The following is the comparison of non-heuristic and heuristic search result metrics of solving deterministic logistics planning problems for an Air Cargo transport system. There are three problems to solve, and each problem's initial state and goal are the following:

Problem 1 initial state and goal:

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 initial state and goal:

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 initial state and goal:

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))

Regarding non-heuristic search, four types of algorithms were used to solve the problems: breadth-first search, depth-first graph search, uniform cost search, and greedy best first search. The following tables illustrate the results of each non-heuristic search algorithm used to solve the problems:

Uninformed non-heuristic search results

Problem1	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Breadth First	Yes	6	0.0240	43	56	180
Depth First Graph	No	12	0.0075	12	13	48
Uniform Cost	Yes	6	0.0438	55	57	224
Greedy Best First	Yes	6	0.0046	7	9	28

Problem2	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Breadth First	Yes	9	8.55	3,343	4,609	30,509
Depth First Graph	No	575	4.57	582	583	5,211
Uniform Cost	Yes	9	11.60	4,853	4,855	44,041
Greedy Best First	No	21	2.40	998	1,000	8,982

Problem3	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Breadth First	Yes	12	39.7	14,663	18,098	129,631
Depth First Graph	No	596	4.4	627	628	5,176
Uniform Cost	Yes	12	48.0	18,223	18,225	159,618
Greedy Best First	No	22	18.4	5,578	5,580	49,150

The following is the details of the comparison of each uninformed non-heuristic algorithms.

- **Breadth First Search:** This algorithm was optimal in all problems, as it is always expanding the shortest paths first, so wherever the goal is hiding, it's going to find the goal. However, it expands the nodes by 2^n nodes; therefore, it will take time to search the optimal solution.
- **Depth First Graph Search:** This algorithm was not optimal for all problems since it tries to go as deep as it can first, and doesn't necessarily find the shortest path of all. It would find the longer path for a solution; however, it expands the nodes by n-nodes rather than 2^n nodes. Therefore, the number of node expansions and time elapsed were less than the other optimal algorithm such as breadth-first search. There is also a disadvantage of using this algorithm when the path is infinite, where it will never get to the goal (not complete).
- **Uniform Cost Search:** This algorithm was always optimal in all problems since it's guaranteed to find the cheapest path of all, assuming that all the individual step costs are

non-negative. However, similar to the breadth-first search algorithm, it expands the nodes by 2^n nodes; therefore, it will take time to find the optimal solution.

- **Greedy Best First Search:** This algorithm was not always optimal (optimal in Problem 1 but not optimal in Problem 2 and 3). This algorithm is using a breadth-first search with a heuristic estimate of the goal. If there are obstacles along the way upon estimating the goal distance, it will not have the optimal solution, since the algorithms try always to get closer and closer to the goal. Therefore, this algorithm will find a path, and it does it by expanding a small number of nodes, but it accepts a path that is longer than the other possible shorter path.

Regarding heuristic search, three types of algorithms were used to solve the problem: A* h_1 (A* search without heuristic), A* h_,ignore_preconditions (A* search with heuristic that ignores the preconditions), and A* h_pg_levelsum (A* search with heuristic that uses a planning graph and level sum). The following tables illustrate the results of each heuristic search algorithm used to solve the problems:

Informed heuristic search results

Problem1	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
A* h_1	Yes	6	0.0358	55	57	224
A* h_ignore_preconditions	Yes	6	0.0373	41	43	170
A* h_pg_levelsum	Yes	6	0.9784	32	34	138

Problem2	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
A* h_1	Yes	9	12.58	4,853	4,855	44,041
A* h_ignore_preconditions	Yes	9	0.37	1,450	1,452	13,303
A* h_pg_levelsum	Yes	9	52.52	1,587	1,589	14,844

Problem3	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
A* h_1	Yes	12	56.5	18,223	18,225	159,618
A* h_ignore_preconditions	Yes	12	17.0	5,040	5,042	44,944
A* h_pg_levelsum	Yes	12	3747.4	8,532	8,534	77,857

The following is the details of the comparison of each informed heuristic algorithms.

- **A* h_1**: A* search algorithm always expands the path that has a minimum value of the function f, which is defined as a sum of the g and h components. Function g of a path is the path cost. Function h of a path is equal to the h value of the state, which is the final state of the path and is equal to the estimated distance to the goal. Minimizing g helps keep the path short. Minimizing h helps keep focused on finding the goal.
  A* will find the shortest length path while expanding a minimum number of paths possible; hence, the result is always optimal. The heuristic "A* h_1" sets the h value equals to one; therefore, this algorithm is nearly equal to the uninformed non-heuristic algorithms of uniform cost search, as the similar results were observed in the tables above. The following table is the comparison of the uniform cost and "A* h_1" algorithms:

Problem1	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Uniform Cost	Yes	6	0.0438	55	57	224
A* h_1	Yes	6	0.0358	55	57	224

Problem2	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Uniform Cost	Yes	9	11.60	4,853	4,855	44,041
A* h_1	Yes	9	12.58	4,853	4,855	44,041

Problem3	Optimality	Plan length	Time elapsed (sec)	Expansions	Goal Tests	New Nodes
Uniform Cost	Yes	12	48.0	18,223	18,225	159,618
A* h_1	Yes	12	56.5	18,223	18,225	159,618

- **A* h_ignore_preconditions:** This heuristic is an admissible heuristic, which is derived from a relaxed planning problem. The heuristic is relaxing the problem by ignoring the preconditions required for an action to be executed. Every action will always be applicable, and any literal can be achieved in one step. As this is an admissible heuristic, this heuristic found an optimal solution to all problems. Also, this heuristic outperformed the "A* h_1", which is the A* search algorithm without any heuristic, regarding the time elapsed and the number of node expansions.
- **A* h_pg_levelsum:** This heuristic uses a planning graph representation of the problem state space with the level sum heuristic. The level sum heuristic returns the sum of the level costs of the goals. This heuristic is inadmissible but works very well in practice for problems that are largely decomposable. Also, it is much more accurate than the

admissible heuristic derived from a relaxed planning problem such as “A* h_ignore_preconditions." Given that this heuristic is more accurate than the admissible heuristic, this heuristic was also optimal for all problems. However, as this heuristic uses a planning graph, which contains a rich source of information about the problem but takes time to construct, the heuristic was slower than the “A* h_1” algorithm, even though there was less number of node expansions required.

Based on the above, the following were the optimal plans for each problem:

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)

Problem 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)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Problem 3:

Load(C2, P2, JFK)
Load(C1, P1, SFO)
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)

In conclusion, the best heuristic for solving the problems was the “A* h_ignore_preconditions” algorithm (outperformed all uninformed non-heuristic search algorithms) since A* search with heuristic is a search strategy that is the best possible where it finds the shortest length path while expanding minimum number of paths possible (reference: AIND video “Search - Quiz: A* Search” 01:39). However, regarding Problem 1, an uninformed non-heuristic search of the breadth-first search was faster to solve the problem than the “A* h_ignore_preconditions” algorithm. That is because of the simplicity of problem, as a simple algorithm is faster to achieve an optimal solution when the problem is also simple to solve. Also, as these three problems were not largely decomposable, an admissible heuristic of “A* h_ignore_preconditions” algorithm was able to achieve the optimal solution for all cases. In case the problem was largely decomposable, the total-order planning (“A* h_ignore_preconditions”) can suffer from inaccuracies. In such case, “A* h_pg_levelsum” will be the best heuristic due to the usage of the special data structure of planning graph, although it might require more computational time to solve the problem (reference: AIMA textbook page 395, 11.4 Planning Graph).