Show Code

AIND 2018 Term 1 Carl Wennstam

Planning Project Report

- Introduction
  - State space
  - Step cost
  - Search Algorithms
  - Heuristics
- Solutions
  - Uninformed (blind) search
  - Informed search
- Complexity
  - Uninformed
  - Informed
- Appendix

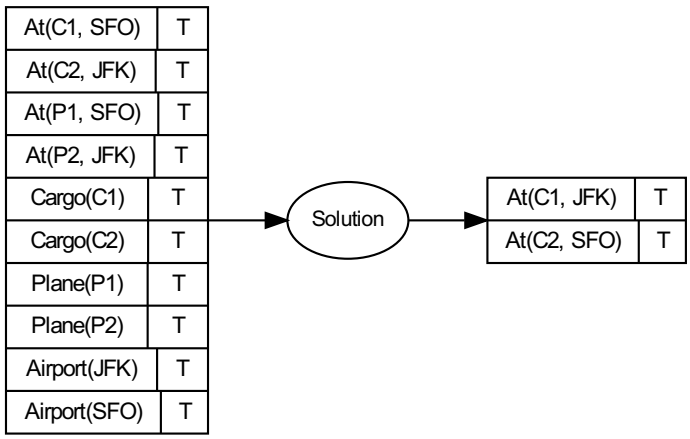
Out[15]:

```
<module 'utils' from 'c:\\Users\\Administrator\\Documents\\GitHub\\AIND-Planning\\utils.py'>
```

Introduction

State space

Problem 1 has a total off 10 initial fluent states, for a state space of $2^{12} = 4,096$. The goal state contains only 2 fluents, meaning the others are Don't Care.



Problems 2 and 3 are gradually larger, with state spaces of $2^{27} = 134,217,728$ and $2^{32} = 4,294,967,296$ respectively.

Step cost

A **solution** to a **planning problem** is a sequence of actions that leads from the initial state to the goal state.

An **optimal** solution is the solution with the **lowest path cost**, or

$$ S_{optimal} \equiv \arg \min_{S} \left(pathcost(S) \right) = \arg \min_{S} \left(length(S) \right) $$

, since we are assuming **unit step cost**.

Solutions that are not optimal are sometimes referred to as **satisficing.**

Search algorithms

Five uninformed search algorithms were tested.

- **Breadth First Search (BFS)** expands the frontier/priority queue/open set and picks the latest added node. With a unit

step cost like in this exercise, this will yeal the shortest path and optimal solution. Time and space complexity both high, at $O(b^d)$
- **Uniform Cost Search (UCS)** stays optimal also for non-zero step costs. On the other hand, with unit step cost it will lag **BFS** by one expansion.
- **Breadth First Tree Search (BFTS)** differs from standard **BFS** in that it does not consider where a node has been visited before, so prone to get stuck in redundant paths.
- **Depth First Graph Search (DFGS)** is non-optimal since, as it will choose the longest path first it is *guaranteed* to return a longer solution when such a solution exists.
- **Depth Limited Search (DLS)**, less vulnerable to redundant paths.

We also have three informed algorithms.

- **Greedy Best First Graph Search (GBFGS)** will go straight for the heuristic-maximing node in the frontier. But this is not neccesarily the optimal solution.
- **A-Star (A*)** fixes problems with **Greedy Best First** by supplementing the heuristic $h(n)$, which can be thought of as the expected future cost, with $g(n)$, the path cost seen so far. Keeps all previously generated nodes in memory.
- **Recursive Best First Search (RBFS)** is similar to A*, but only keeps the current search path and the sibling nodes along the path, achieveing lower space complexity (linear) at the cost of high time complexity.

Heuristics

- **H1** is not a true heuristic as it always returns 1.
- **Level sum (HPL)** returns the path cost up to this point.
- **Ignore Preconditions (HIP)** relaxes the problem to a simpler one, and generates an optimistic heuristic.

Solution quality

Uninformed (blind) search

Problem 1

For Problem 1, all **uninformed** search algorithms were able to find a solution.

Out[79]:

			length	optimal
problem	**algorithm**	**algo**		
1	**Breadth First Search**	**BFS**	6	True
	Uniform Cost Search	**UCS**	6	True
	Breadth First Tree Search	**BFTS**	6	True
	Depth First Graph Search	**DFGS**	20	False
	Depth Limited Search	**DLS**	50	False

- **BFS** found a solution of path cost = 6, which is an optimal solution. (Again, we know this since we know that **BFS** *always* yields an optimal solution.)
- Apart from **BFS**, we also found optimal solutions with **UCS** and **Breadth First Tree Search (BFTS)**, whereas **Depth First Graph Search (DFGS)** and **Depth Limited Search (DFS)** returned suboptimal solutions, having path cost > 6 as expected.

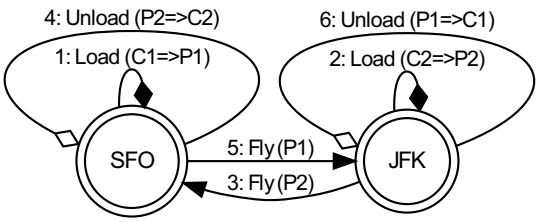
Below are the action sequences of the optimal solutions.

Out[19]:

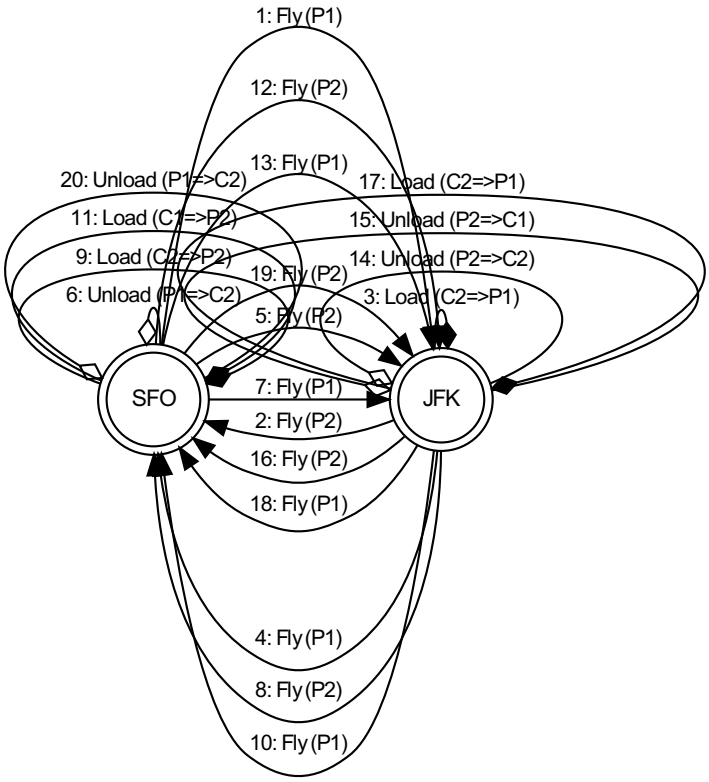
	0	1	2	3	4	5
BFS	Load(C1, P1, SFO)	Load(C2, P2, JFK)	Fly(P2, JFK, SFO)	Unload(C2, P2, SFO)	Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)
BFTS	Load(C1, P1, SFO)	Load(C2, P2, JFK)	Fly(P2, JFK, SFO)	Unload(C2, P2, SFO)	Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)
UCS	Load(C1, P1, SFO)	Load(C2, P2, JFK)	Fly(P1, SFO, JFK)	Fly(P2, JFK, SFO)	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)

The **BFS/BFTS** solution is depicted graphically below, implemented in dot language.

- *Load(P,C,A)* actions are shown as clockwise loop on A ending in a filled diamond.
- *Unload(P,C,A)* actions are shown as counter-clockwise loop on A ending in an unfilled diamond.



THe **UCS** solution is also optimal, with the same length but differing in the order of the actions: it flies both planes and then unloads both cargoes in parallel, whereas **BFS/BFTS** chose to finish one Load-Fly-Unload **high-level action** before starting the next. Now to compare this to the next best suboptimal solution found with **DFGS**.



Problem 2

With Problem 2, some of the blind search algorithm starts to struggle and only **BFS**, **UCS**, **DFGS** finished within the stated 10 min guideline. The optimal solution path cost is 9. Refer to Appendix for details on the action sequences of the optimal solution.

Out[22]:

			length	optimal
problem	**algorithm**	**algo**		
2	**Breadth First Search**	**BFS**	9	True
	Uniform Cost Search	**UCS**	9	True
	Depth First Graph Search	**DFGS**	619	False

Problem 3

Similar to Problem 2, as optimal solution path cost increased to 12.

Out[20]:

			length	optimal
problem	**algorithm**	**algo**		
3	**Breadth First Search**	**BFS**	12	True
	Uniform Cost Search	**UCS**	12	True
	Depth First Graph Search	**DFGS**	392	False

Informed search

We now look at our informed search algorithms.

Problem 1

Out[104]:

			length	optimal
problem	**algo**	**heuristic**		
1	**RBFS**	**H1**	6	True
	GBFGS	**H1**	6	True
	AS	**HPL**	6	True
		HIP	6	True
		H1	6	True

All combinations found an optimal solution.

Out[21]:

	0	1	2	3	4	5
AS	Load(C1, P1, SFO)	Fly(P1, SFO, JFK)	Load(C2, P2, JFK)	Fly(P2, JFK, SFO)	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)
GBFGS	Load(C1, P1, SFO)	Load(C2, P2, JFK)	Fly(P1, SFO, JFK)	Fly(P2, JFK, SFO)	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)
RBFS	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

Out[105]:

			length	optimal
problem	**algo**	**heuristic**		
2	**GBFGS**	**H1**	17	False
	AS	**HPL**	9	True
		HIP	9	True
		H1	9	True

This time **Greedy Best First Graph Search** did not yield an optimal solution

Problem 3

Out[23]:

			length	optimal

problem	algo	heuristic	length	optimal
problem	algo	heuristic		
3	GBFGS	H1	22	False
	AS	HPL	12	True
		HIP	12	True
		H1	12	True

Complexity

Uninformed search algorithms

Problem 1

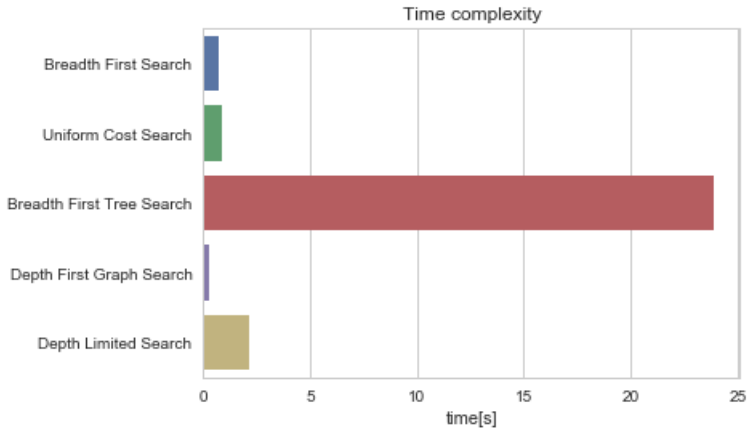
Out[36]:

			created	expansions	length	tests	time[s]	optimal
problem	algorithm	algo						
1	Breadth First Search	BFS	180	43	6	56	0.7	True
	Uniform Cost Search	UCS	224	55	6	57	0.9	True
	Breadth First Tree Search	BFTS	5960	1458	6	1459	23.9	True
	Depth First Graph Search	DFGS	84	21	20	22	0.3	False
	Depth Limited Search	DLS	414	101	50	271	2.2	False

Time complexity

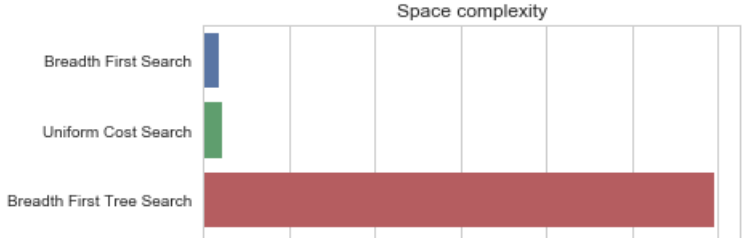
BFS takes the shortest time among the optimal algorithms. **UCS** has similar performance but makes exactly 1 more test, as it tests on removing nodes rather than on adding them. **BFTS** shows horrible results, suffering from many redundant paths, but still ends up finding an optimal path.

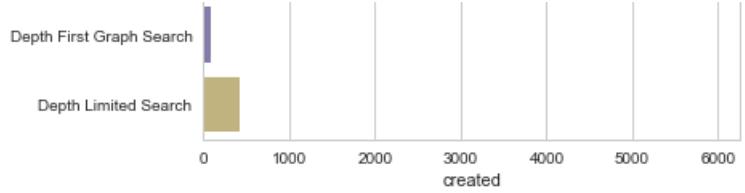
There is a strong linear relationship between **expansions** and **execution time**.



Space complexity

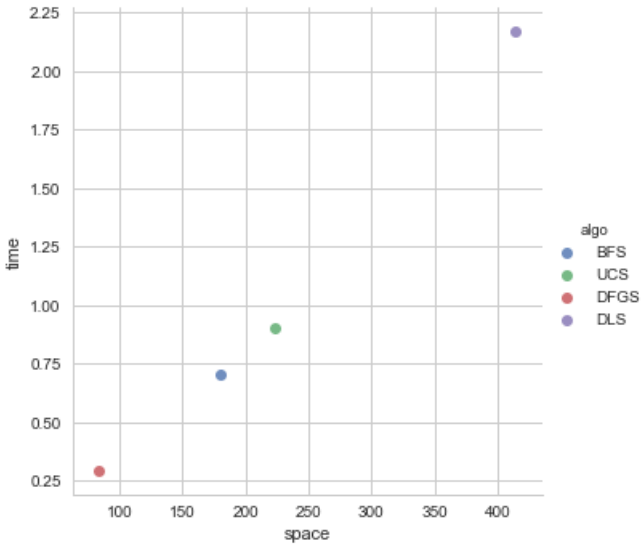
Total number of nodes created. Several of the algorithms, including **BFS** are forced to keep all these nodes in memory throughout the search.





Space vs Time

Below is space vs time comppexity for the different algorithms (BFTS excluded).



Performance summary for all three problems:

Out[50]:

			created	expansions	length	tests	time[s]	optimal
problem	**algorithm**	**algo**						
1	**Breadth First Search**	**BFS**	180	43	6	56	0.7	True
	Breadth First Tree Search	**BFTS**	5960	1458	6	1459	23.9	True
	Uniform Cost Search	**UCS**	224	55	6	57	0.9	True
	Depth First Graph Search	**DFGS**	84	21	20	22	0.3	False
	Depth Limited Search	**DLS**	414	101	50	271	2.2	False
2	**Breadth First Search**	**BFS**	30509	3343	9	4609	21.7	True
	Uniform Cost Search	**UCS**	44030	4852	9	4854	29.3	True
	Depth First Graph Search	**DFGS**	5602	624	619	625	6.2	False
3	**Breadth First Search**	**BFS**	129631	14663	12	18098	2612.8	True
	Uniform Cost Search	**UCS**	159716	18235	12	18237	2201.7	True
	Depth First Graph Search	**DFGS**	3364	408	392	409	85.9	False

Informed search algorithms

Next we have the *informed* search algorithms.

Out[61]:

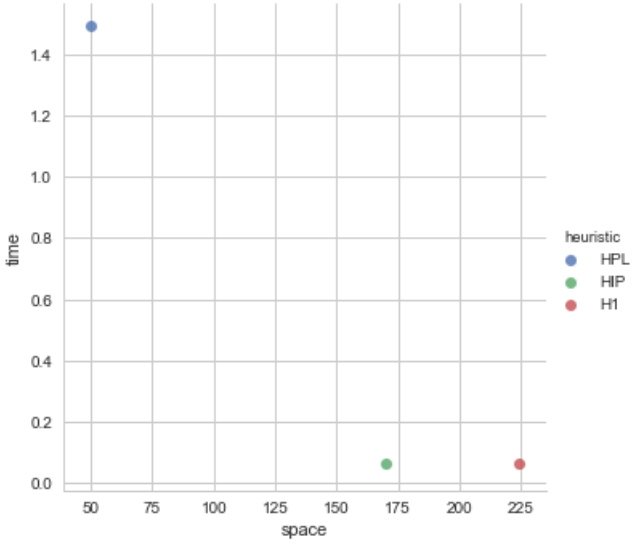
			created	expansions	length	tests	time[s]	optimal
problem	**algo**	**heuristic**						
1	**GBFGS**	**H1**	28	7	6	9	0.0	True

	AS	HPL	50	11	6	13	1.5	True
problem	algo	HIP	170	41	6	43	0.1	True
		H1	224	55	6	57	0.1	True
	RBFS	H1	17023	4229	6	4230	4.9	True

Time complexity

GBFGS is by far the fastest here and still finds an optimal solution. **RBFS** seem completely unusable for this problem.

