Introduction

In Part 1, in the **my_air_cargo_problems.py** module 3 OFF deterministic logistic Air Cargo Planning Problem definitions based on initial states and goals provided and other class methods were correctly represented for the Planning Search Agent of a transport system defined in classical PDDL and solved using PL by running 8 OFF uninformed planning search algorithms including BFS, BFTS, DFGS, DLS, UCS, RBFS (using H1), GBFGS (using H1), A*S (using H1).

A performance comparison of non-heuristic search result metrics in the **Specific Metrics for Performance Comparison section** was achieved through data collection using **run_search.py**, all of which is shown in the **All Metrics section** and through description and analysis (using a table and visualisations) in **Description and Analysis Performance Comparison section**, which includes optimality of the solutions, and metrics including time elapsed, and number of node expansions required, and explains the reason for the observed results using one justification from the AI Nanodegree video lessons, the AIMA textbook, or another resource, for each of the 3 OFF Air Cargo problems against the 3 OFF non-heuristic planning algorithms: BFS, DFGS, and UCS.

In Part 2, additional 3 OFF domain-independent A*S algorithms (using automated heuristics HIP, HIDL for relaxed version of original problem, and HPGLS using the implemented PG of **my_planning_graph.py**) for planning were correctly implemented. The PG is an approximation of the search three representing all possible paths and provides automated admissible heuristics.

The performance comparison approach used in Part 1 was repeated in Part 2 but for heuristic search result metrics for the following 2 OFF planning algorithms (that use heuristics): A*S (using HIP), and A*S (using HPGLS Sum with a PG).

The state space of each problem may be calculated as two (2) to the power of the quantity of fluents in the initial state (both positive and negative. The state space of each problem is as follows:

Problem	Qty of Initial State Space Positive Fluent	Qty of Initial State Space Negative Fluents	Total Qty of State Space Fluents	State Space Size
1	4	8	12	2^{12}
2	6	21	27	2^{27}
2	6	26	32	2^{32}

Table 0.1: State Space Size of each Problem

Times elapsing over 10 minutes using certain search algorithms are shown as “Too Long” in the tables.

The optimal sequence of actions for each of the 3 OFF Air Cargo problems is identified in the **Optimal Plans section**.

The best heuristic used in the 3 OFF Air Cargo Problems is identified in the **Best Heuristic section** and a comparison is made that explains why it is or is not better than all the non-heuristic search planning methods for all problems.

Please refer to the **Appendix** for sections showing **Definitions**, **Abbreviations**, and **References** used.

Section 1: All Metrics

• Problem 1

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	43	56	180	6	0.054	PP
BFTS	-	1458	1459	5960	6	1.580	PP
DFGS	-	21	22	84	20	0.024	PP
DLS	-	101	271	414	50	0.144	PP
UCS	-	55	57	224	6	0.066	PP
RBFS	H1	4229	4230	17023	6	4.437	PP
GBFGS	H1	7	9	28	6	0.009	PP
A*S	H1	55	57	224	6	0.063	PP
A*S	HIP	41	43	170	6	0.068	DIH
A*S	HIDL	55	57	224	6	0.087	DIH
A*S	HPGLS	11	13	50	6	2.160	DIH

Table 1.1: Metrics for Problem 1

- ## Problem 2

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	3343	4609	30509	9	24.970	PP
BFTS	-	-	-	-	-	Too long	PP
DFGS	-	624	625	5602	619	5.480	PP
DLS	-	-	-	-	-	Too long	PP
UCS	-	4852	4854	44030	9	65.983	PP
RBFS	H1	-	-	-	-	Too long	PP
GBFGS	H1	990	992	8910	17	11.063	PP
A*S	H1	4852	4854	44030	9	65.515	PP
A*S	HIP	1506	1508	13820	9	21.434	DIH
A*S	HIDL	4852	4854	44030	9	76.008	DIH
A*S	HPGLS	86	88	841	9	256.832	DIH

