Heuristic Analysis

Project 3 - Implement a planning search

Overview

This project includes skeletons for the classes and functions needed to solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. With progression search algorithms like those in the navigation problem from lecture, optimal plans for each problem will be computed. Unlike the navigation problem, there is no simple distance heuristic to aid the agent.

Planning problems

READ: Stuart Russel and Peter Norvig text:

"Artificial Intelligence: A Modern Approach" 3rd edition chapter 10 *or* 2nd edition Chapter 11 on Planning, available on the AIMA book site sections:

- *The Planning Problem*
- *Planning with State-space Search*

GIVEN: classical PDDL problems

All problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

- *Air Cargo Action Schema:*

2

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))

Solution

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

3

S.N	Method	Expansion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recommended*
1	breadth_first_search	43	56	180	6	0.048424812000121165	True
2	depth_first_graph_search	12	*13*	*48*	*12*	*0.012774283999988256*	*False*
3	uniform_cost_search	55	*57*	*224*	*6*	*0.053854786000101 74*	*True*

- **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))

Solution

4

```
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)
```

S.N	Method	Expansion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recommended*
1	breadth_first_search	3401	4672	31049	9	18.154399800999954	True
2	depth_first_graph_search	17783	*17785*	*155920*	*12*	*65.28177692898316*	*False*
3	uniform_cost_search	4761	*4763*	*43206*	*9*	*15.71866179099993 1*	*True*

- **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))

Solution

```
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)
```

S.N	Method	Expansion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recommended*
1	breadth_first_search	14491	17947	128184	12	133.3775406429997	True
2	depth_first_graph_search	1948	*1949*	*16253*	*1878*	*25.79679587599957 8*	*False*
3	uniform_cost_searc	17783	*17785*	*155920*	*12*	*70.05867157900002*	*True*

From tables given above this can be concluded that “breadth_first_search” and “uniform_cost_search” are the recommended methods for optimal performance for all problems (1,2,3).

For problem 1 breadth_first_search and uniform_cost_search have performed almost similar, the differences are the time taken where breadth_first_search finished fast.

For problem 2, like above both methods performed similar but here some of the difference are uniform_cost_search finishes fast having expansion as 4761 and goal test as 4763 which is little less than uniform_cost_search.

For problem 3, uniform_cost_search performed better than all other methods with plan length *155920 in 70 second.*

Overall DFS works better in all complexity including space and time, according to the author *“In DFS all the branches are removed from the memory once the final traversal completes for the specific node.”*

Domain-independent heuristics

Problem 1:

S.N	Method	Expansion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recommended*
1	astar_search h_1	55	57	224	6	0.060823766999419604	True
2	astar_sear	41	*43*	*170*	*6*	*0.0435327*	*True*

	ch h_ignore_ preconditi ons					*4599998258*	
3	astar_sear ch h_pg_leve lsum	11	*13*	*50*	*6*	*0.732736283999656 7*	*True*

Problem 2 :

S.N	Metho d	Expan sion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recom mende d*
1	astar_sear ch h_1	4761	4763	43206	9	16.109473 56700035	True
2	astar_sear ch h_ignore_ preconditi ons	1450	*1452*	*13303*	*9*	*4.9270837 22000134*	*True*
3	astar_sear ch h_pg_leve lsum	86	*88*	*841*	*9*	*66.021773 41300012*	*True*

Problem 3 :

S.N	Method	Expansion	*Goal tests*	*New nodes*	*Plan length*	*Time*	*Recommended*
1	astar_search h_1	17783	17785	155920	12	75.12439497200012	True
2	astar_search h_ignore_preconditions	5003	*5005*	*44586*	*12*	*23.99084785899958 7*	*True*
3	astar_search h_pg_levelsum	311	*313*	*2863*	*12*	*326.9079867000000 4*	*True*

In the given table above, specifically for the time complexity, the recommended method will be astar_search h_ignore_preconditions, whereas for space complexity better to go with the less node expansion as it reduces the branch traversal (astar_search h_pg_levelsum).

For problem 1, aster_sum method is recommended as this has less time needed to finish. This has the expansion value of 11 and new nodes as 50.

For problem 2 and 3, by the analysis, this is again similar to the problem 1 and it has astar_search h_1 method which is the recommended method in my opinion. It uses 16 seconds with an expansion of 4761 nodes to finish all nodes for problem 2 and for problem 3 it is 75 seconds and 17783 as expansion nodes.