Caio Moreira Gomes

Recife, February 12, 2018

AI Planning and Search Algorithms Analysis

INTRODUCTION

In order of accomplish the task proposed by Udacity Artificial Intelligence Nanodegree an analysis of different search algorithms and heuristics have been made to know which one is the best to be applied for every case.

It's important to mention that the software analysis have been done using the pypy JIT Compiler for python in order to reduce execution time.

Methodology

The analysis in this review used one specific efficiency measure to assume which one is the best search method, this efficiency proposes to degrade most the position of searches which spend a lot of nodes to find a solution, following it we degrade not so much by the amount of time necessary to find the solution and the number of nodes expansion necessary to find the solution, because once the computational power gets bigger the node expansion means a theoretic bigger amount of time necessary to spend. The efficiency equation can be described as follows based in the degrade rate proposed:

$$\eta = \frac{1}{2^s \cdot t \cdot E}$$

where the s represents the solution size, or the cost of the solution, the t represents the computational time necessary for solving the problem, E is the node expansion.

Also in the tables exposed here the values are placed in a relative value, it is, the efficiency only is useful to measure how good is a certain method against others being applied to the same situation, so the tables here for an easier visualization are represented in percentage of the best efficiency computed.

1. Non Heuristic Search Methods

The non Heuristic search methods used in this task are Breadth First Search(BFS), Depth First Graph Search(DFS), Uniform Cost Search(UCS or Dijkstra algorithm), the tests reveal the following table:

	Problem 1				
Method	Node Expansion	Time (s)	Cost	N/s	Efficiency
Breadth First Search	43	0,1074579	6	400	100,00%
Depth First Graph Search	21	0,0417096	20	503	0,03%
Uniform Cost Search	55	0,1043957	6	527	80,48%
	Problem 2				
Method	Node expansion	Time (s)	Cost	N/s	Efficiency
Breadth First Search	3343	4,9828154	9	671	100,00%
Depth First Graph Search	624	1,8730109	619	333	0,00%
Uniform Cost Search	4852	7,5767614	9	640	45,31%
	Problem 3				
Method	Node expansion	Time (s)	Cost	N/s	Efficiency
Breadth First Search	14663	17,5333031	12	836	100,00%
Depth First Graph Search	408	0,6687978	392	610	0,00%
Uniform Cost Search	18235	21,3191185	12	855	66,13%

With the usage of the efficiency indicator it's easy to know that the best search is the BFS, because even that the DFS is the fastest one it returns a awful solution then, comes the UCS but it loses in speed, it occurs once every node has an equal cost them the BFS has lower amount of computational power necessary to find an optimal solution, the result would revert once the actions or states have been attributed different costs.

2. Heuristic Search

The heuristic searches are Greedy Best First Graph Search Heuristic h_1, A* Search h_1, A* Search Heuristic Ignore Preconditions and A* Search Heuristic Planning Graph Level sum, these searches originated the following table:

	1				
Method	Node Expansion	Time (s)	Cost	N/s	Efficiency
Greedy Best First Graph Search Heuristic 1	7	0,0212680	6	329	100,00%
A* Search h_1	55	0,1042810	6	527	2,60%
A* Search Heuristic Ignore Preconditions	41	0,1197009	6	343	3,03%
A* Search Heuristic P.G. Level Sum	41	1,1211233	6	37	0,32%
	2				
Method	Node expansion	Time (s)	Cost	N/s	Efficiency
Greedy Best First Graph Search Heuristic 1	990	1,5399287	17	643	1,53%
A* Search h_1	4852	6,0513578	9	802	20,35%
A* Search Heuristic Ignore Preconditions	1450	4,1204538	9	352	100,00%
A* Search Heuristic P.G. Level Sum	1450	48,8442388	9	30	8,44%
	3				
Method	Node expansion	Time (s)	Cost	N/s	Efficiency
Greedy Best First Graph Search Heuristic 1	5614	6,7514355	22	832	0,15%
A* Search h_1	18235	22,8876109	12	797	13,88%
A* Search Heuristic Ignore Preconditions	5040	11,4956843	12	438	100,00%
A* Search Heuristic P.G. Level Sum	5040	211,170111	12	24	5,44%

As it's possible to see in two of three cases the best one in the efficiency methodology is the A* using the Ignore Preconditions Heuristics. It's important to note that the greedy algorithm is faster but out of the problem one it start generating paths with a higher cost which are degraded by our efficiency equation. The A* Ignore Preconditions has the same amount of node expansion from the A* with level sum heuristic, but, it's clearly better because of the amount of time necessary to compute it, this situation can be explained by the amount of time necessary to build the planning

graph, once the number of nodes expanded tends to be the same as the Ignore Preconditions heuristic. This difference in the computational complexity of these algorithms is well exposed as follows:

> *"A planning graph is polynomial in the size of the planning problem. For a planning problem with l literals and a actions, each S i has no more than l nodes and l 2 mutex links, and each A i has no more than a + l nodes (including the no-ops), (a + l) 2 mutex links, and 2(al + l) precondition and effect links. Thus, an entire graph with n levels has a size of O(n(a + l) 2). The time to build the graph has the same complexity." (Artificial Intelligence: A Modern Approach 3rd edition, p. 382)*

This complexity is much higher than the implementation complexity of Ignore Preconditions heuristics which sums all the missing goals, being O(n), where n is the number of goals in the problem.

3. Optimal Results

Based in the results collected by the best results in tests the solution to the problems are as follow:

Problem 1	Problem 2	Problem 3
Load(C1, P1, SFO) **Fly(P1, SFO, JFK)** **Unload(C1, P1, JFK)** **Load(C2, P2, JFK)** **Fly(P2, JFK, SFO)** **Unload(C2, P2, SFO)**	**Load(C3, P3, ATL)** **Fly(P3, ATL, SFO)** **Unload(C3, P3, SFO)** **Load(C1, P1, SFO)** **Fly(P1, SFO, JFK)** **Unload(C1, P1, JFK)** **Load(C2, P2, JFK)** **Fly(P2, JFK, SFO)** **Unload(C2, P2, SFO)**	**Load(C2, P2, JFK)** **Fly(P2, JFK, ORD)** **Load(C4, P2, ORD)** **Fly(P2, ORD, SFO)** **Unload(C4, P2, SFO)** **Load(C1, P1, SFO)** **Fly(P1, SFO, ATL)** **Load(C3, P1, ATL)** **Fly(P1, ATL, JFK)** **Unload(C3, P1, JFK)** **Unload(C1, P1, JFK)** **Unload(C2, P2, SFO)**

CONCLUSION

These search algorithms has really different results, but, using the efficiency methodology it's possible to see that the best way to find an answer for every single problem is using the A* Search with the Ignore Preconditions heuristic, it's important to note that it only loses for the greedy algorithm in the first instance, where the average error goes to zero.

REFERENCES

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach 3rd edition.