Table 1.2: Metrics for Problem 2

- ## Problem 3

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	14663	18098	129631	12	196.914	PP
BFTS	-	-	-	-	-	Too long	PP
DFGS	-	408	409	3364	392	3.574	PP
DLS	-	-	-	-	-	Too long	PP
UCS	-	18235	18237	159716	12	595.826	PP
RBFS	H1	-	-	-	-	Too long	PP
GBFGS	H1	5164	5616	49429	22	169.054	PP
A*S	H1	18235	18237	159716	12	605.810	PP
A*S	HIP	5118	5120	45650	12	138.025	DIH
A*S	HIDL	18235	18237	159716	12	623.272	DIH
A*S	HPGLS	-	-	-	-	-	DIH

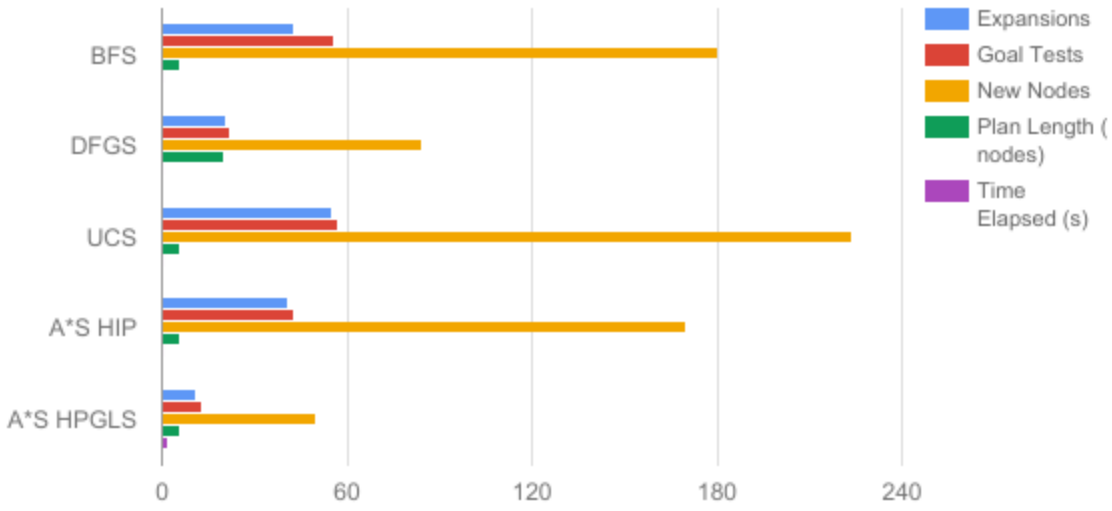
Table 1.3: Metrics for Problem 3

Section 2: Specific Metrics for Performance Comparison

- ### Problem 1

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	43	56	180	6	0.054	PP
DFGS	-	21	22	84	20	0.024	PP
UCS	-	55	57	224	6	0.066	PP
A*S	HIP	41	43	170	6	0.068	DIH
A*S	HPGLS	11	13	50	6	2.160	DIH

