Planning Search Written Analysis

In the context of the AIND Planning Search Heuristic project, multiple search algorithms were tested using 3 problems on the Air Cargo scenario. These tests were done over an Ubuntu 14 virtual machine with 2 GB of RAM and 2 processor cores.

Planning Problem Representation

The following 3 problems where given to test the search algorithms:

Problem 1

$Init(At(C1, SFO)$

$\wedge\ At(C2, JFK) \wedge At(P1, SFO) \wedge At(P2, JFK)$

$\wedge\ Cargo(C1) \wedge Cargo(C2)$

$\wedge\ Plane(P1) \wedge Plane(P2)$

$\wedge\ Airport(JFK)\ \wedge\ Airport(SFO))$

$Goal(At(C1, JFK)\ \wedge\ At(C2, SFO))$

Problem 2

$Init(At(C1, SFO)$

$\wedge\ At(C2, JFK) \wedge At(C3, ATL) \wedge At(P1, SFO) \wedge At(P2, JFK) \wedge At(P3, ATL)$

$\wedge\ Cargo(C1) \wedge Cargo(C2) \wedge Cargo(C3)$

$\wedge\ Plane(P1) \wedge Plane(P2) \wedge Plane(P3)$

$\wedge\ Airport(JFK)\ \wedge\ Airport(SFO)\ \wedge\ Airport(ATL))$

$Goal(At(C1, JFK)\ \wedge\ At(C2, SFO)\ \wedge\ At(C3, SFO))$

Problem 3

$Init(At(C1, SFO)$

$\wedge\ At(C2, JFK) \wedge At(C3, ATL) \wedge At(C4, ORD) \wedge At(P1, SFO) \wedge At(P2, JFK)$

$\wedge\ Cargo(C1) \wedge Cargo(C2) \wedge Cargo(C3) \wedge Cargo(C4)$

$\wedge\ Plane(P1) \wedge Plane(P2)$

$\wedge\ Airport(JFK)\ \wedge\ Airport(SFO)\ \wedge\ Airport(ATL)\ \wedge\ Airport(ORD))$

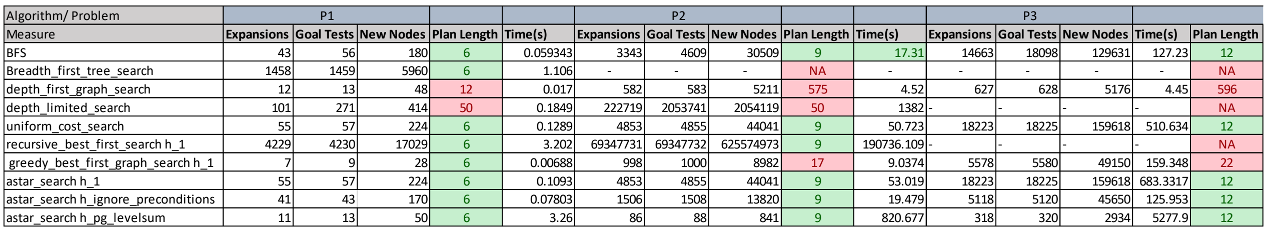
$Goal(At(C1, JFK)\ \wedge\ At(C3, JFK)\ \wedge\ At(C2, SFO)\ \wedge\ At(C4, SFO))$

Optimal Sequence of Actions

To select the optimal sequence of actions for each problem, the performance of the algorithms was evaluated using the time, space, optimality and completeness criteria described in the lectures. As a consequence, the following measures were taken after each algorithm run:

- Number of node expansions: The least the better, as it will take less space.
- Execution time: time taken to reach the goal. The lest, the better.
- Number of actions taken: the least actions taken the better.
- Completeness: defines if reaches the goal (Boolean value). Problems that took more than 10 minutes are not complete. If it didn't reach the goal, a "- "sign can be found in the table.

The different paths given by the algorithms can be found in the *Analysis.xlsx* file in the root of this project.



Algorithm/ Problem		P1					P2					P3			
Measure	Expansions	Goal Tests	New Nodes	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes	Plan Length	Time(s)	Expansions	Goal Tests	New Nodes	Time(s)	Plan Length
BFS	43	56	180	6	0.059343	3343	4609	30509	9	17.31	14663	18098	129631	127.23	12
Breadth_first_tree_search	1458	1459	5960	6	1.106	-	-	-	NA	-	-	-	-	-	NA
depth_first_graph_search	12	13	48	12	0.017	582	583	5211	575	4.52	627	628	5176	4.45	596
depth_limited_search	101	271	414	50	0.1849	222719	2053741	2054119	50	1382	-	-	-	-	NA
uniform_cost_search	55	57	224	6	0.1289	4853	4855	44041	9	50.723	18223	18225	159618	510.634	12
recursive_best_first_search h_1	4229	4230	17029	6	3.202	69347731	69347732	625574973	9	190736.109	-	-	-	-	NA
greedy_best_first_graph_search h_1	7	9	28	6	0.00688	998	1000	8982	17	9.0374	5578	5580	49150	159.348	22
astar_search h_1	55	57	224	6	0.1093	4853	4855	44041	9	53.019	18223	18225	159618	683.3317	12
astar_search h_ignore_preconditions	41	43	170	6	0.07803	1506	1508	13820	9	19.479	5118	5120	45650	125.953	12
astar_search h_pg_levelsum	11	13	50	6	3.26	86	88	841	9	820.677	318	320	2934	5277.9	12

Figure 1 Algorithms run results

Given the results, the following optimal sequence of actions (OSA) were selected based on an analysis where the objective is to minimize the number of actions, nodes expanded and time taken for each algorithm:

Problem 1

$Load(C1, P1, SFO)$

$Load(C2, P2, JFK)$

$Fly(P1, SFO, JFK)$

$Fly(P2, JFK, SFO)$

$Unload(C1, P1, JFK)$

$Unload(C2, P2, SFO)$

For problem 1, choosing the OSA is easy. Even if 8 algorithms give results with 9 actions, the SA given by *greedy best first graph search* h_1 takes only 0.00688 seconds with 7 nodes. It's the minimum amount of time, space, and actions needed to achieve the goal, between all the available options.

Problem 2

$Load(C2, P2, JFK)$

$Load(C1, P1, SFO)$

$Load(C3, P3, ATL)$

$Fly(P2, JFK, SFO)$

$Unload(C2, P2, SFO)$

$Fly(P1, SFO, JFK)$

$Unload(C1, P1, JFK)$

$Fly(P3, ATL, SFO)$

$Unload(C3, P3, SFO)$

Problem 2 is more complicated, as there are 6 algorithms that take 9 actions. The SA given by *A* search* levelsum shows the least amount of expanded nodes, which is space efficient, but takes 820.677 seconds to run so it's very time inefficient. The result given by a simple *BFS* gives the least amount of time between all algorithms with a 17.31 execution time by expanding 3343 nodes, which would be the minimum of all available results ignoring *A* search levelsum* node expansion.

Problem 3

$Load(C2, P2, JFK)$

$Fly(P2, JFK, ORD)$

$Load(C4, P2, ORD)$

$Fly(P2, ORD, SFO)$

$Unload(C4, P2, SFO)$

$Load(C1, P1, SFO)$

$Fly(P1, SFO, ATL)$

$Load(C3, P1, ATL)$

$Fly(P1, ATL, JFK)$

$Unload(C3, P1, JFK)$

$Unload(C1, P1, JFK)$

$Unload(C2, P2, SFO)$

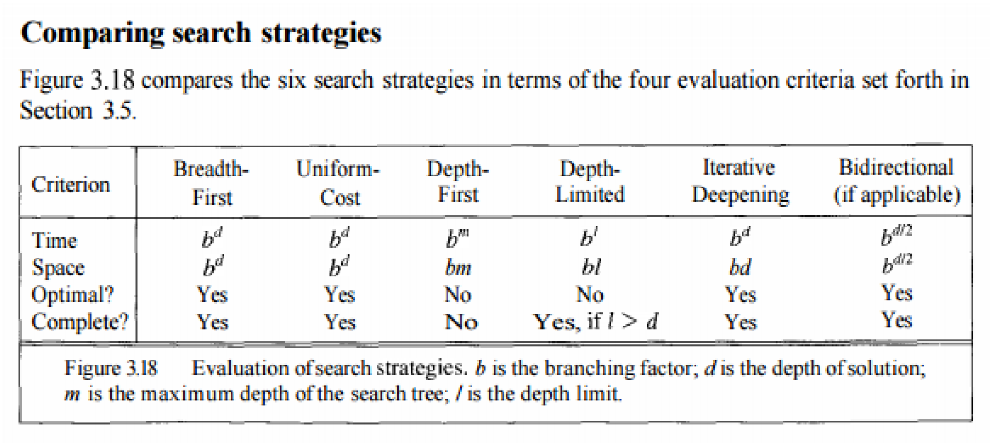
For problem 3, there are 5 algorithms that take 12 actions. However, choosing the optimal SA is quite easy considering that *A* Ignore Preconditions* takes 125.953 seconds vs 127.23, 510.63, and 683.33 given by the other algorithms with 12 actions and expands 5118 nodes, less than 1/3 of the second best option. Nevertheless, it's important to mention that *depth first graph search* found the SA with the least amount of nodes expanded and in the shortest time but suggests a SA of 596 actions to take, which isn't considered according to the 10-minute constraint of this project.

Performance Comparison

As seen in *Artificial Intelligence, a Modern Approach* from Russell & Norvig, a proper way of evaluating the performance of the search strategy search algorithms is through the following criteria:

- Completeness: Is the strategy guaranteed to find a solution when there is one?
- Time complexity: how long does it take to find a solution?
- Space complexity: how much memory does it need to perform the search?
- Optimality: does the strategy find the highest-quality solution when there are several different solutions?

A comparison of these measures can be seen in figure 2. The performance comparison based on the execution of the different search algorithms is presented below.



Comparing search strategies

Figure 3.18 compares the six search strategies in terms of the four evaluation criteria set forth in Section 3.5.

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete?	Yes	Yes	No	Yes, if $l > d$	Yes	Yes

Figure 3.18 Evaluation of search strategies. b is the branching factor; d is the depth of solution; m is the maximum depth of the search tree; l is the depth limit.

Figure 2 Comparison of search algorithms taken from Artificial Intelligence, a Moderon Approach

Optimality

An optimal solution to a problem is one that is the best solution according to some measure of solution quality. In the context of the course, the measures of solution quality are the minimum number of actions (or plan length), expanded nodes (memory) and execution time.

Uninformed planning algorithms

Algorithm/ Problem	Optimal?	P1		P2			P3			
Measure		Expansions	Time(s)	Plan Length	Expansions	Time(s)	Plan Length	Expansions	Time(s)	Plan Length
BFS	yes	43	0.059343	6	3343	17.31	9	14663	127.23	12
Breadth_first_tree_search	yes	1458	1.106	6	-	-	NA	-	-	NA
depth_first_graph_search	No	12	0.017	12	582	4.52	575	627	4.45	596
depth_limited_search	No	101	0.1849	50	222719	1382	50	-	-	NA
uniform_cost_search	yes	55	0.1289	6	4853	50.723	9	18223	510.634	12
recursive_best_first_search h_1	yes	4229	3.202	6	69347731	190736.109	9	-	-	NA
greedy_best_first_graph_search h_1	yes	7	0.00688	6	998	9.0374	17	5578	159.348	22

Figure 3 Metrics over Uninformed algorithms per problem

Problem 1

From the given results, it's clear that all the algorithms are guaranteed to find a solution for P1. It can be observer that *BFS, Breadth First Tree Search, Uniform Cost Search, Recursive Best First Search and greedy best first graph search* find plans with the least amount of actions needed, with low execution times. However, *greedy best first graph search* excels vs the other algorithms as it creates a short plan in only 0.00688 seconds and with the least amount of expansions. Nevertheless, it would be a mistake to conclude of the performance of the algorithms with this test as the problem has a small search space.

Problem 2

Problem 2 results are much more interesting. On one side, *Depth First Graph Search* returns a sequence of actions in the shortest time (4.52s) and expanding the least amount of nodes. However, the path contains 575 actions, which is not optimal in the Air Cargo Scenario (or in real life). On the other side, *BFS, Uniform Cost Search* and *Recursive Best First Search* return plans with the least amount of actions – that is 9 actions – and it can be seen that *BFS* takes the second least amount of time (17.31s) and expanding the second least amount of nodes. *BFS* is guaranteed to find a solution if there's one, and it will always find the shallowest goal first (Russell, & Norvig, 1994).

No data could be gathered for *Breadth First Tree Search*, as it didn't returned results under the 10 min limit defined for the project (and even some hours further).

Problem 3

Problem 3 results show that, contrary to what was seen in Problem 1, *greedy best first graph search* did not return an optimal plan, as the returned plan requires 22 actions while *Recursive Best First Search* and *BFS* return 12 actions. However, it can be seen that it is memory efficient, as expands only 5578 nodes, almost 1/3 vs 14663 expanded by BFS. It's execution time is close to BFS, taking 159 s vs 127s (32 more). It's important to mention that *Depth First Graph Search* shows up again as the most time and memory efficient algorithm, however it returns a plan with 596 actions, which is not optimal. Uniform cost search manages to return a 12 action plan, however, it shows to be very expensive in memory and time vs the other available algorithms.

In this case, to choose the best search algorithm, a trade off analysis must be done for the problem that needs to be solved. If time and memory are not an issue, *BFS* is the best option and uniform cost would be the second best option. If time and memory are critical and any number of actions will satisfy the problem conditions, Depth *First Graph Search* will be the best choice, followed by *greedy best first graph search*.

No data could be gathered for *Breadth First Tree Search, Depth Limited Search* and *Recursive Best First Search*, as they didn't returned results under the 10 min limit defined for the project (and even some hours further).

Conclusions

- *BFS* behaves as the best search algorithm delivering the plan with the least amount of actions needed to achieve the goal. However, as seen in *Artificial Intelligence a Modern Approach*, it might not be the best option to use if there are memory and time limitations.
- *Depth First Graph Search* is a good search algorithm option when memory is limited, as it only needs to store a branch of the search tree. It might also execute in fewer time than *BFS* (for these problems, it did) but it might return the wrong path or a non-optimal plan (if there's any).
- *Breadth First Tree Search, Depth Limited Search* and *Recursive Best First* algorithms are shown to take long execution times when the search space is to big as they expand a vast amount of nodes. Under certain constraints (and scenarios) they are not able to find the solution.

Automatic heuristics (Informed Search)

Algorithm/ Problem	Optimal?	P1			P2			P3		
Measure		Expansions	Time(s)	Plan Length	Expansions	Time(s)	Plan Length	Expansions	Time(s)	Plan Length
astar_search h_1	yes	55	0.1093	6	4853	53.019	9	18223	683.3317	12
astar_search h_ignore_preconditions	yes	41	0.07803	6	1506	19.479	9	5118	125.953	12
astar_search h_pg_levelsum	yes	11	3.26	6	86	820.677	9	318	5277.909	12

For a start, it's Important to note that A* is optimally efficient for any given heuristic and complete on locally infinite graphs (Russell, & Norvig, 1994), thus, making the 3 algorithms with different heuristics optimal and finding solutions in all cases (though, sometimes surpassing the 10 limit restriction on this project).

As a result, there's no point on making a separate comparison for each problem. It can be observed that all the problems give a plan with the least amount of actions needed (6 for P1, 9 for P2 and 12 for P3). The *levelsum* heuristic expands the least amount of nodes for the 3 problems but takes the most time and the *ignore preconditions* takes the least amount of time expanding few nodes vs simple *A**. The algorithm (or heuristic in this case) to select, depends on the restrictions of time and memory.

Conclusions

- It was observed that A* will always be optimal for any given heuristic.
- In many cases, a more accurate heuristic is obtained by considering at least the positive interactions arising from actions that achieve multiple goals (Russell, & Norvig, 1994).

Bibliography

(2017). Retrieved 9 May 2017, from http://robotics.usc.edu/~geoff/cs599/RusselNorvig.pdf

Artificial Intelligence Popular Search Algorithms. (2017). *www.tutorialspoint.com*. Retrieved 9 May 2017, from https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_popular_search_algorithms.htm

David Poole, D., & Mackworth, A. (2010). *Artificial Intelligence - foundations of computational agents -- 1.4.1 Defining a Solution*. *Artint.info*. Retrieved 9 May 2017, from http://artint.info/html/ArtInt_9.html

Russell, S., & Norvig, P. (1994). *Artificial intelligence A Modern Approach* (1st ed.).