2018

Heuristic Analysis

(Implement a planning search project)

ARTIFICIAL INTELLIGENCE NANO DEGREE, UDACITY

Problem Definition

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:

```
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))
```

Given this schema, provide an optimal plan for Problems 1, 2, and 3.

Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.

Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

Provide tables or other visual aids as needed for clarity in your discussion.

Problem number 1

Initial state & 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))
```

Optimal plan (length=6):

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

Results obtained:

Search method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
Breadth first search	43	6	0.0428	Yes
Depth first graph search	21	20	0.0188	No
Greedy best first search	7	6	0.0073	Yes

For this first problem, it can be seen that the *greedy best first search* performs best in both takin less time and obtaining an optimal solution, at the same time it consumes the least amount of memory (nodes expanded). In the case of the *breadth first search* it obtains the optimal result, although it takes more time and consumes more memory than the case of the *greedy best first search*. In the case of the *depth first search*, it doesn't obtain the optimal result, although it requires less memory than the *breadth first search* and takes less time to obtain the result.

Search method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
A* search with ignore preconditions	41	6	0.05088	Yes
A* search with level sum	11	6	0.9000	Yes

It can be clearly seen in the results that heuristics search obtains better results than non-heuristic search.

When comparing both methods for the heuristic search, *search with level sum* takes some more time, although it uses less memory, resulting both cases in the optimal solution.

Problem number 2

Initial state & 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))
```

Optimal plan (length=9):

```
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)
```

Results obtained:

Search method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
Breadth first search	3343	9	16.1973	Yes
Depth first graph search	624	619	4.8855	No
Greedy best first search	990	17	5.1360	No

For this second problem, it can be seen that the *breadth first search* is the only one to reach an optimal result, although in comparison to the *depth first graph search* and the *greedy best first search*, these last two use up far less memory and take a far less amount of time to reach the result. In the case of the *depth first graph search* the path length is very large compared to the other two option.

Like in problem number 1, the *breadth first search* case obtains an optimal result, although the number of nodes expanded is very large compared to the other two methods.

Search method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
A* search with ignore preconditions	1450	9	7.8313	Yes
A* search with level sum	86	9	80.8422	Yes

Regarding the heuristic search, in this case it can be observed that the *search with level sum* takes a significantly greater amount of time in comparison with the *search with ignore preconditions*. The *search with level sum* also consumes less memory compared to the other method.

In this problem, both methods achieve an optimal result.

Problem number 3

Initial state & 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))
```

Optimal plan (length=12):

```
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)
```

Results obtained:

Search Method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
Breadth first search	14663	12	98.6335	Yes
Depth first graph search	408	392	3.0904	No
Greedy best first search	5614	22	37.9533	No

For this third problem, it can be seen that the *breadth first search* obtains once again an optimal result, although as in problem 2, it requires both a greater use of memory than the other two methods and takes a significantly greater amount of time (33 times the time required for *depth first graph search* and 3 times the time required for *greedy best first search*).

On the other hand, the *depth first graph search* takes a significantly low amount of time and uses up the smallest amount of memory from all three methods, but it doesn't achieve an optimal result. The *greedy best first search* lays somewhere in between the other two methods, using up less memory than the *breadth first search* and reaching a result closer to the optimal than the *depth first graph search*.

Search Method	Nodes expanded	Plan Length	Time elapsed (seconds)	Optimal result
A* search with ignore preconditions	5040	12	36.4486	Yes
A* search with level sum	318	12	392.8886	Yes

Regarding the heuristic search for this third problem, the result is similar to the previous cases. The *search with level sum* takes a significantly greater amount of time than the *search with ignore preconditions*, although it uses up a significantly smaller amount of memory.

Like in the previous cases, both methods achieve an optimal result.

Conclusion

Based on the previous results, it can be concluded that the *a-star search with ignore preconditions* obtains the best results from all methods in terms of time taken and optimal results, although it uses up more memory than the *search with level sum* method. Therefore, in cases where memory is indeed a priority over processing time, the *search with level sum* option would be the preferred choice.

The heuristic based searches provide optimal results in a reasonable amount of time, although for simpler problems with fewer literals, the *non-heuristic* methods have a better performance (case of the *greedy best first search* in problem 1). In the case of using such methods instead of the heuristic ones, the *breadth first search* method would be the one to choose in terms of reaching optimal results and the *greedy best first search* for cases similar to problem 1 in order to improve processing time.