Table 2.1: Specific Metrics for Problem 1

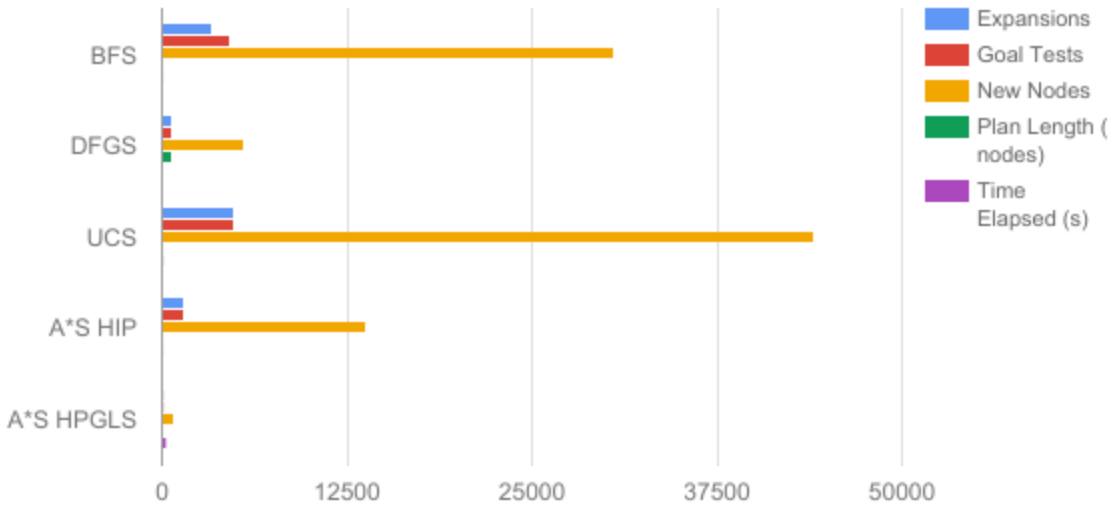


Graph 2.1: Specific Metrics for Problem 1

- ## Problem 2

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	3343	4609	30509	9	24.970	PP
DFGS	-	624	625	5602	619	5.480	PP
UCS	-	4852	4854	44030	9	65.983	PP
A*S	HIP	1506	1508	13820	9	21.434	DIH
A*S	HPGLS	86	88	841	9	256.832	DIH

Table 2.2: Specific Metrics for Problem 2

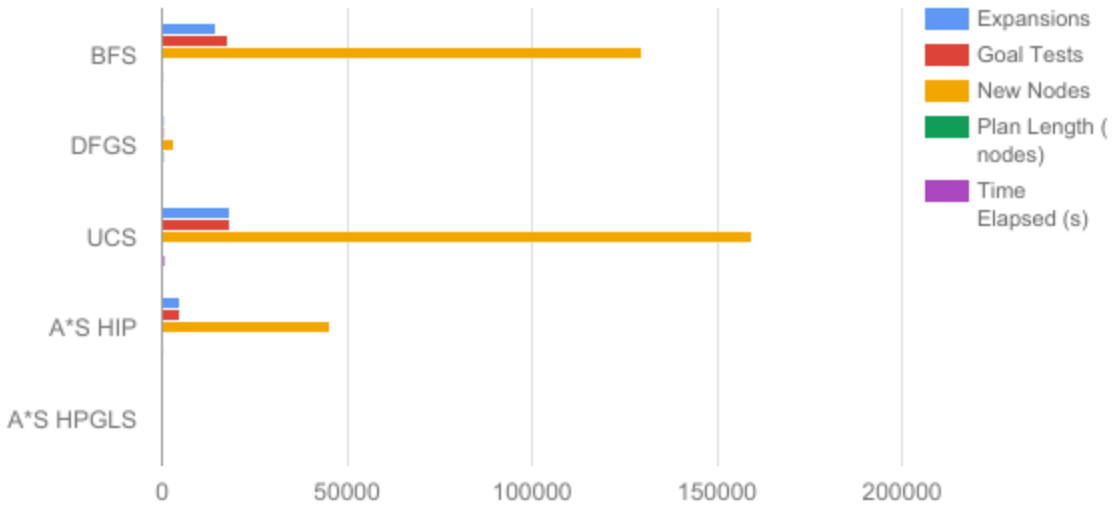


Graph 2.2: Specific Metrics for Problem 2

- ## Problem 3

Algorithm	Heuristic	Expansions	Goal Tests	New Nodes	Plan Length (nodes)	Time Elapsed (s)	Category
BFS	-	14663	18098	129631	12	196.914	PP
DFGS	-	408	409	3364	392	3.574	PP
UCS	-	18235	18237	159716	12	595.826	PP
A*S	HIP	5118	5120	45650	12	138.025	DIH
A*S	HPGLS	-	-	-	-	-	DIH

