PLANNING HEURISTIC ANALYSIS

Project Scope: Define a group of problems in classical PDDL (Planning Domain Definition Language) for the Air Cargo transport system discussed in the lectures.

The computer that I am using for this project is a 2016 MacBook Pro with the following specs:

- 2.4 GHz Intel Core i7
- 16 GB RAM

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))
```

SOLUTION

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

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))
```

SOLUTION

```
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

```
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))
```

SOLUTION

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

PART 1: PLANNING PROBLEMS

The table below gives the results from running the uninformed search using Breadth-First Search, Depth-First Search, and Uniform-Cost Search

Problem	Search Type	Number of Expansions	Number of Goal Tests	Plan Length	Time Elapsed (seconds)	Optimal
1	Breadth-First	43	56	**6**	0.02814	**Yes**
	Depth-First	**21**	22	20	**0.01228**	No
	Uniform Cost	55	57	**6**	0.03397	**Yes**

2	Breadth-First	3343	4609	**9**	12.48966	**Yes**
	Depth-First	**624**	625	619	**3. 17561**	No
	Uniform Cost	4853	4855	**9**	10.79940	**Yes**
3	Breadth-First	14415	17516	**12**	87.96976	**Yes**
	Depth-First	**1119**	1120	805	**2 .69938**	No
	Uniform Cost	17263	17265	**12**	45.57594	**Yes**

From the table above, we see that the optimal search types for this experiment of finding the optimal plans are BFS and uniform-cost search, giving the optimal plan length for Problems 1, 2, and 3 as 6, 9, and 12, respectively. This should be no surprise to us since it was learned from the lectures that BFS and uniform-cost search are complete and optimal. We can also observe that both type of searches is very similar in performance. As mentioned in Norvig and Russell's textbook, because all step costs are the same, *"uniform-cost search is similar to BFS, except that the latter stops as soon as it generates a goal, whereas 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."* (Norvig and Russell 85) And we can observe this performance difference by looking in the column of Number of Expansions. We can see that uniform-cost search expands more nodes than BFS, however, we also see that uniform-cost arrives at a solution much faster than BFS.

When it comes to space and time complexity, DFS trumps both searches but with the expense of optimality. As we see in the table, DFS had the least number of expanded nodes and had the fastest execution times for all three problems. In the textbook, the authors mention that DFS *"needs to store only a single path from the root to a leaf node, along with the remaining unexpanded sibling nodes for each node on the path. Once a node has been expanded, it can be removed from memory as soon as all its descendants have been fully explored. For a state space with branching factor b and maximum depth m, depth-first search requires storage of only* ***O(bm)*** *nodes."* (Norvig and Russell 87)

PART 2: DOMAIN-INDEPENDENT HEURISTICS

The table below gives the results from running the informed search using Breadth A* searches with their respective heuristics.

Problem	Heuristic	Number of Expansions	Number of Goal Tests	Plan Length	Time Elapsed (seconds)	Optimal
1	h1	55		6	**0.03455**	**Yes**
	ignore preconditions	41		6	0.04566	**Yes**
	level-sum	**36**		6	1.43518	**Yes**
2	h1	4853		9	10.80255	**Yes**
	ignore preconditions	1450		9	**3.84257**	**Yes**
	level-sum	**194**		9	370.52816	**Yes**
3	h1	17263		12	43.79517	**Yes**
	ignore preconditions	**7593**		13	**21.73347**	**Yes**
	level-sum	—		—	—	—

After analyzing the results, we can observe that A* search is optimal. For the heuristic performance comparison, we can see that the heuristic level-sum suffers greatly when it comes to execution as seen for Problem 3, no information was recorded because it had passed 10-minute limit. However, we see that in smaller Problems where we can run this heuristic that it expands the least nodes thus giving us the better space complexity. On the other hand, the heuristic ignore-prediction gives a fast execution time but with the cost of node expansions.

With this analysis, we can say that the best heuristic will depend on the domain of the problem. If we are concerned about the time complexity we can choose the heuristic ignore preconditions to satisfy the requirement, otherwise we use level-sum heuristic if the domain requires a good space complexity.

CONCLUSION

In conclusion using a domain-independent heuristic with A* search gives us an optimal solution with a better performance compared to the optimal non-heuristic search planning methods, thus making the informed search much better to use than the uninformed search.

WORKS CITED

Norvig and Russell. *Artificial Intelligence A Modern Approach*. 3rd Edition. n.d.

Udacity. "Foundations of AI: Search." *Artificial Intellignece Nandodegree*. 2017. Video Lecture.