Carl Hinkle
AIND - December 2017 Cohort
Project 3 Report

Introduction

This project compared several different implementations of planning search. Each algorithm was executed on each of the three planning problems. An optimal route was discovered for each. The algorithms are are as follows, and the results are presented in tabular form at the end of the report.

1. Breadth First
2. Depth First Graph Search
3. Uniform Cost Search
4. A* h_1
5. A* h_ignore_preconditions
6. A* h_pg_levelsum

Optimal Routes

Problem 1 - Optimal Length = 6

```
Goal = [
    expr('At(C1, JFK)'),
    expr('At(C2, SFO)'),
]

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

Problem 2 - Optimal Length = 9

```
Goal =  [
    expr('At(C1, JFK)'),
    expr('At(C2, SFO)'),
    expr('At(C3, SFO)')
]

Path =
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)

Problem 3 - Optimal Length = 12

```
Goal = [
    expr('At(C1, JFK)'),
    expr('At(C3, JFK)'),
    expr('At(C2, SFO)'),
    expr('At(C4, SFO)')
]
```

Path =
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(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Discussion of Optimality

There are several ways that we can define optimality in this project, and there are tradeoffs that can be considered when deciding which algorithm to choose depending on what criteria is being optimized for.

The fastest algorithm in all cases is Depth First, but that is clearly not optimal unless you are strictly optimizing for average execution speed (unless there's a loop!). It can be seen that Depth First never produces an optimal plan length (although there may be cases where it does produce an optimal result, there are also cases where the algorithm will never complete), so unless you're planning on taking the scenic route on a trip that may never end, it's best to avoid DFS.

Breadth first and Uniform Cost (two other non-heuristic algorithms) both produce optimal plan lengths, with Breadth first performing a bit better on Node Expansions, goal tests, and new nodes. BFS also edges out Uniform Cost in execution speed.

From these results, it seems that in most (if not all) cases, BFS is the desirable non-heuristic based search algorithm to use.

Three heuristic based algorithms were also compared. The first A* h_1 is the same as Uniform Cost search and will not be discussed further (results are in the tables). A* h_ignore_pre produced an optimal plan length, with fewer node expansions, goal tests, and new nodes than any of the non-heuristic searches (DFS excluded as discussed above. DFS is also excluded in all further discussion regarding optimality). It also ran faster than all of the non-heuristic searches.

This leaves only A* h_pg_levelsum, which used the levelsum algorithm, making use of a Planning Graph, to perform its heuristic analysis. This heuristic outperformed all of the others by one to two orders of magnitude on all of the metrics that we are interested in, except run time. In this category h_pg_levelsum underperformed dramatically. I question whether this performance is inherent in the algorithm (there are many steps and it's not surprising that it runs slower than the others), or in my implementation. I think that the node equality tests, which create new throwaway nodes simply to test for equivalence, take a big hit out of the overall runtime performance.

Given how much better all of the non-time based metrics are with h_pg_levelsum, if it was within a reasonable margin of the second place finisher (h_ignore_pre) I would be comfortable using it. But given how much faster h_ignore_pre performed, while still performing well on all of the other metrics, I have to recommend preferring h_ignore_pre in most cases.

The only exception would be the special case of having enough foreknowledge of the problem such that you could guarantee that DFS would not loop, and you didn't care about the length of the path. In this case DFS would be a reasonable approach.

However, I think that with optimizations, h_pg_levelsum could be made the best in all cases.

Data

Problem 1	Node Expansions	Number of Goal Tests	New Nodes	Time Elapsed	Plan Length	Optimality
Breadth First	43	56	180	0.037682	6	Length
Depth First Graph Search	21	22	84	0.016657	20	Time
Uniform Cost Search	55	57	224	0.047604	6	Length
A* h_1	55	57	224	0.045466	6	Length
A* h_ignore_pre	41	43	170	0.055859	6	Length
A* h_pg_levelsum	11	13	50	0.835722	6	Expansions Tests New Nodes Length

Problem 2	Node Expansions	Number of Goal Tests	New Nodes	Time Elapsed	Plan Length	Optimality
Breadth First	3346	4612	30534	9.930703	9	Length
Depth First	107	108	959	0.376721	105	Time
Uniform Cost Search	4853	4855	44041	15.2941	9	Length
A* h_1	4853	4855	44041	14.7994	9	Length
A* h_ignore_pre	1450	1452	13303	7.25323	9	Length
A* h_pg_levelsum	86	88	841	84.248286	9	Expansions Tests New Nodes Length

Problem 3	Node Expansions	Number of Goal Tests	New Nodes	Time Elapsed	Plan Length	Optimality
Breadth First	14120	17673	124926	46.8461256	12	Length
Depth First	292	293	2388	1.30293	288	Time Expansions Tests
Uniform Cost Search	18223	18225	159618	61.008843	12	Length
A* h_1	18223	18225	159618	66.27832	12	Length
A* h_ignore_pre	5040	5042	44944	27.843063	12	Length
A* h_pg_levelsum	316	318	2912	403.0281511	12	Length