Table 2.3: Specific Metrics for Problem 3



Graph 2.3: Specific Metrics for Problem 3

Section 3: Description and Analysis Performance Comparison

Comparison of Problems, Algorithms, and Heuristics	Type	Comparison (* with references to AIND Videos or AIMA textbook)
P1 with BFS vs DFGS	Non-Heuristic	**Optimality:** BFS is optimal and guaranteed to find the shortest path (see Reference[0]) with plan length of 6. DFGS is not optimal since there is more than one goal state (see Reference[0]), so it has a plan length of 20. **Time Elapsed:** DFGS finds goal 2x faster than BFS. **Node Expansions:** BFS expands >2x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n

		nodes in BFS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** In the Tri-City Search Problem of Reference[0] we considered the 2^{12} State Space of Problem 1 and Time advantages and disadvantages of each algorithm: DFGS option that checks every node from bottom leaf left to right was dismissed as not viable as it may cross the country multiple times just for the first branch. BFS option requires 2x sequential searches from JFK to SFO and SFO to JFK to determine shortest path, with a search range further than necessary. Since there are only two nodes, the issue of exploring intersecting nodes repeatedly (duplicates) is not apparent.
P1 with BFS vs UCS	Non-Heuristic	**Optimality:** Both BFS and UCS are optimal with plan length of 6. **Time Elapsed:** BFS finds goal 1.2x faster than UCS. **Node Expansions:** UCS expands 1.2x more nodes than BFS. **Justification of Observed Results *:** BFS finds the shortest path in terms of the least number of steps, but it will not find the shortest path in terms of the shortest total cost (sum of step costs). UCS takes longer and expands more nodes than BFS since even after finding a path to the goal state it continues searching to try and find a cheaper path that also reaches the goal state (see Reference[0])
P1 with DFGS vs UCS	Non-Heuristic	**Optimality:** Same as P1 with BFS vs DFGS but replace results of BFS with UCS. **Time Elapsed:** DFGS finds goal 3x faster than BFS. **Node Expansions:** UCS expands >2x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n nodes in UCS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** Same justification applies as in P1 with BFS vs DFGS, but replacing UCS with BFS.
P2 with BFS vs DFGS	Non-Heuristic	**Optimality:** BFS is optimal and guaranteed to find the shortest path (see Reference[0]) with plan length of 9. DFGS is not optimal since there is more than one goal state (see Reference[0]), so it has a plan length of 619.

		Time Elapsed: DFGS finds goal 5x faster than BFS. **Node Expansions:** BFS expands >6x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n nodes in BFS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** Same justification as in P1 with BFS vs DFGS, but in P2 where there are 3x goal states, the BFS option requires 4x sequential searches from JFK to SFO, SFO to JFK, ATL to SFO, and SFO to ATL to determine shortest path, with a search range further than necessary. Since there are more than two nodes, the issue of exploring intersecting nodes repeatedly (duplicates) is apparent.
P2 with BFS vs UCS	Non-Heuristic	**Optimality:** Both BFS and UCS are optimal with plan length of 9. **Time Elapsed:** BFS finds goal >2x faster than UCS. **Node Expansions:** UCS expands 1.5x more nodes than BFS. **Justification of Observed Results *:** Same justification as for P1 with BFS vs UCS.
P2 with DFGS vs UCS	Non-Heuristic	**Optimality:** Same as P2 with BFS vs DFGS but replace BFS with UCS. **Time Elapsed:** DFGS finds goal 10x faster than UCS. **Node Expansions:** UCS expands >8x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n nodes in UCS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** Same justification applies as in P1 with BFS vs DFGS, but replacing results of BFS with UCS.
P3 with BFS vs DFGS	Non-Heuristic	**Optimality:** BFS is optimal and guaranteed to find the shortest path (see Reference[0]) with plan length of 12. DFGS is not optimal since there is more than one goal state (see Reference[0]), so it has a plan length of 392 (>30 times longer). **Time Elapsed:** DFGS finds goal 50x faster than BFS. **Node Expansions:** BFS expands >35x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n

