Jason Asaytuno
Heuristics Evaluation
AIND – Planning

Uninformed planning searches breadth-first (BFS), depth-first (DFS), and uniform-cost (UCS) search were conducted. According to Table 1, BFS and UCS expanded the most nodes, goal tests, and took the longest time to complete but offered optimal solutions. DFS took the least time, expanded the least amount of nodes and goal tests but offered non-optimal solutions. Node expansion and time completion grew exponentially as the problems increased from one to three.

BFS and UCS are optimal because it searches the frontier of the state space using a queue and priority queue respectively, ensuring the goal state nearest to the initial state is found first, ensuring optimality. Since it is searching every node at the frontier, it is expanding more nodes. DFS searches for any goal state, going down a search graph until it stumbles upon a goal state, offering a non-optimal solution while expanding fewer nodes.

Search	Expansions	Goal tests	New Nodes	Time	Solution Length	Problem
bfs	43	56	180	0.025	6	1
bfs	3,343	4,609	30,509	12.896	9	2
bfs	14,663	18,098	129,631	117.919	12	3
dfs	21	22	84	0.010	20	1
dfs	624	625	5,602	3.758	619	2
dfs	408	409	3,364	1.578	392	3
ucs	55	57	224	0.028	6	1
ucs	4,852	4,854	44,030	70.313	9	2
ucs	18,235	18,237	159,716	673.451	12	3

Table 1

Heuristic planning searches using A*: constant, level-sum, and ignore-preconditions heuristic searches were conducted. According to Table 2, node expansion and goal tests were the least for level-sum by magnitudes less than ignore-preconditions and constant heuristics, which had similar number of expansions and goal tests relative to level-sum. On the other hand, level-sum took the longest to complete, followed ignore-preconditions and the constant heuristic. All three produced optimal planning solutions.

Level-sum expanded fewer nodes because it builds a planning graph for each new state, calculating a heuristic that sums the first encountered level that holds a goal fluent. Ignore-preconditions follows second by calculating the total actions need from each state to reach a goal state if preconditions are ignored. Constant heuristic is the same with UCS, so it performs similarly. Level-sum takes longer because it is building a planning graph for each new state.

Search	Expansions	Goal tests	New Nodes	Time	Solution Length	Problem
h1	55	57	224	0.035	6	1
h1	4,852	4,854	44,030	70.161	9	2
h1	18,235	18,237	159,716	711.453	12	3

h_levelsum	11	13	50	2.208	6	1
h_levelsum	86	88	841	239.447	9	2
h_levelsum	403	405	3,708	1,505.060	12	3
h_ignore	41	43	170	0.538	6	1
h_ignore	1,506	1,508	13,820	195.819	9	2
h_ignore	5,118	5,120	45,650	1,001.509	12	3

Table 2

Optimal planning plans for air cargo problems 1, 2, and 3 are as follows:

