Heuristic Analysis

JuHyung Son

In this experiment, I solve Air Cargo transport system problem using planning search. Automated heuristics, A* algorithm is used and compared with uninformed non-heuristic search, BFS, DFS. This analysis is focused on their performance comparison.

Air Cargo Transport system problem

Three problems are carried out. This is the action schema:

```
Action(Load(c, p, a),
        PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
        EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
        PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
        EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
        PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
        EFFECT: ¬ At(p, from) ∧ At(p, to))
```

And These are the three problems:

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))
```

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))
```

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))
```

So the goal is satisfying the problem’s goal with optimal learning time and plan length.

Experiments

Problem1

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time	Optimal
BFS	43	56	180	6	0.0364	T
BFTS	1458	1459	5960	6	0.9841	T
DFGS	12	13	48	12	0.0085	F
Depth limited search	101	271	414	50	0.9794	F
uniform cost search	55	57	224	6	0.3795	T
recursive best first search h_1	4229	4230	17029	6	2.9906	T
greedy best first graph search h_1	7	9	28	6	0.0049	T
a* search h_1	55	57	224	6	0.0461	T
a* h_ignore_preconditions	41	43	170	6	0.0366	T
a* h_pg_levelsum	11	13	50	6	0.9947	T

Problem 2

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time	Optimal
BFS	3346	4612	30534	9	10.173	T
DFGS	1124	1125	10017	1085	8.9764	F

uniform cost search	4779	4781	43370	9	12.842	T
greedy best first graph search h_1	590	592	5310	17	1.5291	T
a* search h_1	4779	4781	43370	9	13.742	T
a* h_ignore_preconditions	1450	1452	13303	9	4.8638	T
a* h_pg_levelsum	86	88	841	9	173.25	T

Problem 3

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time	Optimal
BFS	14663	18098	129631	12	126.22	T
DFGS	627	628	5176	596	4.0525	F
uniform cost search	17710	17712	155337	12	64.605	T
greedy best first graph search h_1	4047	4049	36226	27	14.876	F
a* search h_1	17710	17712	155337	12	65.232	T
a* h_ignore_preconditions	5034	5036	44886	12	21.381	T
a* h_pg_levelsum	320	322	2953	12	968.05	T

Analysis

In non-heuristic algorithm, BFS and uniform cost search looks optimal considering plan length and Time. BFS and uniform cost search show same plan length in all 3 problems, as problem being complicated, uniform cost search has faster execution time. DFGS shows fast execution time, but it has too long plan length which can not be used in real world. Plan length and execution time should be trade off. As a result, as BFS uses less memory than uniform cost search, I recommend to use BFS.
Considering A* search, A* always shows good plan length. However, for A* with three heuristics, each execution time is very different. For example in problem 3, A* h_ignore_preconditions has 21.381 execution time, but A* h_pg_levelsum has 968.05 time, which is very huge. But A* h_pg_levelsum has least memory. The trade off is in memory and execution time. I recommend A* h_ignore_preconditions because It shows fastest execution in all 3 problems and medium memory.
Finally, between BFS and A* h_ignore_preconditions, A* h_ignore_preconditions always shows faster execution time and less memory. My last recommend for search strategy is A* h_ignore_preconditions.

Optimal plans

Problem1	Problem 2	Problem 3
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(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) 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)

Reference

Stuart J. Russell, Peter Norvig(2010), Artificial Intelligence : A Modern Approach