**Heuristic Analysis for Planning Project**

June 6, 2017

**1. OPTIMAL SOLUTION FOR EACH PROBLEM**

**Problem 1** initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK)

∧ At(P1, SFO) ∧ At(P2, JFK)

∧ Cargo(C1) ∧ Cargo(C2)

∧ Plane(P1) ∧ Plane(P2)

∧ Airport(JFK) ∧ Airport(SFO))

Goal(At(C1, JFK) ∧ At(C2, SFO))

An optimal plan for Problem 1:

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

**Problem 2** initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)

∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)

∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)

∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))

Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

An optimal plan for Problem 2:

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)

**Problem 3** initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)

∧ At(P1, SFO) ∧ At(P2, JFK)

∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)

∧ Plane(P1) ∧ Plane(P2)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))

Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

An optimal plan for Problem 3:

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C4, P2, SFO)

Unload(C3, P1, JFK)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

**2. ANALYSIS OF UNINFORMED PLANNING ALGORITHMS**

This section compares three uninformed planning algorithms, breadth\_first\_search, depth\_first\_graph\_search and greedy\_best\_first\_graph\_search, for each problem.

As seen in Figure 1a, 1b and 1c, breadth\_first\_search found the optimal solution for each problem. However, it is more time-consuming and requires much more memory. According to Norvig and Russell’s textbook Artificial Intelligence: A Modern Approach (3e) (AIMA), space complexity is a bigger problem for breadth\_first\_search than is time complexity. Therefore, while I’d recommend this type of search for relatively simple problems like these Air Cargo problems, I would think twice for much more complex problems that have very large branching factors.

depth\_first\_graph\_search is less costly in terms of time and space usage for these three problems, but the plan lengths are extremely long, especially for problems 2 and 3. None of the solutions are optimal, just as the AIMA textbook mentions that depth\_first\_search is not optimal. I would not recommend using it for these not-so-complicated problems.

greedy\_best\_first\_graph\_search appears to be a nice compromise. It is much faster than breadth\_first\_search. Without any heuristic function, it was able to find an optimal plan for problem 1, and the plan lengths for the other two problems are not so bad compared to depth\_first\_graph\_search. As mentioned in the AIMA textbook, greedy\_best\_first\_graph\_search will perform even better with a good heuristic function. Therefore, while I do not recommend greedy\_best\_first\_graph\_search for these three problems as it can’t find the optimal solution most of the time, it can be used instead of breadth\_first\_search for much more complex problems to save time and memory while achieving reasonably good results.

***Figure 1a – Metrics of breadth\_first\_search, depth\_first\_graph\_search and greedy\_best\_first\_graph\_search for Problem 1***

|  |  |  |  |
| --- | --- | --- | --- |
|  | breadth\_first\_search | depth\_first\_graph\_search | greedy\_best\_first\_graph\_search |
| Node expansions | 43 | 12 | 7 |
| Goal tests | 56 | 13 | 9 |
| Time elapsed (seconds) | 0.0511 | 0.0225 | 0.0099 |
| Plan length | 6 | 12 | 6 |
| Optimality of solution | Yes | No | Yes |

***Figure 1b – Metrics of breadth\_first\_search, depth\_first\_graph\_search and greedy\_best\_first\_graph\_search for Problem 2***

|  |  |  |  |
| --- | --- | --- | --- |
|  | breadth\_first\_search | depth\_first\_graph\_search | greedy\_best\_first\_graph\_search |
| Node expansions | 3343 | 582 | 998 |
| Goal tests | 4609 | 583 | 1000 |
| Time elapsed (seconds) | 14.9 | 3.5 | 2.9 |
| Plan length | 9 | 575 | 21 |
| Optimality of solution | Yes | No | No |

***Figure 1c – Metrics of breadth\_first\_search, depth\_first\_graph\_search and greedy\_best\_first\_graph\_search for Problem 3***

|  |  |  |  |
| --- | --- | --- | --- |
|  | breadth\_first\_search | depth\_first\_graph\_search | greedy\_best\_first\_graph\_search |
| Node expansions | 14663 | 627 | 5578 |
| Goal tests | 18098 | 628 | 5580 |
| Time elapsed (seconds) | 104.7 | 3.6 | 17.9 |
| Plan length | 12 | 596 | 22 |
| Optimality of solution | Yes | No | No |

**3. ANALYSIS OF A\* SEARCHES USING HEURISTICS**

This section compares two A\* planning searches using “ignore-preconditions” and “level-sum” from the Planning Graph. Both searches achieve an optimal solution. However, A\* search with “ignore-preconditions” is much faster. Although it has more node expansions than A\* search with “level-sum”, the space complexity is not so bad. For these problems, I would recommend A\* search with “ignore-preconditions”.

***Figure 2a – Metrics of A\* planning searches using “ignore-preconditions” and “level-sum” for Problem 1***

|  |  |  |
| --- | --- | --- |
|  | A\* search with “ignore-preconditions” | A\* search with “level-sum” |
| Node expansions | 41 | 11 |
| Goal tests | 43 | 13 |
| Time elapsed (seconds) | 0.0597 | 0.6211 |
| Plan length | 6 | 6 |
| Optimality of solution | Yes | Yes |

***Figure 2b – Metrics of A\* planning searches using “ignore-preconditions” and “level-sum” for Problem 2***

|  |  |  |
| --- | --- | --- |
|  | A\* search with “ignore-preconditions” | A\* search with “level-sum” |
| Node expansions | 1450 | 86 |
| Goal tests | 1452 | 88 |
| Time elapsed (seconds) | 4.8 | 39.5 |
| Plan length | 9 | 9 |
| Optimality of solution | Yes | Yes |

***Figure 2a – Metrics of A\* planning searches using “ignore-preconditions” and “level-sum” for Problem 1***

|  |  |  |
| --- | --- | --- |
|  | A\* search with “ignore-preconditions” | A\* search with “level-sum” |
| Node expansions | 5040 | 318 |
| Goal tests | 5042 | 320 |
| Time elapsed (seconds) | 17.8 | 196.6 |
| Plan length | 12 | 12 |
| Optimality of solution | Yes | Yes |

**CONCLUSION:** A\* searches with heuristics are overall better than the three uninformed searches analyzed in section 1. In particular, A\* search with “ignore-preconditions” is faster and requires less memory than breadth\_first\_search and is able to find the optimal solution. Therefore, A\* search with “ignore-preconditions” is the best choice for Air Cargo problems.