Planning Search Heuristic Analysis

Planning Problem

We were given three planning problems in the Air Cargo domain that use the same 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))
```

The three problems have the following initial states and goals:

Problem 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))
```

Problem 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))
```

Problem 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))
```

The goals above can be reached using different plans, but the optimal plan lengths for problems 1,2, and 3 are 6, 9, and 12 actions. The following plan examples are those with an optimal length.

Problem1 :

Load(c1,p1,SFO)
Load(c2,p2,JFK)
Fly(p1,SFO,JFK)
Fly(p2,JFK,SFO)
Unload(c1,p1,JFK)
Unload(c2,p2,SFO)

Problem2 :

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(c1,p1,JFK)
Unload(c2,p2,SFO)
Unload(c3,p3,SFO)

Problem3 :

Load(c1,p1,SFO)
Load(c2,p2,JFK)
Fly(p1,SFO,JFK)
Load(c3,p1,ATL)
Fly(p2,JFK,SFO)
Load(c4,p2,ORD)
Fly(p1,ATL,JFK)
Fly(p2,ORD,SFO)
Unload(c1,p1,JFK)
Unload(c2,p2,SFO)
Unload(c3,p1,JFK)
Unload(c4,p2,SFO)

Uninformed Search Strategies Analysis

Search strategies that come under the heading of uninformed search have no additional information about states beyond that provided in the problem definition. All they can do is generate successors In this section, we compare the performance of seven such strategies in terms of speed (execution ti me, measured in seconds), memory usage (measured in search node expansions) and optimality (Yes , if a solution of optimal length is found; No, otherwise). Performance measures were collected using the following commands:

python run_search.py -p 1 -s 1 2 3 4 5 6 7

python run_search.py -p 2 -s 1 3 5 7

python run_search.py -p 3 -s 1 3 5 7

Problem1 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
Breadth first search	yes	6	0.09	43	56	180
Breadth first tree search	yes	6	2.75	1458	1459	5960
Depth first graph search	no	12	0.03	12	13	48
Depth limited search	no	50	0.28	101	271	414
Uniform cost search	yes	6	0.10	55	57	224
Recursive best first search	yes	6	7.94	4229	4230	17029
Greedy best first graph search	yes	6	0.01	7	9	28

Problem2 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
Breadth first search	yes	9	60.06	3109	4672	31049
Breadth first tree search	---	---	---	---	---	---
Depth first graph search	no	346	7.06	350	358	2514

Depth limited search	---	---	---	---	---	---
Uniform cost search	yes	9	43.59	4834	4836	43866
Recursive best first search	---	---	---	---	---	---
Greedy best first graph search	yes	9	10.03	950	952	8550

Problem3 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
Breadth first search	yes	12	335.98	14491	17947	128184
Breadth first tree search	---	---	---	---	---	---
Depth first graph search						
Depth limited search	---	---	---	---	---	---
Uniform cost search	yes	12	206.77	17513	17515	153613
Recursive best first search	---	---	---	---	---	---
Greedy best first graph search	no	27	30.52	4404	4406	38847

Analysis

With this 3 problems , Breadth First Search and Uniform Cost Search are the only two uninformed search strategies that yield an optimal action plan under the 7min time limit. When it comes to execution speed and memory usage, Depth First Graph Search uses the least memory. However, it does not generate an optimal action plan.

Informed (Heuristic) Search Strategies Analysis

In this section, we compare the performance of A* Search using three different heuristics. Here again , we evaluate these strategies in terms of speed, memory usage and optimality.
Performance measures were collected using the following commands:

python run_search.py -p 1 -s 8 9 10

python run_search.py -p 2 -s 8 9 10

python run_search.py -p 3 -s 8 9

Problem1 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	6	0.222	55	57	224
A*Search with ignore preconditions heuristic	yes	6	0.119	41	43	170
A*Search with level Sum heuristic	yes	6	2.655	55	57	224

Problem2 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	9	214.32	4834	4836	48866
A*Search with ignore preconditions heuristic	yes	9	45.28	1506	1508	13820
A*Search with level Sum heuristic	yes	9	1692.50	4834	4836	43866

Problem3 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	12	2177.24	17513	17515	153613
A*Search with ignore preconditions heuristic	yes	12	558.05	5102	5104	45488
A*Search with level Sum heuristic						

Analysis

While all heuristic yield an optimal action plan, only the h1 and ignore Preconditions heuristic return ressults within the 10mn execution time set by the Udacity. Of the two strategies mentioned above, the Time elapsed of A*Search with ignore preconditions heuristic is very big. If we let search run to completion on our computer, A*Search with level Sum heuristic uses the least memory, but its Time elapsed is much slower.

Informed vs Uninformed Search Strategies

The search strategies that generate optimal plans are Breadth First Search, Uniform Cost Search, and A* Search with all three heuristics.

As we saw earlier, when it comes to execution speed and memory usage of uninformed search strategies, **Depth First Graph Search** is faster and uses less memory than Uniform Cost Search. As for informed search strategies, **A* Search with Ignore Preconditions heuristic** is the fastest and uses the least memory. So, really, the choice is between Depth First Graph Search and A* Search with Ignore Preconditions heuristic. Here we compare their results against our 3-problem set

Problem1 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	6	0.222	55	57	224
A*Search with ignore preconditions heuristic	yes	6	0.119	41	43	170

Problem2 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	9	214.32	4834	4836	48866
A*Search with ignore preconditions heuristic	yes	9	45.28	1506	1508	13820

Problem3 result

Search strategy	Optimal	Plan length	Time elapsed	Node expansion	Goal tests	New node
A*Search with h1 heuristic	yes	12	2177.24	17513	17515	153613
A*Search with ignore preconditions heuristic	yes	12	558.05	5102	5104	45488

From the results above, because it is faster and uses less memory, **A* Search with Ignore Preconditions heuristic** would be the best choice overall for our Air Cargo problem

Conclusion

The above results clearly illustrate the benefits of using well-researched search strategies with customized heuristics on poorly-informed search techniques when searching for an optimal plan. The advantages are significant both in terms of speed and memory utilization.