HEURISTIC ANALYSIS FOR DOMAIN INDEPENDENT PLANNER

AIR-CARGO PROBLEM STATEMENTS & OPTIMAL SOLUTIONS

Problem 1

Problem Statement

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))

Optimal Plan Solution Sequence

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

Problem Statement

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))

Optimal Plan Solution Sequence

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)

HEURISTIC ANALYSIS FOR DOMAIN INDEPENDENT PLANNER

Problem 3

Problem Statement

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))

Optimal Plan Solution Sequence

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)

HEURISTIC ANALYSIS FOR DOMAIN INDEPENDENT PLANNER

SEARCH STRATEGIES RESULT COMPARISON

	AC Problem 1				
Search Algo | Heuristic (optional)	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (sec)
breadth first search	43	56	180	6	0.0706
breadth first tree search	1458	1459	5960	6	0.7893
depth first graph search	21	22	84	20	0.0301
depth limited search	101	271	414	50	0.1718
uniform cost search	55	57	224	6	0.0744
recursive best first search | h1	4229	4230	17023	6	1.8194
greedy best first graph search | h1	7	9	28	6	0.0144
a* search | h1	55	57	224	6	0.0708
a* search | h_ignore_preconditions	55	57	224	6	0.0969
a* search | h_pg_levelsum	18	20	77	6	0.4058

The results for the first problem indicate the optimal solution to the problem is discovered by all search strategies except for **depth first graph search** and **depth limited search**. With respect to the time taken, we find **greedy best first graph search** to be the fastest search algorithm for this given problem. We can also observe that it uses the least number of node expansions, goal tests as well as new-node creation. In lieu of the same, we can conclude that greedy BFS is the appropriate search algorithm pertaining to this problem. We can see that regular BFS also offers an optimal solution with a reasonable number of node expansions thereby making its results comparison comparable to that of the greedy BFS search for this problem. DFS on the other hand is the second most time-consuming search with non-optimal results. Comparing the automated heuristics used in A* search, both *ignore_preconditions* and *levelsum* yield the optimal solution. The reason for *levelsum* could be the relatively limited search space that this problem requires, as well as the fact that the subgoal independence condition holds for this problem and it is quite decomposable – which makes it work well in this case. Due to its admissible nature, an optimal is expected when using *ignore_preconditions*

HEURISTIC ANALYSIS FOR DOMAIN INDEPENDENT PLANNER

	AC Problem 2				
Search Algo | Heuristic (optional)	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (sec)
breadth first search	3343	4609	30509	9	4.6533
breadth first tree search	N.A.	N.A.	N.A.	N.A.	N.A.
depth first graph search	624	625	5602	619	0.8956
depth limited search	N.A.	N.A.	N.A.	N.A.	N.A.
uniform cost search	4853	4855	44041	9	3.6913
recursive best first search | h1	N.A.	N.A.	N.A.	N.A.	N.A.
greedy best first graph search | h1	998	1000	8982	21	0.9001
a* search | h1	4853	4855	44041	9	3.4939
a* search | h_ignore_preconditions	4853	4855	44041	9	5.2572
a* search | h_pg_levelsum	1524	1526	14305	9	37.0416

The results for this problem indicate **BFS** as the ideal search strategy, wherein even though it has a slightly higher runtime relative to UCS, it uses a lower number of node expansions, goal tests and significantly lower count of new nodes. The search strategies yielding non-optimal solutions for this problem are **depth first graph search** and **greedy best first graph search** with breadth first tree search, depth limited search and recursive best first search having prohibitively high runtimes. Analyzing the automated heuristics used by A* search, both *ignore_preconditions* and *levelsum* yield optimal solutions with *levelsum* requiring a far higher runtime than any of the other search algorithms. This is because it requires constructing a new planning graph with every expansion which is very inefficient – thereby making it extremely slow even though the number of node expansions is on the lower side relative to the other search algorithms. However, it does produce an optimal solution due to the subgoal independence being met and the problem being decomposable. As expected, *ignore_preconditions* produces an optimal solution due to it being an admissible heuristic

HEURISTIC ANALYSIS FOR DOMAIN INDEPENDENT PLANNER

	AC Problem 3				
Search Algo | Heuristic (optional)	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (sec)
breadth first search	14663	18098	129631	12	15.7597
breadth first tree search	N.A.	N.A.	N.A.	N.A.	N.A.
depth first graph search	408	409	3364	392	0.4452
depth limited search	N.A.	N.A.	N.A.	N.A.	N.A.
uniform cost search	18223	18225	159618	12	12.1730
recursive best first search | h1	N.A.	N.A.	N.A.	N.A.	N.A.
greedy best first graph search | h1	5579	5581	49159	22	4.0348
a* search | h1	18223	18225	159618	12	12.3550
a* search | h_ignore_preconditions	18223	18225	159618	12	19.3824
a* search | h_pg_levelsum	11261	11263	99869	13	403.2716

The results for the third problem indicate the ideal search strategy to be **BFS** which while slightly slower than UCS uses a lower number of node expansions. In addition, **depth first graph search**, **greedy best first search** and **a* w/t levelsum** yield non-optimal solutions with non-deterministic results for breadth-first tree search, depth limited search and recursive best first search which can be treated as non-optimal search strategies as well due to their prohibitively long runtimes. DFS appears to have the minimum number of node expansions, goal-tests and instances of new-node creation in addition to an exceptionally short runtime – however, it is impractical due to the extremely inefficient solution it yields (it has the highest path length associated with it w.r.t any of the other search algorithms). Analyzing the automated heuristics for A* search in this problem show us that *ignore_preconditions* continues to yield optimal solutions in a reasonable amount of time due to its admissible nature. On the other hand, *levelsum* fails for this problem with a non-optimal solution derived as well as a very high runtime. This is expected since *levelsum* is an inadmissible heuristic and the subgoal independence condition is not met in this problem. In addition, the very high runtime can be explained by the same reason as that described for Problem 2. This can only be improved by optimizing graph construction which could be achieved by pre-computing & caching the nodes of the planning graph and all of the static mutexes