Planning Search Heuristic Analysis

The problem involves creating an agent that solves the air cargo problems, which involves planning the sequence of action to achieve a specific goal. The Planning Domain Definition Language was adopted.

The three problems are defined as:

Problem 1:

```
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:

```
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:

```
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))
```

```
NB: C1, C2, C3, C4 are cargoes. 'JFK','SFO','ATL','ORD' are airports.
P1,P2, P3 are planes.
```

```
There were three possible actions defined by the action schemas below:

Action(Load(c, p, a),
  PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
  EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
  PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
  EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
  PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
  EFFECT: ¬ At(p, from) ∧ At(p, to))
```

A planning graph was used to which uninformed non-heuristic search methods and heuristic search method (**automatic domain-independent heuristics with A* search).**
The different search methods gave different solution to the problems, but the **optimal plan lengths** for problems 1,2, and 3 are **6, 9, and 12 actions**, the action sequences are given below.

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(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, ATL, JFK)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Analysis of Uninfomed search strategies.

This search strategies have no additional information about the states beyond the problem definition, so they only generate successors and test for goal states. In this project they are 7 in number, they are: Breadth first search, Breadth first tree search, Depth first graph search, Depth limited search, Uniform cost search, Recursive best first search, Greedy best first graph search. The strategies are compared on the basis of **speed** (execution time, measured in seconds), **memory usage** (measured in search node expansions) and **optimality** (Yes, if a solution of optimal length is found; No, otherwise).

Problem 1 Result

Search type	**Optimal**	**Expansions**	**Plan length**	**Time elapsed (seconds)**
Breadth first search	Yes	43	6	0.13526131078509163
Breadth first tree search	Yes	1458	6	3.0198156929219784
Depth first graph search	NO	12	12	0.03187458981723745
Depth limited search	NO	101	50	0.26496919763781657
Uniform cost search	YES	55	6	0.12111775480575204
Recursive best first search with h_1	YES	4229	6	9.216611379623982
Greedy best first graph search with h_1	YES	7	6	0.01963806308438798

For problem 1, only depth first graph search and depth limited search didn't give optimal path solutions. In terms of speed Greedy best first graph search comes first with approximately 0.0196 seconds with recursive best first search coming last with 9.2 seconds. In terms of memory used (judged in terms number of expansions done to reach a solution) Greedy best first graph search came first requiring just 7 expansions with recursive best first graph search coming last with 4229 expansions required. For problem 1, it is clear that the best uninformed search strategy is the greedy best first graph search as it yields an optimal solution in the shortest time and with the smallest memory space.

Problem 2 result

Search type	**Optimal**	**Expansions**	**Plan length**	**Time elapsed (seconds)**
Breadth first search	YES	3343	9	48.59207630227439
Breadth first tree search	—	—	—	—
Depth first graph search	—	—	—	—
Depth limited search	—	—	—	—
Uniform cost search	YES	4853	9	43.87667389008465
Recursive best first search with h_1	—	—	—	—
Greedy best first graph search with h_1	NO	998	21	8.334654907141513

For problem 2, breadth first tree search, depth first graph search, depth limited search and recursive best first search yielded no solutions within 15 minutes, only breadth first search and uniform cost search yielded optimal results (with a plan length of 9), greedy best first graph yielded a non-optimal result (plan length of 21) . In terms of speed Greedy best first

graph search comes first with approximately 8.33 seconds with breadth first search coming last with 48.59 seconds. In terms of memory used (judged in terms number of expansions done to reach a solution) Greedy best first graph search came first requiring 998 expansions with uniform cost search coming last with 4853 expansions required.

For problem 2, if what is needed is just an optimal solution at regardless of the time and memory cost, either breadth first search or uniform cost search would give optimal solutions but at a significantly higher time and memory cost compared to greedy best first graph search. Greedy best first search yields a solution with the least amount of time and memory consumption.

Problem 3

Search type	**Optimal**	**Expansions**	**Plan length**	**Time elapsed (seconds)**
Breadth first search	YES	14663	12	319.3527341658095
Breadth first tree search	__	__	__	__
Depth first graph search	__	__	__	__
Depth limited search	__	__	__	__
Uniform cost search	YES	18223	12	178.34752515616623

Recursive best first search with h_1	__	__	__	__
Greedy best first graph search with h_1	NO	5578	22	54.21172685220881

For problem 3, breadth first tree search, depth first graph search, depth limited search and recursive best first search yielded no solutions within 15 minutes, only breadth first search and uniform cost search yielded optimal results (with a plan length of 12), greedy best first graph yielded a non-optimal result (plan length of 22) . In terms of speed Greedy best first graph search comes first with approximately 54.21 seconds with breadth first search coming last with 319.35 seconds. In terms of memory used (judged in terms number of expansions done to reach a solution) Greedy best first graph search came first requiring 5578 expansions with uniform cost search coming last with 18223 expansions required.

For problem 3, if what is needed is just an optimal solution at regardless of the time and memory cost, either breadth first search or uniform cost search would give optimal solutions but at a significantly higher time and memory cost compared to greedy best first graph search. Greedy best first search yields a solution with the least amount of time and memory consumption.

Generally, the greedy best first search seems to be the best search algorithm to use as it saves significantly more time and memory space even though it might not yield the optimal solution.

INFORMED (HEURISTIC) SEARCH STRATEGIES ANALYSIS

This search strategies use problem specific knowledge beyond the problem definition, they are generally more efficient compared to uninformed search strategies. Three different

searches were used (A* with h1 heuristic, A* search with ignore preconditions heuristic, A* search with level sum heuristic). The strategies are compared on the basis of **speed** (execution time, measured in seconds), **memory usage** (measured in search node expansions) and **optimality** (Yes, if a solution of optimal length is found; No, otherwise).

Problem 1 Result.

SEARCH TYPES	**OPTIMAL**	**EXPANSIONS**	**PLAN LENGTH**	TIME ELAPSED (SECONDS)
Astar search with h_1	YES	55	6	0.1560595370661595
Astar search with h_ignore preconditions	YES	41	6	0.15018866575077006
Astar search with h pg levelsum	YES	11	6	3.344779514765701

For problem 1, they all yielded optimal solutions (plan length of 6). In terms of time cost, A* search with ignore preconditions search yielded the best result (0.150 seconds), with A* search with h1 heuristic coming second with 0.156 seconds and A* search with levelsum heuristic coming last with 3.345 seconds. However, A* search with level sum heuristic comes top in terms of memory use requiring just 11 node expansion, while A* with ignore preconditions and A* with h1 heuristics came 2nd and third respectively with 41 and 55 expansions required respectively.
Choosing the heuristic to use would depend on the need required. In terms of optimality, any of the solutions can be used. In terms of time A* search with ignore preconditions is the fastest. In terms of memory use regardless of the time cost, A* with level sum heuristic would be the ideal as it uses significantly lesser memory space compared to the rest. However, A* search with ignore preconditions has the best memory and time balance.

Problem 2 result.

SEARCH TYPES	OPTIMAL	EXPANSIONS	PLAN LENGTH	TIME ELAPSED (SECONDS)
Astar search with h_1	YES	4853	9	41.46982007504421
Astar search with h_ignore preconditions	YES	1450	9	15.622997963705473
Astar search with h pg levelsum	YES	86	9	544.7542652412117

For problem 2, they all yielded optimal solutions (plan length of 9). In terms of time cost, A* search with ignore preconditions search yielded the best result (15.62 seconds), with A* search with h1 heuristic coming second with 41.47 seconds and A* search with levelsum heuristic coming far behind with 544.75 seconds. However, A* search with level sum heuristic comes top in terms of memory use requiring just 86 node expansion, while A* with ignore preconditions and A* with h1 heuristics came 2nd and third respectively with 1450 and 4853 expansions required respectively.
Choosing the heuristic to use would depend on the need required. In terms of optimality, any of the solutions can be used. In terms of time A* search with ignore preconditions is the fastest. In terms of memory use regardless of the time cost, A* with level sum heuristic would be the ideal as it uses significantly lesser memory space compared to the rest. However, A* search with ignore preconditions has the best memory and time balance.

Problem 3 results

SEARCH TYPES	OPTIMAL	EXPANSIONS	PLAN LENGTH	TIME ELAPSED (SECONDS)
Astar search with h_1	YES	18223	12	169.72341979060195
Astar search with h_ignore preconditions	YES	5040	12	56.802706891129105
Astar search with h pg levelsum	YES	318	12	5928.61399482763

For problem 3, they all yielded optimal solutions (plan length of 12). In terms of time cost, A* search with ignore preconditions search yielded the best result (56.80 seconds), with A* search with h1 heuristic coming second with 169.72 seconds and A* search with levelsum heuristic coming far behind with 5928.61 seconds. However, A* search with level sum heuristic comes top in terms of memory use requiring just 318 node expansion, while A* with ignore preconditions and A* with h1 heuristics came 2nd and third respectively with 5040 and 18223 expansions required respectively.
Choosing the heuristic to use would depend on the need required. In terms of optimality, any of the solutions can be used. In terms of time A* search with ignore preconditions is the fastest. In terms of memory use regardless of the time cost, A* with level sum heuristic would be the ideal as it uses significantly lesser memory space compared to the rest. However, A* search with ignore preconditions has the best memory and time balance.

COMPARING INFORMED AND UNINFORMED SEARCH STRATEGIES.
Only Breadth first search, uniform cost search and the A* searches always yielded optimal solutions. However, In terms of speed A* with ignore preconditions was the best (being the fastest of all optimal solutions in problems 2 and 3). In terms of memory use, of all the searches that generated optimal solutions, A* with level sum search was the best in all the three problems it however lagged behind significantly in terms of time.
I would say the best search strategy to use to generate optimal solutions while trying to balance out the cost of time and memory is between breadth first search and A* search with ignore preconditions heuristic.

Problem 1 Result.

SEARCH TYPES	**OPTIMAL**	**EXPANSIONS**	**PLAN LENGTH**	TIME ELAPSED (SECONDS)
Breadth first search	Yes	43	6	0.13526131078509163
Astar search with h_ignore preconditions	YES	41	6	0.15018866575077006

Problem 2 Result.

SEARCH TYPES	**OPTIMAL**	**EXPANSIONS**	**PLAN LENGTH**	TIME ELAPSED (SECONDS)
Breadth first search	YES	3343	9	48.59207630227439
Astar search with h_ignore preconditions	YES	1450	9	15.622997963705473

Problem 3 Result.

SEARCH TYPES	**OPTIMAL**	**EXPANSIONS**	**PLAN LENGTH**	TIME ELAPSED (SECONDS)
Breadth first search	YES	14663	12	319.3527341658095
Astar search with h_ignore preconditions	YES	5040	12	56.802706891129105

However, it is evident that A* with ignore preconditions is the best search strategy that would give optimal solution with good time and memory used balance making it the best search strategy for the problem.