Heuristic Analysis for Planning Search

In this project, we were asked to solve deterministic logistics planning problems for an Air Cargo transportation system using a planning search agent. We computed optimal plans by using progression search algorithms with domain-independent heuristics instead of simple distance heuristics. This paper will include some discussions on discovering optional heuristics for planning search problems by comparing heuristic search result matrices (optimality, time elapsed and number of node expansions).

Data Collection:
The following table will show the performances of each search agent for each set of problems.

Problem 1

	Expansions	Goal Tests	New Nodes	Optimal	Time Elapsed
BFS	43	56	180	T	0.0442
DFS	12	13	48	F	0.0185
Uninform_cost_search	55	57	224	T	0.0479
A*_search h_1	55	57	224	T	0.0481

Problem 2

	Expansions	Goal Tests	New Nodes	Optimal	Time Elapsed
BFS	3343	4609	30509	T	20.196
DFS	582	583	5211	F	4.285
Uninform_cost_search	4853	4855	44041	T	15.433
A*_search h_1	4853	4855	44041	T	15.697

Problem 3

	Expansions	Goal Tests	New Nodes	Optimal	Time Elapsed
BFS	14663	18098	129631	T	162.71
DFS	627	628	5176	F	8.026
Uninform_cost_search	18151	18153	159038	T	77.31
A*_search h_1	18151	18153	159038	T	75.22

Heuristic Analysis:
While testing all 3-set of problems, non-heuristic search strategies like BFS, DFS and uniform_cost_search were mainly generating inheritors and determine the goal state. By comparing these three methods, we can see that BFS performs the best due to its optimality, relatively faster speed and less memory usage. The core functionality of Breadth-first search is that it expands the shallowest nodes first, it is complete, optimal for unity step costs, but it has exponential space complexity [1]. So, BFS is not space friendly. Uniform_cost_search is more optimal than DFS, but less efficient than BFS. However, in terms of speed and memory usage, DFS handles better than other two search agents because it is the fastest and uses least memory. Besides looking at non-

heuristic search strategies, we also can find something interesting, which is that A*_search h_1 function has very similar performance as Uniform_cost_search does. They almost have the same speed and memory usage.

Experiment for different A*_search:

Problem 1:

	Expansions	Goal Tests	New Nodes	Time Elapsed
A*_search ignore_preconditions	41	43	170	0.073
A*_search pg_levelsum	11	13	50	1.339

Problem 2:

	Expansions	Goal Tests	New Nodes	Time Elapsed
A*_search ignore_preconditions	1450	1452	13303	6.23
A*_search pg_levelsum	-	-	-	-

Problem 3:

	Expansions	Goal Tests	New Nodes	Time Elapsed
A*_search ignore_preconditions	5038	5040	44926	26.63
A*_search pg_levelsum	-	-	-	-

In problem 3, performance data could not be collected for A* search levelsum, because its executing time was more than 10min. Comparing heuristic search result metrics using A* with the “ignore preconditions” and “level-sum” heuristics for 3 set of problems, we can tell that A* with the “ignore preconditions” has better performance because it finishes executing within 10min. Although A*search levelsum runs slower, it costs less memory.

Optimal solution for problem 1 to 3:

Problem 1: (Use BFS)

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

Problem 2: (Use A* Search ignore_preconditions)

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

Problem 3: (Use A* Search ignore_preconditions)

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

Conclusion

Based on the experiment data and heuristic analysis, it is observable that informed search strategies with customized heuristics are better than over uninformed search techniques. Informed search strategies are faster and use less memory. Overall for our Air Cargo problem, the best heuristic should be A* Search ignore_preconditions. According to the AIMA book, we know that neither forward nor backward search is efficient without a good heuristic function, an admissible heuristic like A* search can find optimal solutions, and it also minimize the total estimated solution cost by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal [1].

References:

1. Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition).