Planning Search Agent Analysis

This document provides analysis of a planning search agent written to solve air cargo transportation planning problem. The agent defines the planning problem as a search problem with initial state, actions function, result function for a given action and a goal test. The search uses uninformed planning algorithms and planning graph with automatic domain independent heuristics functions.

Air Cargo Planning Problem

The agent solves three problems in the air cargo domain that have the same below defined action schema but different initial states and goal.

Action Schema

```
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))
```

	Problem 1	Problem 2	Problem 3
Initial State	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))	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))	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	Goal(At(C1, JFK) ∧ At(C2, SFO))	Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))	Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
Optimum Plan Length	6	9	12

Optimum Plan	Load(C1, P1, SFO)
	Load(C2, P2, JFK)
	Fly(P2, JFK, SFO)
	Unload(C2, P2, SFO)
	Fly(P1, SFO, JFK)
	Unload(C1, P1, JFK)
		Load(C3, P3, ATL)
		Fly(P3, ATL, SFO)
		Unload(C3, P3, SFO)
		Load(C2, P2, JFK)
		Fly(P2, JFK, SFO)
		Unload(C2, P2, SFO)
		Load(C1, P1, SFO)
		Fly(P1, SFO, JFK)
		Unload(C1, P1, JFK)
			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(C2, P2, SFO)
			Unload(C1, P1, JFK)

Uninformed Planning algorithms results analysis

The uninformed search algorithms reach a solution by generating successors until the goal state is reached. These do not have any additional information than the state defined in the search problem. The below table provides the comparison of the results of seven such search algorithms in terms of execution speed (time, measured in seconds), memory usage (number of nodes expanded) and optimality (solution plan length)

The best result in each category for each problem is in bold.

Problem	Algorithm	Path Length	Execution time(secs)	Node Expansions
Problem 1	Breadth First Search	**6**	0.05487	43
	Breadth First Tree Search	**6**	2.03110	1458
	Depth First Graph Search	20	0.0272	21
	Depth limited Search	50	0.1737	101
	Uniform Cost Search	**6**	0.1063	55
	Recursive Best First Search	**6**	5.7442	4229
	Greedy Best First Graph Search	**6**	**0.00875**	**7**
Analysis:	Recursive best first search is the least performant algorithm as it takes the most memory and the most time to complete the search. Depth first graph search and depth limited search are fast but not optimal as the path length is much higher than the optimal path length of 6 Greedy best first graph is the fastest with the least no of nodes expanded. Breadth First and Uniform cost search results are optimal in terms of path length, execution time and node expansions.			

For problem 2, the execution time exceeded 10 minutes for Breadth First Tree search and Recursive Best First Search and the search was stopped

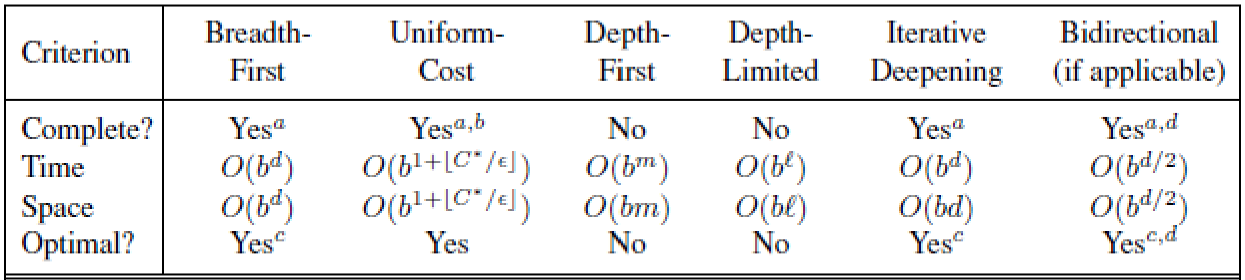
Problem	Algorithm	Path Length	Execution time(secs)	Node Expansions
Problem 2	Breadth First Search	**9**	24.8778	3343
	Breadth First Tree Search	**-**	-	-
	Depth First Graph Search	619	6.60002	**624**
	Depth limited Search	50	2242.8109	222719
	Uniform Cost Search	**9**	32.6143	4853
	Recursive Best First Search	**-**	-	-
	Greedy Best First Graph Search	15	**4.1158**	998
Analysis:	Depth first graph search is fast enough but not optimal as the path length is much higher than the optimal path length of 9 Greedy best first graph is the fastest and less nodes expanded but not optimal with path length 15. Breadth First and Uniform cost search results are optimal in terms of path length, execution time and node expansions.			

