Artificial Intelligent Nanodegree Program
Heuristic Analysis | Project 3: Implement a Planning Search
Jeanne Sim

This project requires us to implement a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system. Solutions for this project involve using planning graph and automatic domain-independent heuristics with A* search. Thereafter, a comparison of results and performance is done amongst the uninformed non-heuristic search methods.

Provide an optimal plan for Problems 1, 2, and 3.

- Problem 1:

Initial States	Optimal Plan (Plan length: 6)
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))	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:

Initial States	Optimal Plan (Plan length: 9)
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))	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:

Initial States	Optimal Plan (Plan length: 12)
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))	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(C2, P2, SFO) Unload(C1, P1, JFK)

Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3.

- Problem 1:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes	Remark
Breadth First Search	6	0.0544 sec	43	56	180	Optimal
Breadth First Tree	6	1.62 sec	1,458	1,459	5,960	Optimal
Depth First Graph	12	0.02 sec	12	13	48	Not optimal
Depth Limited Search	50	0.207 sec	101	271	414	Not optimal
Uniform Cost Search	6	0.0788 sec	55	57	224	Optimal
Recursive Best First Search	6	4.53 sec	4,229	4,230	17,023	Optimal, most expansive
Greedy Best First Graph Search	6	0.00897 sec	7	8	28	Most optimal, least expansive

- Problem 2:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes	Remark
Breadth First Search	9	28.7 sec	3,343	4,609	30,509	Optimal
Breadth First Tree	NA	NA	NA	NA	NA	
Depth First Graph	466	2.94 sec	476	477	4253	Not optimal, least expansive
Depth Limited Search	50	1343 sec	222,719	2,053,741	2,054,119	Optimal, most expansive
Uniform Cost Search	9	14.5 sec	4,852	4,854	44,030	Optimal
Recursive Best First Search	NA	NA	NA	NA	NA	
Greedy Best First Graph Search	21	4.036 sec	990	992	8,910	Optimal

- Problem 3:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes	Remark
Breadth First Search	12	194.9 sec	14,663	18,098	129,631	Optimal, most expansive
Breadth First Tree	NA	NA	NA	NA	NA	
Depth First Graph	1,442	19.95 sec	1,511	1,512	12,611	Not optimal
Depth Limited Search	NA	NA	NA	NA	NA	
Uniform Cost Search	12	61.03 sec	18,234	18,236	159,707	Optimal
Recursive Best First Search	NA	NA	NA	NA	NA	
Greedy Best First Graph Search	22	18.24 sec	5,605	5,607	49,360	Optimal, least expansive

Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

- Problem 1:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes
A* Search with h_1	6	0.06675 sec	55	57	224
A* Search with h_ignore_preconditions	6	0.0868 sec	41	43	170
A* Search with h_pg_levelsum	6	2.29 sec	11	13	50

- Problem 2:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes
A* Search with h_1	9	21.59 sec	4,852	4,854	44,030
A* Search with h_ignore_preconditions	9	9.821 sec	1,450	1,452	13,303
A* Search with h_pg_levelsum	9	190.97 sec	86	88	841

- Problem 3:

Search Strategy	Path Length	Execution Time	Expansions	Goal Tests	New Nodes
A* Search with h_1	12	58.62 sec	18,234	18,236	159,707
A* Search with h_ignore_preconditions	12	26.66 sec	5,040	5,042	44,944
A* Search with h_pg_levelsum	12	761.96 sec	325	327	3,002

What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

For non-heuristic search, greedy best first search works well with little nodes and uniform cost search works well with many nodes. Greedy best first search works well for simpler problem set, it explores the most promising node chosen while uniform cost search cuts the chase by finding the shortest path in a weighted path. However, heuristic search algorithms work better than faster than non-heuristic search algorithms for problems 2 and 3. The A* search algorithms expands to more nodes and explores lesser nodes at a shorter time. For less complex problems, I will use greedy best first search; for more complex problems, I will use A* search with h_ignore_preconditions.