		nodes in BFS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** Same justification as in P1 with BFS vs DFGS, but in P2 where there are 4x goal states, the BFS option requires 10x sequential searches from JFK to SFO, SFO to JFK, JFK to ORD, ORD to JFK, ORD to SFO, SFO to ORD, SFO to ATL, ATL to SFO, ATL to JFK, and JFK to ATL, to determine shortest path, with a search range further than necessary. Since there are more than two nodes, the issue of exploring intersecting nodes repeatedly (duplicates) is apparent.
P3 with BFS vs UCS	Non-Heuristic	**Optimality:** Both BFS and UCS are optimal with plan length 12. **Time Elapsed:** BFS finds goal >2x faster than UCS. **Node Expansions:** UCS expands 1.3x more nodes than BFS. **Justification of Observed Results *:** Same justification as for P1 with BFS vs UCS.
P3 with DFGS vs UCS	Non-Heuristic	**Optimality:** Same as P3 with BFS vs DFGS but replace BFS with UCS. **Time Elapsed:** DFGS finds goal 170x faster than UCS. **Node Expansions:** UCS expands >45x more nodes than DFGS since DFGS only uses Storage Space of n nodes instead of 2^n nodes in UCS when not tracking the explored set (see Reference[0]). **Justification of Observed Results *:** Same justification applies as in P2 with BFS vs DFGS, but replacing results of BFS with UCS.
P1 with A*S HIP vs A*S HPGLS	Heuristic	**Optimality:** A*S algorithm is not guaranteed to be optimal (see Reference[1]) but when used in combination with HIP and HPGLS heuristics, constraints are applied and optimal plan lengths of 6 are found. **Time Elapsed:** A*S HIP finds goal 30x faster than A*S HPGLS. **Node Expansions:** A*S HIP expands 4x more nodes than A*S HPGLS. **Justification of Observed Results *:** The HPGLS heuristic uses a Planning Graph but just estimates the sum of all actions that must be carried out from the current state to satisfy each individual goal condition, but the HIP heuristic finds the goal faster since it ignores preconditions required for an action to be

		executed to make the problem easier in order to estimate the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions (see Reference[2])
P2 with A*S HIP vs A*S HPGLS	Heuristic	**Optimality:** A*S algorithm will always find the lowest cost path to the goal dependent on whether the heuristic estimate function h for a state is less than the true cost of the path to the goal through that state, so it is therefore not guaranteed to be optimal (see Reference[1]) but when used in combination with HIP and HPGLS heuristics, constraints are applied and optimal plan lengths of 9 are found. **Time Elapsed:** A*S HIP finds goal 12x faster than A*S HPGLS. **Node Expansions:** A*S HIP expands >17x more nodes than A*S HPGLS. **Justification of Observed Results *:** Same justification as for P1 with A*S HIP vs A*S HPGLS.
P3 with A*S HIP vs A*S HPGLS	Heuristic	**Optimality:** A*S algorithm is not guaranteed to be optimal (see Reference[1]) but when used in combination with the HIP heuristics, constraints are applied and optimal plan length of 12 is found. No solution was found when using the HPGLS even after 10 minutes. **Time Elapsed:** Unable to compare as no solution was found when using HPGLS even after 10 minutes. **Node Expansions:** Unable to compare as no solution was found when using HPGLS even after 10 minutes. **Justification of Observed Results *:** The reason for HPGLS taking so long is explained in Section 3 and Section 3.2 of the paper Reference[3], which highlights the importance of using an approach like HIP that minimises the set of actions to remove redundant actions from consideration instead of HPGLS that uses a Planning Graph that estimates the sum of all actions including non-minimal ones that makes the search procedure unnecessarily slow.

Table 3.1: Performance Comparison between Algorithms and Heuristics for Problem 1, 2, and 3

Section 4: Optimal Plans

• Problem 1

Below are the optimal sequences of actions output by the respective algorithms and heuristics shown using Problem 1. Sequence No. 3 that was found using A*S HIP is clearly the most optimal plan since it achieves 50% of its At(C1, JFK) ∧ At(C2, SFO) goal fastest (by the 3rd action) by focusing on finishing one task at a time. It uses the shortest possible plan length of 6 and reaches the goal in a comparably fast time of 0.068 sec according to Table 2.1.

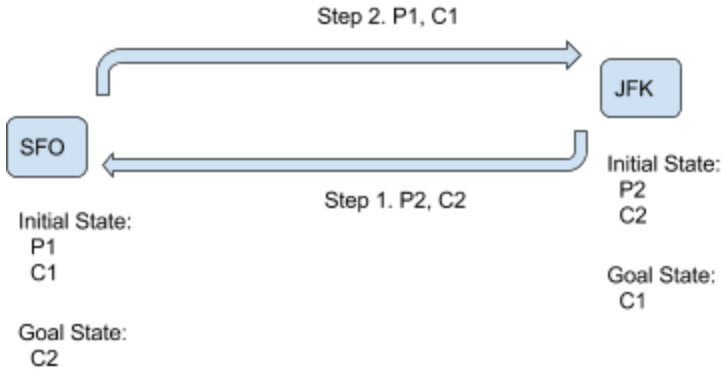


Figure 4.1: Visualisation of Optimal Solution for Problem 1

Sequence No.	Optimal Sequence of Actions	Plan Length (nodes)	Time Elapsed (s)	From Algorithms / Heuristics
1	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	6	See Table 2.1	BFS, BFTS, RBFS
2	Load(C1, P1, SFO) Load(C2, P2, JFK)	6	See Table 2.1	UCS, GBFGS, A*S H1, A*S HIDL

	Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)			
3	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	6	0.068	A*S HIP
4	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)	6	See Table 2.1	A*S HPGLS

Table 4.1: Optimal Solutions using Algorithms and Heuristics for Problem 1

• Problem 2

Below are the optimal sequences of actions output by the respective algorithms and heuristics shown using Problem 2.

Sequence No. 3 that was found using A*S HIP is clearly the most optimal plan since it achieves 33.33% of its At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, JFK) goal fastest (by the 3rd action) and 66.67% of its goal by the 6th action by focusing on finishing one task at a time. It uses the shortest possible plan length of 9 and reaches the goal in the fastest time of 21.434 sec as compared to all other optimal sequences according to Table 2.2.

It takes the shorter route from ATL to SFO first (instead of the longer route from SFO to JFK or JFK to SFO) to deliver a first portion of the goal faster.

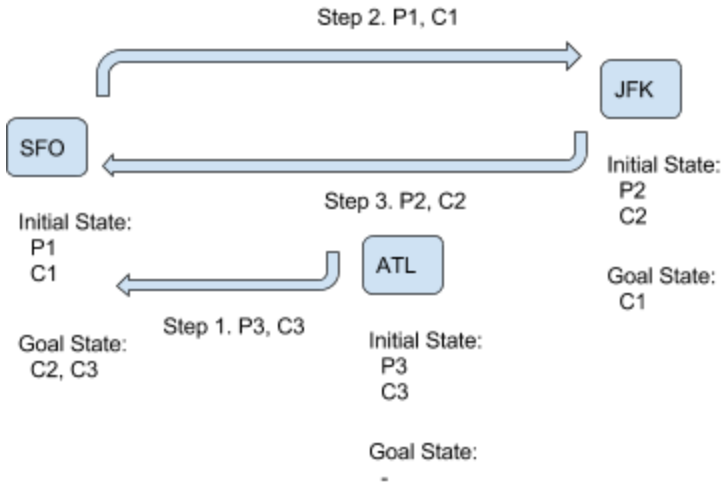


Figure 4.2: Visualisation of Optimal Solution for Problem 2

Sequence No.	Optimal Sequence of Actions	Plan Length (nodes)	Time Elapsed (s)	From Algorithms / Heuristics
1	Load(C1, P1, SFO) Load(C2, P2, JFK) 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)	9	See Table 2.2	BFS
2	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)	9	See Table 2.2	UCS, A*S H1, A*S HIDL

