

Parth Bhoiwala

Udacity AI Nanodegree (May 2017)
AI Planning Search - Heuristic Analysis

The goal of this project was to define a group of problems in classical PDDL (Planning Domain Definition Language) for the air cargo domain discussed in the lectures. A planning graph was used along with many search methods and heuristic functions to implement the plan. There were total of three air cargo problems and below are the problem's initial state and goal.

Problem 1:

```
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))
```

Problem 2:

```
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))
```

Problem 3:

```
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))
```

For each of the problems mentioned above, the plan search was computed and the results are recorded below.

Problem 1 was run for all the search functions and Table 1 below shows the results.

Table 1: Air Cargo Problem 1 Results

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|-------------------------------------|-------------|--------------|------------|------------|-----------|
| Breadth First | 6 | 0.054 | 43 | 56 | 180 |
| Breadth First Tree | 6 | 1.069 | 1458 | 1459 | 5960 |
| Depth First Graph | 12 | 0.009 | 12 | 13 | 48 |
| Depth Limited | 50 | 0.101 | 101 | 271 | 414 |
| Uniform Cost | 6 | 0.041 | 55 | 57 | 224 |
| Recursive Best First (h1) | 6 | 3.091 | 4229 | 4230 | 17029 |
| Greedy Best First Graph (h1) | 6 | 0.006 | 7 | 9 | 28 |

As it can be seen, the optimal solutions include the ones with Breadth First, Breadth First Tree, Uniform Cost, Recursive Best First (h1) and Greedy Best First Graph (h1) search. The reason why these are considered optimal is because their plan length is the lowest at 6. Other search methods have plan lengths of 12 and 50 which is a lot of unnecessary work. Out of the five optimal solutions, the best solution would be Greedy Best First Graph search with h1. Not only it provides a plan length of 6 but it's execution time was the fastest at 0.006 seconds and only expanded 7 nodes. The solution provided by Greedy BFG search is shown below:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Problem 2 was run for Breadth First, Depth First, Uniform and Greedy Best First searches only. The other search functions were terminated after running for more than 10 minutes.

Table 2: Air Cargo Problem 2 Results

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|-------------------------------------|-------------|--------------|------------|------------|-------------|
| Breadth First | 9 | 14.26 | 3401 | 4672 | 31049 |
| Depth First Graph | 346 | 1.523 | 350 | 351 | 3142 |
| Uniform Cost | 9 | 12.84 | 4761 | 4763 | 43206 |
| Greedy Best First Graph (h1) | 9 | 1.425 | 550 | 552 | 4950 |

For problem 2, the optimal solutions include Breadth First, Uniform Cost and Greedy BFG search as they all yield a plan with a plan length of 9. DFG Search yielded a plan with 346 which is too inefficient. Out of the 3 optimal searches, Greedy BFG search wins again this time with the fastest execution time of 1.425 seconds. Below is the solution provided by Greedy BFG search:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Problem 3 was also run for Breadth First, Depth First, Uniform and Greedy Best First searches only. The other search functions were terminated after running for more than 10 minutes.

Table 3: Air Cargo Problem 3 Results

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|-------------------------------------|-------------|--------------|--------------|--------------|---------------|
| Breadth First | 12 | 99.32 | 14491 | 17947 | 128184 |
| Depth First Graph | 1878 | 19.06 | 1948 | 1949 | 16253 |
| Uniform Cost | 12 | 52.99 | 17783 | 17785 | 155920 |
| Greedy Best First Graph (h1) | 22 | 12.05 | 4031 | 4033 | 35794 |

For problem 3, Breadth First and Uniform Cost were optimal because they had smallest plan length of 12. Depth First Graph is too inefficient because it's plan length was 1878. Now, Greedy Best First Graph yielded a plan length of 22 which could or could not be considered optimal. This is tricky because a little more analysis is required. Breadth First and Uniform Cost, which yielded a plan length of 12 had execution times of 99.32 and 52.99 seconds respectively. On the other hand, Greedy BFG search, which yielded a plan length of 22 had an execution time of only 12.05 seconds.

So the question to ask here is *"What's more important? Plan length or Execution time?"*.

Because Greedy BFG search wins by providing the fastest execution time of 12.05 seconds which is 4 times as fast as Uniform cost and 9 times faster than Breadth first search. The catch is that Greedy BFG search costs 10 extra steps and if that doesn't matter, then it wins otherwise Uniform Cost wins. Below are the solutions for Uniform Cost and Greedy BFG search.