For problem 3, the execution time exceeded 10 minutes for Breadth First Tree search, Depth first Graph search, Depth limited search, and Recursive Best First Search and the search was stopped.

Problem	Algorithm	Path Length	Execution time(secs)	Node Expansions
Problem 3	Breadth First Search	**12**	175.0502	14663
	Breadth First Tree Search	**-**	-	-
	Depth First Graph Search	-	-	**-**
	Depth limited Search	-	-	-
	Uniform Cost Search	**12**	94.2784	18236
	Recursive Best First Search	**-**	-	-
	Greedy Best First Graph Search	21	**27.5472**	**5622**
Analysis:	Greedy best first graph is the fastest but not optimal with path length 21. Breadth First and Uniform cost search results are optimal in terms of path length, execution time and node expansions.			

Uninformed Search results Analysis:

From the above 3 problem search results, Breadth First and Uniform cost search yield the optimum path length result within the 10 minutes time. If the optimal path length is critical, then the **Breadth First search is recommended as it yields better results – faster and less memory than uniform cost search and it is a complete solution**.
Breadth first search requires high memory if branching factor is high and time is also a major factor.
When all step costs are equal, breadth-first search is optimal because it always expands the shallowest unexpanded node and Uniform-cost search does not care about the *number* of steps a path has, but only about their total cost. Uniform-cost search examines all the nodes at the goal's depth to see if one has a lower cost; thus uniform-cost search does strictly more work by expanding nodes at depth d unnecessarily
Greedy best first graph is the fastest and less memory in all three search results. Since the path length is not significantly higher, it is recommended if optimum path length is not too critical.



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Informed (Heuristic) Planning algorithms results analysis

Informed search strategy uses problem-specific knowledge beyond the definition of the problem itself and can find solutions more efficiently than an uninformed strategy.
In this section, we compare the performance of A* search using three different heuristic functions.

Problem	Algorithm	Path Length	Execution time(secs)	Node Expansions
Problem 1	A* Search with H1 heuristic	**6**	**0.1321**	55
	A* Search with Ignore Preconditions heuristic	**6**	0.2727	41
	A* Search with PG level sum heuristic	**6**	3.4317	**11**
Problem 2	A* Search with H1 heuristic	**9**	26.2087	4853
	A* Search with Ignore Preconditions heuristic	**9**	**9.3224**	1450
	A* Search with PG level sum heuristic	**9**	368.2724	**86**
Problem 3	A* Search with H1 heuristic	**12**	86.9553	18236
	A* Search with Ignore Preconditions heuristic	**12**	**29.3072**	5040
	A* Search with PG level sum heuristic	**12**	1757.6892 (Beyond 10 mins)	**318**

Analysis:
A* Search with PG level sum heuristic for Problem 3, did not return results within the 10 minute period so we cannot consider it due to time requirement, although it returns an optimal solution with the least memory usage.
Of the remaining two, **A* Search with Ignore Preconditions** is the fastest with less memory usage and so it has better performance than A* Search with h1 heuristic.

Informed vs Uninformed Search

Below is the comparison of the best uninformed and informed search results:

Problem	Algorithm	Path Length	Execution time(secs)	Node Expansions
Problem 1	Breadth First Search	**6**	**0.1321**	55
	A* Search with Ignore Preconditions heuristic	**6**	0.2727	**41**
Problem 2	Breadth First Search	**9**	26.2087	4853
	A* Search with Ignore Preconditions heuristic	**9**	**9.3224**	**1450**
Problem 3	Breadth First Search	**12**	86.9553	18236
	A* Search with Ignore Preconditions heuristic	**12**	**29.3072**	**5040**

From the above results, **A* Search with ignore Preconditions** performed the best with the fastest execution time and less memory usage.

Conclusion:

Informed search strategy performs better than uninformed search strategy. Heuristic functions are the most common form in which additional knowledge of the problem is imparted to the search algorithm. A relaxed heuristic is faster and depending on the trade-off that can be made in terms of speed, memory usage and path length we have the flexibility of using different heuristic functions.
Among optimal algorithms of the type that extend search paths from the root and use the same heuristic information - **A∗ is optimally efficient** for any given consistent heuristic. That is, no other optimal algorithm is guaranteed to expand fewer nodes than A*