	Unload(C1, P1, JFK) Unload(C2, P2, SFO)			
3	Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO) Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	9	See Table 2.2	A*S HIP
4	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(C1, P1, JFK) Unload(C2, P2, SFO)	9	See Table 2.2	A*S HPGLS

Table 4.2: Optimal Solutions using Algorithms and Heuristics for Problem 2

- ## Problem 3

Below are the optimal sequences of actions output by the respective algorithms and heuristics shown using Problem 3.

Sequence No. 3 that was found using A*S HIP is clearly the most optimal plan since it achieves portions of its goal soonest. It achieves 25% of its At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, JFK) ∧ At(C4, SFO) goal fastest (by the 4th action), 50% of its goal by the 10th action, 75% by the 11th action, and 100% by the 12th action. It uses the shortest possible plan length of 12 (would use

least amount of fuel) and reaches the goal in the fastest time of 138.025 sec as compared to all other optimal sequences according to Table X.X.
It takes the shorter route from JFK to SFO via ORD first (instead of the longer SFO to JFK via ATL) to deliver the first portion of the goal faster.

It could be improved by moving Unload(C2, P2, SFO) to immediately after Unload(C4, P2, SFO) since it is likely most efficient to load/unload cargo FIFO. This would mean C2 would be unloaded from P2 at SFO at the same time it unloads C4 from P2 at SFO, instead of partially unloading the cargo (i.e. C4) from P2 at SFO but leaving C2 on board until P1 finishes transporting and unloading C1 and C3 at JFK.

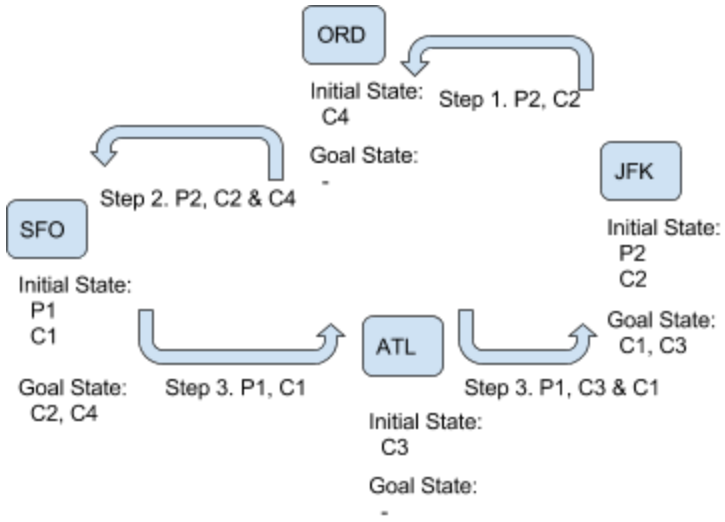


Figure 4.3: Visualisation of Most Optimal Solution for Problem 2

Most Optimal Sequence of Actions
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, 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)

Table 4.3: More Optimal Plan than A*S HIP result in Problem 3

Sequence No.	Optimal Sequence of Actions	From Algorithms / Heuristics
1	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)	BFS
2	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Fly(P1, ATL, JFK) Unload(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C1, P1, JFK)	UCS, A*S H1, A*S HIDL

	Unload(C2, P2, SFO)	
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)	A*S HIP

Table 4.4: Optimal Solutions using Algorithms and Heuristics for Problem 3

Section 5: Best Heuristic

The best heuristic used in Problems 1, 2, and 3, was the A*S HIP heuristic.
The pros and cons of using it compared to all the best optimal non-heuristic search planning methods for all the problems is as follows:

Problem	Comparison	Pros	Cons	Neutral
1	A*S HIP vs BFS vs UCS	DFS was not optimal and required a plan length nearly 4x times larger than A*S HIP.		BFS and UCS had similar timeframe and expansion results to A*S HIP and were all optimal.

