Search Performance Comparison

> An analysis of the performance of search algorithms solving cargo problems

This is part of the Udacity nanodegree for Artificial Intelligence. The complete assignment can be found in the README to this repository.

This analysis covers three different problems that are solved with a variety of algorithms.

Problem Space

All three problems are in the same domain of cargo logistics at airports. The underlying Action Schema is the following

```
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))
```

where the problems are defined as

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))
```

Non-heuristic Searches

Used Algorithms

The following algorithms are used to solve the above problems.

1. Breadth First Search (BFS)
2. Depth First Graph Search (DFS)
3. Depth Limited Search (DLS)
4. Uniform Cost Search (UCS)
5. Greedy Best First Search (GBFS)

Comparison

The complete combination of problems and algorithms can be run with python run_search.py -p 1 2 3 -s 1 3 4 5 7

	Goal	Plan

Problem	Algorithm	Expansions	Tests	Nodes	Length
1	BFS	43	56	180	6
1	DFS	12	13	48	12
1	DLS	103	271	414	50
1	UCS	55	57	224	6
1	GBFS	7	9	28	6
2	BFS	3,343	4,609	30,509	9
2	DFS	582	583	5211	575
2	DLS	222,719	2,053,741	205,4119	50
2	UCS	4,852	4,854	44,030	9
2	GBFS	990	992	8,910	15
3	BFS	14,663	18,098	129,631	12
3	DFS	627	628	5,176	596
3	DLS	-	-	-	-
3	UCS	18,164	18,166	1591,47	12
3	GBFS	5,399	5,401	47,691	17

Observations

One thing that clearly shows in the comparison table is that algorithms that only run a small number of expansions and therefore cover smaller number of nodes lead to longer paths. This is expected because without a proper heuristic the algorithm would need to cover as much solution space as possible to find the best answer.

Those algorithms that yield the best results have a high number of

iterated nodes. Breadth First and Uniform Cost Search seem to perform benchmark when looking at the path length. Depth First seems to deliver the fastest solution but in complex scenarios like Cargo 2 & 3 it delivers a really costly solution. Greedy Best First seems to provide a good tradeoff between solution speed and the cost of the calculated solution.

To my surprise the Depth Limited Search performs really bad in both categories. It takes longer than other algorithms to find a solution and the solution has one of the longest paths.

Conclusion

With the observations made from this comparison the following conclusions can be made:

1. For scenarios where the search duration needs to be minimized the Greedy Best First Algorithm delivers solutions quickly, however not the best solution
2. If the application allows for more time during the search the algorithms Breadth First or Uniform Cost Search deliver the solution with the lowest cost (shortest path).

Heuristic Searches (A*)

In this chapter multiple heuristics for the A-Star algorithm will be benchmarked on the three Cargo Problems.

Comparison

The complete combination of problems and algorithms can be run with python run_search.py -p 1 2 3 -s 8 9 10

Problem	Heuristic	Expansions	Goal Tests	Nodes	P L
1	H1	55	57	224	6

1	h_ignore_preconditions	39	41	150	6
1	h_pg_levelsum	11	13	50	6
2	H1	4852	4854	44030	9
2	h_ignore_preconditions	3348	3350	29614	9
2	h_pg_levelsum	86	88	841	9
3	H1	18164	18166	159147	1
3	h_ignore_preconditions	12532	12534	106226	1
3	h_pg_levelsum	316	318	2913	1

Observations

Heuristic algorithms take a lot longer to come up with the solution in this problem space. This might be due to the nature of heuristic approaches as they require additional computing time to calculate the heuristic score in each iteration. Although comparing the increase in search duration shows that different heuristics have a different effect. They all find a solution with the same path length though.

Heuristic	**Problem1: Multiples of BFS duration (absolute Duration)**	**Problem2: Multiples of BFS duration (absolute Duration)**	**Problem3: Multiples of BFS duration (absolute Duration)**
H1	1.6 (0.05 s)	1.6 (15.9 s)	1.2 (59.8 s)
h_ignore_preconditions	3.3 (0.1 s)	2.9 (28.1 s)	2.5 (127.0 s)
h_pg_levelsum	46.6 (1.4 s)	28.8 (281.9 s)	27.3 (1363.3 s)

Conclusion

Different heuristics have a big impact on the number of expansions but dominantly on the duration of the planning operation. The above table however shows that with increasing problem complexity the heuristic approach gets closer to the best time reachd by a non-heuristic algorith (Breadth First Algorithm).

The best heuristic in this case was the H1 followed by Ignore Preconditions. They takae significantly longer than their non-heuristic counterparts in finding an optimal solution though.

Comparing heuristic and non-heuristic approaches

Heuristic approaches require more computing power to find a solution. However they do not necessarily take more node expansions to reach the goal state. This property is useful in real world scenarios where computing power but the outcome of an action might be unknown unless tested out. In such a scenario the agent would take the costly step in the real world, monitor the result and do the heuristic calculations depending on the outcome.

Optimal solutions

Problem 1: 6 steps

Generated with Breadth First Search.

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

Problem 2: 9 steps

Generated with Uniform Cost Search

```
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)
```

Problem 3: 12 steps

Generated with A-Star (h_pg_levelsum heuristic)

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