Planning Search Analysis

Jae Min Baek

This project is to implement planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system. With progression search algorithms, optimal plans for each problem has been computed. Unlike the navigation problem, there is no simple distance heuristic to aid the agent. Instead, we implemented domain-independent heuristics.

PDDL(Planning Domain Definitions Language) problems

- **Air Cargo 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))

- **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))

Search Result

We omit theBreadth first tree, Depth Limited, Recursive Best First search result because those search algorithms took more than five minutes.

Below is the code to run search algorithms to each problem.
Python run_search.py -p 1 -s 1 2 3 4 5 6 7 8 9 10
Python run_search.py -p 2 -s 1 3 5 7 8 9 10
Python run_search.py -p 3 -s 1 3 5 7 8 9 10

Actual output is stored in ‘p1_search_result.txt’, ‘p2_search_result.txt’
, ‘p3_search_result.txt’ respectively.

Problem 1 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node Expansions
Breadth first	**6**	0.026	43
Breadth first tree	**6**	0.798	1458
Depth first graph	12	**0.006**	**12**
Depth limited	50	0.075	101
Uniform cost	**6**	0.036	55
Recursive best first	**6**	2.328	4229
Greedy best first	**6**	0.004	7
A* with h1 heuristic	**6**	**0.033**	55
A* with ignore preconditions heuristic	**6**	0.038	41
A* with level sum heuristic	**6**	2.87	**11**

Problem 2 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node

			Expansions
Breadth first	**9**	11.603	3343
Breadth first tree	-	-	-
Depth first graph	575	**2.638**	**582**
Depth limited	-	-	-
Uniform cost	**9**	37.416	4853
Recursive best first	-	-	-
Greedy best first	21	6.115	998
A* with h1 heuristic	**9**	37.602	4853
A* with ignore preconditions heuristic	**9**	**12.104**	1506
A* with level sum heuristic	**9**	69.771	**86**

Problem 3 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node Expansions
Breadth first	**12**	85.214	14663
Breadth first tree	-	-	-
Depth first graph	596	**2.718**	**627**
Depth limited	-	-	-
Uniform cost	**12**	315.329	17783
Recursive best first	-	-	-
Greedy best first	22	55.983	4031
A* with h1 heuristic	**12**	317.333	17783
A* with ignore preconditions heuristic	**12**	**73.070**	5081
A* with level sum heuristic	**12**	450.426	**404**

Analysis

We can break search analysis into two parts: Uninformed Search and Informed Search.

Uninformed Search Analysis

All search algorithms but A* is uninformed search.
Among the uninformed search strategies, **'Breadth First Search'** and **'Uniform Search'** found optimal plan length (6, 9, 12 for plan 1, plan 2, plan3 respectively). And **'Depth First Search'** was fastest and it used the least node expansions.

Informed Search Analysis

A* Algorithms are informed search.
All A* algorithms performed optimal plan. Among the informed search strategies, while **'A* with ignore precondition heuristic'** was the fastest, **'A* with level sum heuristic'** used the least node expansions.

Summary

Among the all strategies which provides optimal plan, we picked 'Breadth First Search' and 'A* with ignore preconditions heuristic' to find the best performance strategy. The result below shows **'A* with ignore preconditions heuristic'** is faster than 'Breadth First' (significantly faster when the optimal length is over 10) and uses less memory.

Problem 1 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node Expansions
Breadth first	6	**0.026**	43
A* with ignore preconditions heuristic	6	0.038	**41**

Problem 2 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node Expansions
Breadth first	9	**11.603**	3343
A* with ignore preconditions heuristic	9	12.104	**1506**

Problem 3 Result

Search Algorithm	Plan Length	Execution Time (seconds)	Node Expansions

Breadth first	12	85.214	14663
A* with ignore preconditions heuristic	12	**73.070**	**5081**

Conclusion

Among the all search strategies, we suggest to choose ‘A* with ignore preconditions heuristic’ for Air Cargo Transport System in the efficiency(optimality), the speed, and the memory usage.