Heuristic Analysis: Classical Planning

By Simon Müller

Optimal Plans for Problems 1 to 3

The initial state and goal for problem 1 is given by

```
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))
```

In words: In the goal state, cargo C1 needs to be on airport JFK and cargo C2 on airport SFO. Initially, cargo C1 is in SFO and cargo C2 on JFK. We have two planes (P1 and P2), where plane P1 is on SFO and P2 is on JFK. So, the optimal way to solve this planning problem is:

1. Load C1 in P1 -> Fly P1 from SFO to JFK -> Unload C1 from P1 -> Goal achieved
2. Load C2 in P2 -> Fly P2 from JFK to SFO -> Unload C2 from P2 -> Goal achieved

Overall, we need 6 actions in the optimal plan.

The initial state and goal for problem 2 is given by

```
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))
```

Similar as in problem 1, we have on each airport on piece of cargo and one plane. Each cargo needs to be transported to another airport. The optimal plan is then, as in problem 1, to load the cargo into the plane that is located on the same airport and fly afterwards to the desired target airport.

1. Load C1 in P1 -> Fly P1 from SFO to JFK -> Unload C1 from P1
2. Load C2 in P2 -> Fly P2 from JFK to SFO -> Unload C2 from P2
3. Load C3 in P3 -> Fly P3 from ATL to SFO -> Unload C3 from P3

Overall, we need 9 actions in the optimal plan.

The initial state and goal for problem 3 is given by

```
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))
```

This problem is more complicated than problem 1 and 2. We have 4 pieces of cargo on 4 different airports, but we have just 2 planes. The minimum number of actions for this problem is 14 and the sequences are:

1. Load C1 in P1 -> Fly P1 from SFO to JFK -> Unload C1 -> Fly P1 from JFK to ATL -> Load C3 -> Fly P1 from ATL to JFK -> Unload C1
2. Load C2 in P2 -> Fly P2 from JFK to SFO -> Unload C2 -> Fly P2 from SFO to ORD -> Load C4 -> Fly P2 from ORD to SFO -> Unload C4

Unfortunately, each plane must fly one route empty else we would not achieve the goal state.

Note: After comparing my solution to the solution from the planning simulation, I recognized that I made a wrong assumption. I believed that a plane just can load one piece of cargo, but that is not true (and never be mentioned).

Compare and contrast non-heuristic search result metrics

	Search Method	Expansions	Goal Tests	New Node	Path length	Time elapse in [s]
Problem 1	Breadth-first	43	56	180	6	0.032
	Depth-first	12	13	48	12	0.0085
	Greedy Graph H1	7	9	28	6	0.006
Problem 2	Breadth-first	3341	4609	30509	9	15.48
	Depth-first	582	583	5211	575	3.47
	Greedy Graph H1	990	992	8910	9	2.58
Problem 3	Breadth-first	14663	18098	129631	12	113.37
	Depth-first	627	628	5176	596	3.64
	Greedy Graph H1	5614	5616	49429	22	17.75

Additional to the breadth-first (1) and depth first searching (2), I choose Greedy best first graph search h1 (3) as the third algorithm. If we look at the metric path length, then just algorithm (1) returns in all 3 problems the optimal path length. Algorithm (2) even deliver really worse paths (in the sense of far away from the shortest path length). So, When there are high costs for an action (as flying from A to B as in our problem), then the algorithm that delivers the optimal path should be chosen. But when we have also to consider time, then algorithm (1) is for all 3 problems the worst one. Best algorithm is in view of this metric algorithm (2). So, if an action is cheap and time is more critical, then one should choose algorithm (2). Algorithm (3) is in some way a trade-off between (1) and (2). (3) is compared to (1) much faster and compared to (2), the algorithm delivers a not too far away solution of the optimal plan. Depending on costs of an action and how fast results are necessary, algorithm (3) may be a good choice.

The good results (path length) of algorithm (1) are a result of the search strategy. This algorithm searches much more nodes and do much more expansions as the both other algorithms in each problem. So, that we have the guaranty to find the optimal solution. Algorithm (2) just searches in the depth. So, a solution may be found (in a finite tree), but this must not be the optimal one (similarly to the route problem from the video lectures). Algorithm (3) searches compared to (1) much less nodes and do much less (expansions). As we see in problem 3, this may lead to (if we see it as a graph problem) a situation during the search where this algorithm choose the “wrong” (that lead to the optimal path) direction in the graph and find a suboptimal solution.

	Search Method	Expansions	Goal Tests	New Node	Path length	Time elapse in [s]
Problem 1	A* (Level Sum)	11	13	50	6	0.89
	A* (ignore pre.)	41	43	170	6	0.046
Problem 2	A* (Level Sum)	86	88	841	9	199.10
	A* (ignore prec.)	1450	1452	13303	9	4.86
Problem 3	A* (Level Sum)	14663	18098	129631	12	113.37
	A* (ignore pre.)	5040	5042	44944	12	19.78

When comparing the heuristic ‚level-sum‘ and ‚ignore precondition‘ in an A* search one general sees that the heuristic ‘ignore precondition’ (IP) has a much better performance (time metric) than the ‘level-sum’ (LS) heuristic. Both deliver in all 3 problems the optimal path length. There is one interesting observation: while LS has in problem 1 and 2 much less expansion, goal tests, and new nodes than IP, it is switching on problem 3.