A* is more interesting as we can see an inverse relationship between time and space complexity.



With the **H1** constant heuristic as the benchmark, we can see the incemental improvements to space complexity from the use of a proper heuristic **HPL** (levelsum)

Out[63]:

			created	expansions	length	tests	time[s]	optimal
problem	algo	heuristic						
1	RBFS	H1	17023	4229	6	4230	4.9	True
	GBFGS	H1	28	7	6	9	0.0	True
	AS	H1	224	55	6	57	0.1	True
		HIP	170	41	6	43	0.1	True
		HPL	50	11	6	13	1.5	True
2	AS	H1	44030	4852	9	4854	29.5	True
		HIP	13303	1450	9	1452	9.8	True
		HPL	841	86	9	88	86.1	True
	GBFGS	H1	8910	990	17	992	6.0	False
3	AS	H1	159716	18235	12	18237	3094.8	True
		HIP	44944	5040	12	5042	960.9	True
		HPL	2934	318	12	320	430.8	True
	GBFGS	H1	49429	5614	22	5616	47.5	False

Appendix

Optimal solutions

Out[73]:

Out[73]:

	AS	BFS	BFTS	GBFGS	RBFS	UCS
0	Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C2, P2, JFK)	Load(C1, P1, SFO)
1	Fly(P1, SFO, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
2	Load(C2, P2, JFK)	Fly(P2, JFK, SFO)	Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)	Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)
3	Fly(P2, JFK, SFO)	Unload(C2, P2, SFO)	Unload(C2, P2, SFO)	Fly(P2, JFK, SFO)	Unload(C2, P2, SFO)	Fly(P2, JFK, SFO)
4	Unload(C1, P1, JFK)	Fly(P1, SFO, JFK)	Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)	Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)
5	Unload(C2, P2, SFO)	Unload(C1, P1, JFK)	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)

Out[76]:

	AS	BFS	UCS
0	Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
1	Fly(P1, SFO, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
2	Load(C2, P2, JFK)	Load(C3, P3, ATL)	Load(C3, P3, ATL)
3	Fly(P2, JFK, SFO)	Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)
4	Load(C3, P3, ATL)	Unload(C2, P2, SFO)	Fly(P2, JFK, SFO)
5	Fly(P3, ATL, SFO)	Fly(P1, SFO, JFK)	Fly(P3, ATL, SFO)
6	Unload(C3, P3, SFO)	Unload(C1, P1, JFK)	Unload(C3, P3, SFO)
7	Unload(C1, P1, JFK)	Fly(P3, ATL, SFO)	Unload(C1, P1, JFK)
8	Unload(C2, P2, SFO)	Unload(C3, P3, SFO)	Unload(C2, P2, SFO)

Out[77]:

	AS	BFS	UCS
0	Load(C2, P2, JFK)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
1	Fly(P2, JFK, ORD)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
2	Load(C4, P2, ORD)	Fly(P2, JFK, ORD)	Fly(P1, SFO, ATL)
3	Fly(P2, ORD, SFO)	Load(C4, P2, ORD)	Load(C3, P1, ATL)
4	Load(C1, P1, SFO)	Fly(P1, SFO, ATL)	Fly(P2, JFK, ORD)
5	Fly(P1, SFO, ATL)	Load(C3, P1, ATL)	Load(C4, P2, ORD)
6	Load(C3, P1, ATL)	Fly(P1, ATL, JFK)	Fly(P2, ORD, SFO)
7	Fly(P1, ATL, JFK)	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
8	Unload(C4, P2, SFO)	Unload(C3, P1, JFK)	Unload(C4, P2, SFO)
9	Unload(C3, P1, JFK)	Fly(P2, ORD, SFO)	Unload(C3, P1, JFK)
10	Unload(C1, P1, JFK)	Unload(C2, P2, SFO)	Unload(C1, P1, JFK)
11	Unload(C2, P2, SFO)	Unload(C4, P2, SFO)	Unload(C2, P2, SFO)