Heuristic analysis

Runkun Miao

In this project, we implement several heuristic functions into A* search algorithm to plan three transporting cargo problems, and the initial states and goals has been shown in figure1.

A* search is one of the most popular search algorithm which sums up the cost of path from the starting nodes g(n) and the cheapest cost using heuristic function h(n), the numerical description has been shown below [1],

$$f(n) = g(n) + h(n)$$

Three heuristic functions have been implemented. H_1, ignore precondition, and level sum heuristic. H_1, technically, is not a heuristic function, but a default value function. Ignore precondition is to ignore any precondition and assume that every goal can be achieve in one action and calculate the minimum cost for achieving all the goals [2]. Level sum uses sub goal independent assumption that assume the cost of achieving goals independently and that of achieving goals together are the same [2]. Level sum sums the level costs of the goals in the planning graph.

By comparing the test result for each heuristic function (table 2), we can see that level sum has better performance in node expansions and new nodes searching. But usually it eats more computational power. Especially in problem 3. Where ignore precondition only costs 21 sec to find optimal solution, level sum search used up almost seven minutes to provide the result. However, the number of expansions, goal tests and new nodes are largely decreased using level sum search. During more discrete problems with more goals, we may expect level sum heuristic will outperform ignore precondition heuristic [2].

Apart from a* search algorithm, we also ran some tests with breath first search, depth first search and uniform cost search. Comparing with A* search, those two searches except depth first search has enormous node expansion, but depth first search always fails to find optimal solution, so applying such heuristic function is implausible. Between breath first search and uniform cost search, uniform cost search has better speed advantage. In problem 3, uniform cost search only spent 86 sec compare 169 sec from breath first search, but with more node expansion.

Overall, among those six searches algorithm. A* with ignore precondition has the best run time performance, but in terms of node expansion and new nodes creation, A* with level sum is the best.

- 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))
```

- 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))
```

- 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))
```

Figure 1

		Expansions	Goal Tests	New Nodes	Time elapse	Time
Q1	breath first search	43	56	180	6	0.038504
	depth first search graph search	12	13	48	12	0.010376
	uniform cost search	55	57	224	6	0.04504
	A* H_1	55	57	224	6	0.071688
	A* ignore precondition	41	43	170	6	0.044314
	A* levelsum	11	13	50	6	0.831946

	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)					
Q2	breath first search	3343	4609	30509	9	17.5047
	depth first search graph search	582	583	5211	575	3.92515
	uniform cost search	4853	4855	44041	9	15.06033
	A* H_1	4853	4855	44041	9	17.33593
	A* ignore precondition	1450	1452	13303	9	5.350824
	A* levelsum	86	88	841	9	74.83001
	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)					
Q3	breath first search	14663	18098	129631	12	169.9418
	depth first search graph search	1501	1502	12519	1451	21.6696
	uniform cost search	18234	18236	159707	12	85.80248
	A* H_1	18234	18236	159707	12	67.24391
	A* ignore precondition	5040	5042	44944	12	20.85351
	A* levelsum	318	320	2934	12	413.1922

	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)

Table 1

References

[1] "A* search algorithm," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/A*_search_algorithm.

[2] S. J. Russell and P. Norvig, "classical planning," in *Artificial Intelligence Amodern Approach (3rd edition)*.