Planning Search Heuristic Analysis

The goal for this project was to solve deterministic planning problem in air cargo domain by using both uninformed non-heurisic search methods and planning graph with automatic domain independent heuristic with A* search. Results from using these methods are compared to each other with focus on optimality of resulting plan and overall algorithm performance.

Problem definition

Problems defined in classical PDDL (Planning Domain Definition Language). All problems are in the Air Cargo domain and they have the same action schema defined, but different initial states and goals:

Action schema

Following actions are defined:

```
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))
```

Problems

There are three problems defined for which we need to find solution:

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))
```

Solutions

Optimal solutions for each problem consist from 6, 9 a 12 steps, respectively. As resulting plans may vary in order of actions applied, following are used only as examples:

Problem 1:

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

Problem 2:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
```

Problem 3:

```
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(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P1, JFK)
Unload(C4, P2, SFO)
```

Uninformed non-heurisic search methods

For purpose of this project, three different search methods were chosen:

- Breadth First Search
- Depth First Search
- Uniform Cost Search

Performance of these algorithms differ in terms of optimality, speed and memory usage. Algoritms are compared in context of each problem.

Problem 1:

Algorithm	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	Yes	6	43	56	180	0.058930109797556786
Depth First Search	No	20	21	22	84	0.02711327422680243
Uniform Cost Search	Yes	6	55	57	224	0.07137066496690671

Problem 2:

Algorithm	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	Yes	9	3343	4609	30509	23.42148120808036
Depth First						

Search	No	619	624	625	5602	5.8961648483140365
Uniform Cost Search	Yes	9	4780	4782	43381	21.356530523376684

Problem 3:

Algorithm	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	Yes	12	14663	18098	129631	179.46278996316866
Depth First Search	No	392	408	409	3364	2.969487478949289
Uniform Cost Search	Yes	12	17882	17884	156769	94.41338577354989

Result analysis

Breadth First Search (BFS) and Uniform Cost Search (UCS) yield optimal results for all problems. If we look at number of expanded nodes and speed of these two algorithms, BFS is slower than UCS but also have a smaller memory footprint. However, both of these were surpassed in terms of speed and memory usage by Depth First Search (DFS) although id didn't yield optimal plan. This difference is even more obvious with increasing complexity of defined problems.

These results were expected, as BFS and UCS are both complete an optimal (in contrast to DFS) algorithms. If branching factor is not very high, we would suggest to use UCS (as speed for problem 3 was much better tahn in BFS). However, if optimal solution isn't required or we are constrained by memory size, DFS should be used.

Informed heurisic search methods

Also in this category, three search methods were used:

- A* Seach + h_1 heuristic
- A* Seach + h_ignore_preconditions heuristic
- A* Seach + h_pg_levelsum heuristic

Heuristics are compared in context of each problem.

Problem 1:

Heuristic	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
h_1	Yes	6	55	57	224	0.07157548822696384
h_ignore_preconditions	Yes	6	41	43	170	0.07219937987709785

h_pg_levelsum	Yes	6	11	13	50	1.2009701248889344

Problem 2:

Heuristic	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
h_1	Yes	9	4780	4782	43381	21.483454172229326
h_ignore_preconditions	Yes	9	1450	1452	13303	8.884503440825945
h_pg_levelsum	Yes	9	86	88	841	128.0399663434419

Problem 3:

Heuristic	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
h_1	Yes	12	17882	17884	156769	96.34561754008084
h_ignore_preconditions	Yes	12	5034	5036	44886	31.334021266389442
h_pg_levelsum	Yes	12	314	316	2894	532.2243838813926

Result analysis

As we can see, search methods differ only in heuristic used (algorithm is always the same). Each heuristic always yields optimal solution for given problem so they differ only in terms of speed and memory footprint. As we can see h_pg_levelsum heuristic is slowest although it uses the smallest amount of memory. Heuristic h_1 uses largest amount of memory and isn't even fastest, so we can rule it out from possible candidates for best heuristics. Overall, we would choose h_ignore_preconditions as the best one, as it is the fastest heuristic and it also doesn't have large memory footprint.

Conclusion

In conclusion, we would consider using either Uniform Cost Search or A* Search with h_ignore_preconditions heuristic as they yield optimal plans. Final comparison for these two is as follows:

Problem 1:

Algorithm	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
A* + h_ignore_preconditions	Yes	6	41	43	170	0.07219937987709785
Uniform Cost Search	Yes	6	55	57	224	0.07137066496690671

Problem 2:

Algorithm	Optimal	Actions	Expansions	Goal	New	Time

				Tests	Nodes	
A* + h_ignore_preconditions	Yes	9	1450	1452	13303	8.884503440825945
Uniform Cost Search	Yes	9	4780	4782	43381	21.356530523376684

Problem 3:

Algorithm	Optimal	Actions	Expansions	Goal Tests	New Nodes	Time
A* + h_ignore_preconditions	Yes	12	5034	5036	44886	31.334021266389442
Uniform Cost Search	Yes	12	17882	17884	156769	94.41338577354989

As we can see, A* + h_ignore_preconditions outperforms UCS both in terms of speed and memory footprint. These results empathize the benefits of using informed search strategies in comparison to uninformed search strategies.