Amichai Antebi

AIND Project 3, Implement a Planning Search

June 29, 2017

Heuristic Analysis

# Optimal plans samples:

|  |  |  |
| --- | --- | --- |
| **Problem 1** – 6 actions | **Problem 2** – 9 actions | **Problem 3** – 12 actions |
| Load(C2, P2, JFK)  Load(C1, P1, SFO)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK) | Load(C2, P2, JFK)  Load(C1, P1, SFO)  Load(C3, P3, ATL)  Fly(P2, JFK, SFO)  Unload(C2, P2, SFO)  Fly(P1, SFO, JFK)  Unload(C1, P1, JFK)  Fly(P3, ATL, SFO)  Unload(C3, P3, SFO) | Load(C2, P2, JFK)  Load(C1, P1, SFO)  Fly(P2, JFK, ORD)  Load(C4, P2, ORD)  Fly(P1, SFO, ATL)  Load(C3, P1, ATL)  Fly(P1, ATL, JFK)  Unload(C1, P1, JFK)  Unload(C3, P1, JFK)  Fly(P2, ORD, SFO)  Unload(C2, P2, SFO)  Unload(C4, P2, SFO) |

# Result data (good results in comparison to others are marked with green background):

Table 1 – Problem 1 Results – The easiest problem, we can run all the available searches very fast

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Search** | **Expansions** | **Goal Tests** | **Time (s)** | **Optimality** | **Plan Length** |
|  |  |  |  |  |  |
| breadth\_first\_search | 43 | 56 | 0.05 | Yes | 6 |
| breadth\_first\_tree\_search | 1458 | 1459 | 1.4 | Yes | 6 |
| depth\_first\_graph\_search | 12 | 13 | 0.01 | No | 12 |
| depth\_limited\_search | 101 | 271 | 0.14 | No | 50 |
| uniform\_cost\_search | 55 | 57 | 0.05 | Yes | 6 |
| recursive\_best\_first\_search | 4229 | 4230 | 4.6 | Yes | 6 |
| greedy\_best\_first\_graph\_search | 7 | 9 | 0.007 | Yes | 6 |
|  |  |  |  |  |  |
| A\* using h1 heuristic | 55 | 57 | 0.055 | Yes | 6 |
| A\* using h\_ignore\_preconditions | 41 | 43 | 0.06 | Yes | 6 |
| A\* using h\_pg\_levelsum | 11 | 13 | 0.95 | Yes | 6 |

Table 2 – Problem 2 Results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Search** | **Expansions** | **Goal Tests** | **Time (s)** | **Optimality** | **Plan Length** |
|  |  |  |  |  |  |
| breadth\_first\_search | 3343 | 4609 | 20.1 | Yes | 9 |
| depth\_first\_graph\_search | 582 | 583 | 4.4 | No | 575 |
| greedy\_best\_first\_graph\_search | 990 | 992 | 3.5 | No | 15 |
|  |  |  |  |  |  |
| A\* using h1 heuristic | 4852 | 4854 | 17.9 | Yes | 9 |
| A\* using h\_ignore\_preconditions | 1450 | 1452 | 8.5 | Yes | 9 |
| A\* using h\_pg\_levelsum | 86 | 88 | 96.7 | Yes | 9 |

Table 3 – Problem 3 Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Search** | **Expansions** | **Goal Tests** | **Time (s)** | **Optimality** | **Plan Length** |
|  |  |  |  |  |  |
| breadth\_first\_search | 14663 | 18098 | 153.4 | Yes | 12 |
| depth\_first\_graph\_search | 627 | 628 | 4.7 | No | 596 |
| greedy\_best\_first\_graph\_search | 5614 | 5616 | 26.3 | No | 22 |
|  |  |  |  |  |  |
| A\* using h1 heuristic | 18235 | 18237 | 79.4 | Yes | 12 |
| A\* using h\_ignore\_preconditions | 5040 | 5042 | 25.9 | Yes | 12 |
| A\* using h\_pg\_levelsum | 316 | 318 | 367.6 | Yes | 12 |

# Compare and contrast Uninformed (Non Heuristic) Search

In comparing breadth\_first\_search, depth\_first\_graph\_search and greedy\_best\_first\_graph\_search we see that only breadth\_first\_search gives us the optimal plan for all the 3 problems. When we compare the execution speed we see that depth\_first\_graph\_search is much faster as we could expect because it going deep into the first solution it can find, so we get the solution very fast and with much fewer expansions even for the 3’rd problem, but we pay with plan length that is far away from the optimal plan.

We can say that if we must find an optimal solution, then the **breadth\_first\_search** will give us the optimal plan because it moves to the shortest path on every move so it must be optimal, but we pay a large amount of memory usage (expansions) and slower execution speed.

On the other hand, if we can compromise a little on the optimality and still get a result that is not so far away, we may want to use **greedy\_best\_first\_graph\_search** which unlike depth\_first\_graph\_search it does give us close to optimal plan and still we save a lot of memory (much fewer expansions) and also get better execution speed. We get those results because it expands first to the path that closest to the goal, so it can’t guaranty the optimal solution but it will be close to optimal and match fewer expansions.

# Compare and contrast Informed (Heuristic) Search

We can see that all A\* searches are optimal, but we have a big difference in the expansions number and in the execution time.

A\* using h\_pg\_levelsum is using much fewer expansions, as we could expected because of the precise heuristic which using a planning graph and sum the level costs for every goal requirement, but this heuristic is also very expensive on time-consuming and therefore using h\_pg\_levelsum is only recommended if we much more care of memory usage and less for the execution time.

On the other hand, if we most care of execution time, then A\* using h\_ignore\_preconditions is much faster, we could expected those result because in this heuristic we use arelaxed problem by ignoring the preconditions so it ismuch simpler and need much less time consuming but it’s still very precise in comparison to the h1 heuristic (which is Non-heuristic and actually the same as using uniform\_cost\_search).

# The best heuristic

We can say that most of the times the **A\* using h\_ignore\_preconditions** will be the best choice, it gives us the optimal solution with the best execution time and better memory usage in comparison to optimal non-heuristics searches.

If we must use less memory then **A\* using h\_pg\_levelsum** is our best choice because it is still optimal and have the best usage of the memory, we will pay with much longer execution time