Uniform Cost

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, ATL, JFK)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Greedy BFG with h1

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ORD)
Load(C4, P1, ORD)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, SFO)
Fly(P1, ORD, ATL)
Unload(C4, P1, ATL)
Fly(P1, ATL, ORD)
Fly(P2, SFO, ATL)
Load(C4, P2, ATL)
Fly(P2, ATL, ORD)
Unload(C3, P2, ORD)
Load(C3, P1, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ORD, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Fly(P1, JFK, ORD)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
```

After running these uninformed search functions, informed search functions were run on those three problems.

Below is the result for running problem 1 on A* with h1, A* with ignore_preconditions, and A* with level_sum.

Table 4: Problem 1 - Informed Search Results (Air Cargo Problem)

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|------------------------------|-------------|--------------|------------|------------|------------|
| A* with h1 | 6 | 0.036 | 55 | 57 | 224 |
| A* with ignore_preconditions | 6 | 0.041 | 41 | 43 | 170 |
| A* with level_sum | 6 | 0.979 | 11 | 13 | 50 |

The results in the above table 5 indicate something interesting. It can be concluded that all three search algorithms were optimal because they all yielded solutions with a plan length of 6. However, A* with h1 executed the problem in 0.036 seconds which is significantly faster compared to A* with level_sum which took 0.979 seconds to run. So clearly, A* with h1 wins. Although, another interesting thing to note is that A* with level_sum expanded only 11 nodes during its execution time. Below is the plan suggested by A* with h1:

```

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

```

Same searches were run again for Problem 2 and below are the results for that.

Table 5: Problem 2 - Informed Search Results (Air Cargo Problem)

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|-------------------------------------|-------------|--------------|-------------|-------------|--------------|
| A* with h1 | 9 | 13.03 | 4761 | 4763 | 43206 |
| A* with ignore_preconditions | 9 | 4.684 | 1450 | 1452 | 13303 |
| A* with level_sum | 9 | 197.1 | 86 | 88 | 841 |

For problem 2, the results are a little different from the previous one. The plan length for all three searches are 9 which makes all of them optimal. However, unlike the previous one in which A* with h1 provided the fastest execution, it was almost 3 times slower than A* with ignore_preconditions. A* with h1 executed in 13.03 seconds while A* with ignore_preconditions executed in only 4.684 seconds making it the winner. However, one interesting thing to note here is that the number of nodes expanded is also much larger for A* with ignore_preconditions at 1450 while only 4761 for A* with h1. Below is the plan suggested by A* with ignore_preconditions:

```

Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

```

Problem 3 was run for only A* with h1 and A* with ignore_preconditions. It was terminated for A* with level_sum after 10 minutes.

Table 5: Problem 3 - Informed Search Results (Air Cargo Problem)

| Search | Plan Length | Time(s) | Expansions | Goal Tests | New Nodes |
|------------------------------|-------------|---------|------------|------------|-----------|
| A* with h1 | 12 | 54.44 | 17783 | 17785 | 155920 |
| A* with ignore_preconditions | 12 | 17.33 | 5003 | 5005 | 44586 |

As it can be seen above, both A* with h1 and A* with ignore_preconditions are optimal with a plan length of 12; however, A* with ignore_preconditions wins easily by providing an execution time of 17.33 seconds. Below is the plan:

```

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(C2, P2, SFO)
Unload(C1, P1, JFK)

```

It is fair to say that Greedy Best First Graph search with H1 was the most optimal because not only it founded the solution in least amount of time but it also calculated the smallest possible plans in terms of length (with a small exception in problem 3) discussed above. The reason why Greedy BFG search outperformed other search functions is mainly because the, as its name suggest, the search technique is greedy. That is, it looks and follows the path that it thinks is closest to the goal. The evaluation on how close the current state is from the goal is determined by the heuristic function and as it was emphasized in the lecture videos, if the heuristic evaluation function is good, greedy best first search is hard to beat. Depth first search and Breadth first search could outperform Greedy best search in certain

cases but for an average scenario, Greedy BFG search should be the way to go for its optimal and complete plan.

For the informed searches that were run, it was noticed that A* with ignore_preconditions was optimal and the best choice as it was also the fastest. In the first problem, it was slightly slower than A* with h1, however, it was fastest in average case. A* search is a special case of the Greedy Best first search as it combines Greedy BFG search with Breadth first search. Out of the three A* searches, the one with ignore_preconditions outperformed the others because of the heuristic function. The level_sum heuristic timed out for third problem and h1 did not perform as good as the one with ignore_preconditions which proves the point discussed in lecture that having a good heuristic evaluation function can have a significant impact on your search algorithm.

In conclusion, it is proved that having the best possible search doesn't mean it's the best solution. Having a good heuristic function is vital and can create a significant impact on the overall implementation.