Problem 1:
```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```
Problem 2:
```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```
Problem 3:
```
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Comparing and contrasting the non-heuristic planning results, a graph plot of the three searches is used. The length of each searches' solution set is plotted against the time to complete and node expansion in two graphs, Figure 1 and Figure 2.

In figure 1, BFS and UCS expanded a similar numbers of nodes, growing exponentially as the problem complexity increased and both gave optimal solutions. DFS node expansion does not grow as fast but its solutions are not optimal as indicated by the solution length.

UCS took the longest to complete and Figure 2 suggests a growth rate a magnitude higher and with a similar shape to BFS. DFS finished the quickest with its much more shallower growth rate but trades optimality for speed.

Comparing and contrasting the heuristic planning results, a graph plot of the three searches is used. The length of each searches' solution set is plotted against the time to complete and number of nodes expanded in two graphs, Figure 3 and Figure 4.

Figure 1

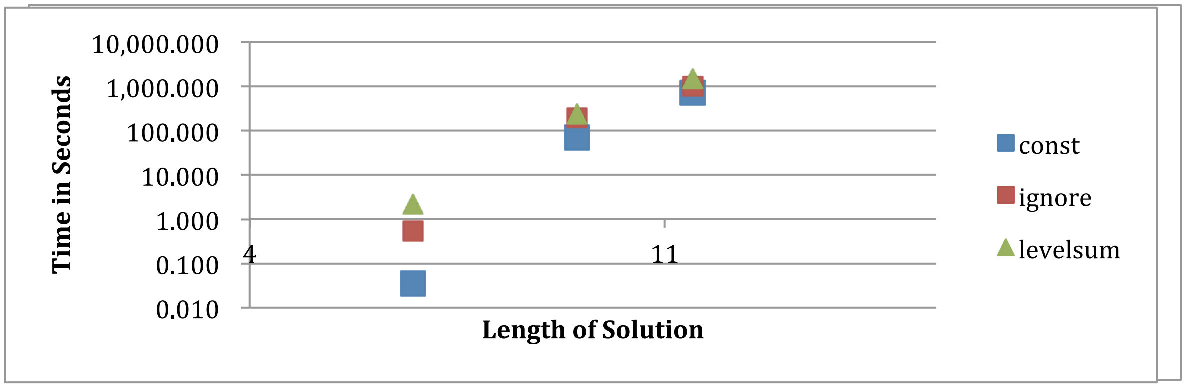

Figure 3

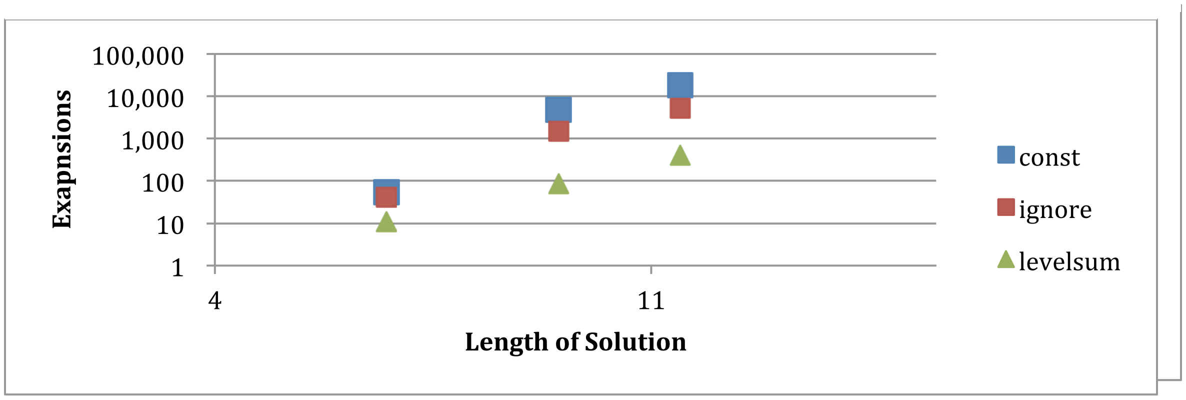

Figure 4

In figure 3, all three searches more or less complete around the same time. Constant heuristic search is the quickest in the first problem followed by level-sum and ignore-preconditions finishing a magnitude slower. As problem complexity increases, the searches seem to converge at problem 3, completing at similar times.

In figure 4, level-sum consistently expands magnitudes less nodes than constant and ignore-preconditions heuristics.

The best heuristic used is the level-sum heuristic. It expands fewer nodes to find a solution and finishes in a time not too dissimilar from the other heuristics.

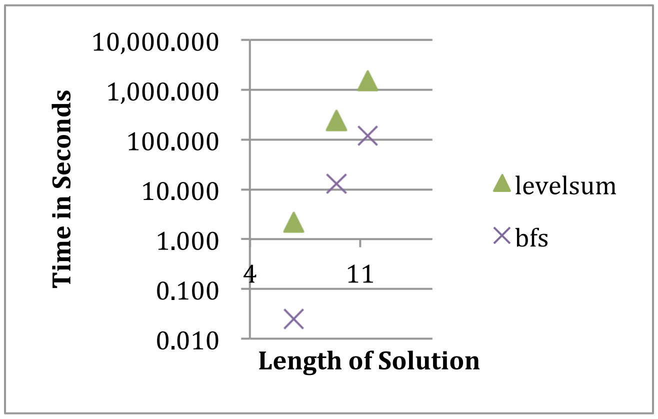

Figure 5

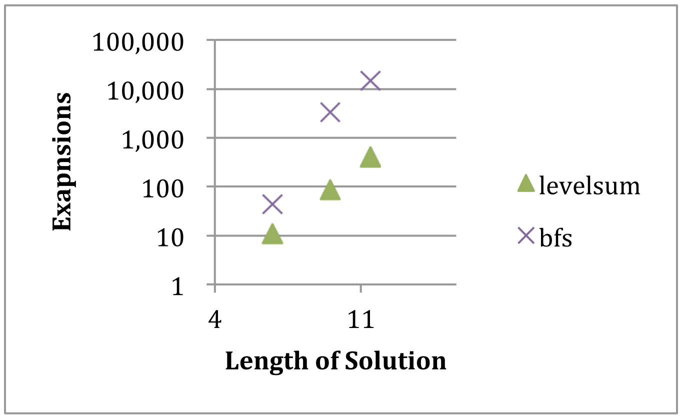

Figure 6

The non-heuristic search BFS performed better than level-sum. BFS gives an optimal solution in a magnitude less time then level-sum but expands a magnitude more nodes (Figure 5 and 6). Level-sum builds a planning graph for each new node, trading off the time to build the heuristic to expand fewer nodes. This might be an advantage for more complex problems but according to the results of problems 1, 2, & 3, BFS performed better with respect to time. The other searches are not better, DFS gives non-optimal solutions and UCS and a constant heuristic are the same search.