing states, also they rule which actions can be executed at every state, it is, suppose the universe in the PDDL section, the planning graph generated would be:

[2] https://www.cs.cmu.edu/~avrim/Papers/graphplan.pdf



Planning graph for the cake problem up to level two - Artificial Intelligence a Modern Approach 3rd Edition

The gray lines represent the mutex edges, and the black lines the edges to the next level of the graph, and the squares represents the persistence actions, it is, the action Have(Cake) is a persistence action, another characteristic of this graph is that once a action appears in one level they keep these actions at every single level. The Graphplan algorithm works over this graph, it aims to find a path to the goal using the planning graph. The algorithm is well described as follow:



```
function GRAPHPLAN(problem) returns solution or failure
  graph ← INITIAL-PLANNING-GRAPH(problem)
  goals ← CONJUNCTS(problem.GOAL)
  nogoods ← an empty hash table
  for tl = 0 to ∞ do
      if goals all non-mutex in $S_t$ of graph then
          solution ← EXTRACT-SOLUTION(graph, goals, NUMLEVELS(graph), nogoods)
          if solution ≠ failure then return solution
      if graph and nogoods have both leveled off then return failure
      graph ← EXPAND-GRAPH(graph, problem)
```

Graphplan Algorithm- Artificial Intelligence a Modern Approach 3rd Edition

3. Symmetries

One of the main points in the search problems is that it contains a really vast amount of possible states in the state space to search until a solution has been found, but, there's a huge amount of situations where parts of the search space of the problem are symmetric, it is, once we find that some plan has a solution, we can modify some of the variables and keep with the same solution, or if a problem do not have a solution, doesn't matters if we modify some of the variables of the problem it will keep unsolvable, for example imagine a planning problem, where we have two cities (C1, C2), two planes (P1, P2) and one cargo (A), if we imagine our problem as follow:

Init *(At(A, C2) ⋀ At(P1, C1) ⋀ At(P2, C1))*
Goal *(At(A, C1))*
Action *(Fly (Plane, Begin, End)*
  Precond : At(Plane, Begin)
  Effect : ¬ At(Plane, Begin) ⋀ At(Plane, End)
Action *(Load(Cargo, Plane, City)*
  Precond : At(Cargo, City) ⋀ At(Plane, City) ⋀ ¬ In(Cargo, Plane)
  Effect : In(Cargo, Plane))

Action *(Unload(Cargo, Plane, City)*
  Precond : At(Cargo, City) ⋀ At(Plane, City) ⋀ In(Cargo, Plane)
  Effect : ¬ In(Cargo, Plane) ⋀ At(Cargo, City))

In this exposed problem, we may use the plan: {Fly(P1, C1, C2), Load(A, P1), Fly(P1, C2, C1)} or the plan {Fly(P2, C1, C2), Load(A, P2), Fly(P2, C2, C1)} which are symmetric, and there's no need to search for both plans in the search space. It's an example of simple symmetry where the simple fact of exchange P1 to P2 resulted in the same situation. In order to explore those symmetries and reduce the search space, David Joslin and Amitabha Roy published in 1997 an article entitled *Exploiting symmetry in lifted CSPs*[3] where they developed a method of search symmetry in planning problems converting it into a CSP(Constraint Satisfaction Problem), which is possible converting the problem states into a graph, and them looking for the symmetry in this graph.

[3] http://www.cse.iitd.ernet.in/~mittal/read_papers/exploiting_symmetries_in_csp.pdf

CONCLUSION

The algorithms and techniques described here represented a huge advancement in the AI Planning and Search field of computer sciences, because it allowed people to solve problems faster, problems which are really hard to handle once some of those has exponential complexity and emerge in real life, so they must be solved and them it's necessary to find and automate techniques to reach the problems solutions. Also have been presented a huge social advancement especially from PDDL once it allowed researchers to communicate in a better way standardizing a method to represent every problem.

Them it's possible to reflect that this problem has a really representative situation in real life problems, especially in logistic companies, but have been obfuscated by the emerging algorithms of machine learning and cognitive computing, once they have been causing really astonishing advancements in all areas of computer sciences, against domain specific necessities as the planning is for logistics.

REFERENCES

- Jussi Rintanen and Jorg Hoffmann, An overview of recent algorithms for AI planning. http://www.cs.toronto.edu/~sheila/2542/w06/readings/RintanenHoffmann01.pdf
- David Joslyn and Amitabha Roy, Exploiting Symmetry in Lifted CSPs http://www.cse.iitd.ernet.in/~mittal/read_papers/exploiting_symmetries_in_csp.pdf
- Avrim L. Blum and Merrick L. Furst, Fast Planning Through Planning Graph Analysis https://www.cs.cmu.edu/~avrim/Papers/graphplan.pdf
- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach 3rd edition.