2	A*S HIP vs BFS vs UCS	UCS took 2x longer than A*S HIP. BFS and UCS required 2x more expansions than A*S HIP. DFS was much faster than both but was not optimal and required a plan length nearly 70x larger.		BFS and A*S HIP both found a solution in the same timeframe.
3	A*S HIP vs BFS vs UCS	BFS took 1.4x longer than A*S HIP to find a solution. UCS took over 4x longer than A*S HIP to find a solution. BFS and UCS required >3x more expansions than A*S HIP. DFS was not optimal and required a plan length nearly 30x times larger.	A*S created 3x more new nodes than BFS and UCS.	

Table 5.1: Best Heuristic compared to optimal non-heuristic for all problems

Appendix

- ## Abbreviations

Category	Abbreviation	Meaning
Non-Heuristic Search	A*S	A* Search
Non-Heuristic Search	BFS	Breadth First Search
Non-Heuristic Search	BFTS	Breadth First Tree Search
Non-Heuristic Search	DFGS	Depth First Graph Search
Non-Heuristic Search	DLS	Depth Limited Search
Non-Heuristic Search	GBFGS	Greedy Best First Graph Search

Non-Heuristic Search	RBFS	Recursive Best First Search
Non-Heuristic Search	UCS	Uniform Cost Search
Heuristic Search	H1	H1
Heuristic Search	HIP	H Ignore Preconditions
Heuristic Search	HIDL	H Ignore Delete Lists
Heuristic Search	HPGLS	H PG Level Sum
Languages & Techniques	PDDL	Planning Domain Definition Language
Languages & Techniques	PL	Propositional Logic
Languages & Techniques	PG	Planning Graph
Problem Types	PP	Planning Problems
Problem Types	DIH	Domain-Independent Heuristics
Problem Types	P1	Problem 1
Problem Types	P2	Problem 2
Problem Types	P3	Problem 3
Approaches	FIFO	First In First Out

Table A.0: Abbreviations

Definitions

- **BFS** algorithm searches by expanding first the shortest possible path steps, since it always chooses from the Frontier one of the paths that has not been considered yet, which is shortest possible. It is Optimal and Complete (if goal at finite depth).

- **UCS** algorithm searches by expanding first for path with cheapest possible total cost first. It is Optimal and Complete (if goal has finite cost).

- **DFGS** algorithm searches by expanding first the longest path with the most links in it. It is NOT Optimal, and NOT Complete (since given an infinite path, DFS would keep following it and never get to path consisting of the goal). DFS only uses Storage Space of n nodes (instead of 2^n nodes in BFS) so saves a huge amount of space. But if it also tracks the explored set we do not save as much.

- **A*S** algorithm searches by always expanding path with the minimum value of the heuristic function f that is generated by relaxing the problem definition (defined as sum of g and h components) so result is a search strategy that is the best possible (i.e. it finds shortest length path whilst expanding minimum quantity of paths possible), where g of a path equals path cost (sum of path costs from initial state to current state), and where h of a path equals h value of the state (final state of the path), which equals the estimated straight line distance to the goal.

- **HIPS** heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.

- **HPGLS** heuristic uses a Planning Graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

References

- [0] AIND Lesson 7 Search Video
- [1] AIND Lesson 4 Intro to AI Video
- [2] AIND Lesson 11 Planning (Plan Space Search)
- [3] Marchetta, G, "Improving Planning Graph Analysis for Artificial Intelligence Planning" National University of Cuyo, Centro Universitario, Argentina, http://fing.uncu.edu.ar/catedras/informatica/mmarchetta-1/CLEI2008_marchetta.pdf