Udacity AIND Project: Implementing a Planning Search (Heuristic Analysis)

Problem Definition and Results

Problem 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

```
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))
```

Optimal Plan

`Load(C1, P1, SFO)`

`Load(C2, P2, JFK)`

`Fly(P1, SFO, JFK)`

`Fly(P2, JFK, SFO)`

`Unload(C1, P1, JFK)`

`Unload(C2, P2, SFO)`

Result Metrics

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
Breadth First Search	Yes	6	0.036	43
Depth First Graph Search	No	20	0.019	21
Greedy Best First Search	Yes	6	0.008	7

For this simple problem, we see that Greedy Best First Search performs best. It gives optimum results in minimum time, while consuming the least amount of memory (nodes expanded). Breadth First Search yields optimal results, but takes more time and consumes more memory than Greedy Best First Search. Depth First Search, does not yield an optimal result, but it runs in a short time and consumes less memory compared to Breadth First Search.

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
A* with Ignore Preconditions Heuristic	Yes	6	0.048	41
A* with Level Sum Heuristic	Yes	6	0.880	11

It is quite evident from the results that, the heuristics search yields better results compared to the uninformed non-heuristic search. Among the two heuristic search methods, Level Sum Heuristic takes a little more time to complete, but consumes less memory. Both heuristics result in optimal results.

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))
```

Optimal Plan

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

Result Metrics

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
Breadth First Search	Yes	9	16.137	3343
Depth First Graph Search	No	619	4.208	619
Greedy Best First Search	No	21	2.565	998

For this problem, we notice that neither Depth First Graph Search or Greedy Best First Search yields optimal results, even though they run quickly and consume less memory. The path length of the solution output by Depth First Search is however very large compared to the path length of the optimal solution. We observe that like problem #1, Breadth First Search does

arrive at an optimal solution. The number of nodes expanded by Breadth First Search however, is very large compared to the other two search methods.

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
A* with Ignore Preconditions Heuristic	Yes	9	5.264	1450
A* with Level Sum Heuristic	Yes	9	70.322	86

In this problem, we observe that the Level Sum Heuristic runs significantly slower compared to the Ignore Preconditions Heuristic. The Level Sum Heuristic also consumes less memory compared to the Ignore Preconditions Heuristic. Both heuristics converge at optimal results.

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))
```

Optimal Plan

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

Result Metrics

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
Breadth First Search	Yes	12	121.920	14663
Depth First Graph Search	No	392	1.820	408
Greedy Best First Search	No	16	18.578	5580

For this problem, we see that Breadth First Search yet again converges at an optimal solution, albeit at the cost of time and memory. The difference in the computational time required and memory consumed for Breadth First Search is very apparent for this problem. Depth First Graph Search, runs extremely fast and converges at a non-optimal solution, whose path length is

way off the path length of the optimal solution. Greedy Best First Search consumes less computational time and memory, but arrives at a non-optimal solution. This solution however doesn't differ a lot in path length with the optimal solution.

Search Method	Optimality	Plan Length	Time Elapsed (seconds)	#Nodes Expanded
A* with Ignore Preconditions Heuristic	Yes	12	18.755	5040
A* with Level Sum Heuristic	Yes	12	361.150	315

The result for this problem is also like the above problems. The Level Sum Heuristic runs much slower compared to the Ignore Preconditions Heuristic while consuming lesser memory. The heuristics also arrive at the optimal solution, like in previous problems.

Conclusion

We can conclude from the above results that A-star Search with Ignore Preconditions Heuristic works best in terms of time taken and optimality. The heuristic expands more nodes and consumes more memory than the Level Sum Heuristic. So, in situations when memory is scarce compared to computational time, we must prefer A-star Search with the Level Sum Heuristic.

The heuristic based search provides optimal results in reasonable time for hard problems. For simpler problems having fewer literals, the uninformed non-heuristic based search performs best. If we want to guarantee optimality, we should prefer Breadth First Search for such scenarios. If speed is of more importance, we can then go with the Greedy Best First Search.