Mitesh Valechha

Heuristics Analysis

For Airport Cargo Transport Panning Problems

Problems

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))
```

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))
```

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))
```

Optimal Solutions

Problem 1

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

Problem 2

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

Problem 3

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

The optimality of a plan is determined by minimum length of a plan. Here for the above problems, the optimal plan length of problem 1, 2 and 3 are 6, 9 and 12 respectively.

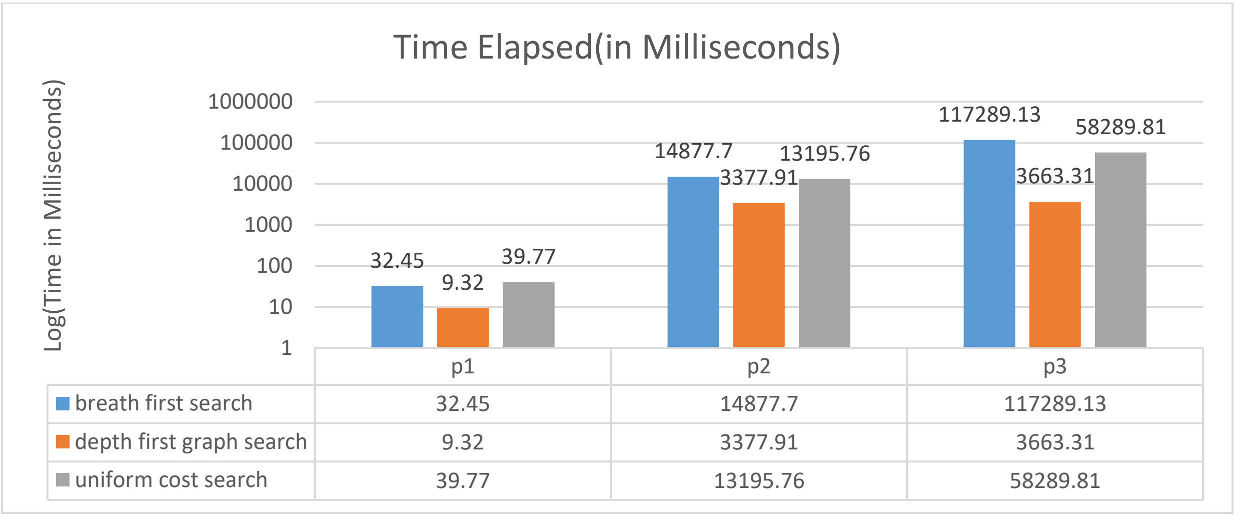
Uninformed Non-Heuristics Planning Search Results

Table 1 Metrics for Uninformed Non-Heuristic Search

Problem	Search Algorithm	Optimal	Plan Length	Time (Seconds)	Expansions	Goal Tests
p1	**breath first search**	YES	6	0.0324496	43	56
	breath first tree search	YES	6	0.99733699	1458	1459
	depth first graph search	NO	12	0.00931515	12	13
	depth limited search	NO	50	0.09353454	101	271
	uniform cost search	YES	6	0.03976806	55	57
p2	**breath first search**	YES	9	14.8777022	3343	4609
	breath first tree search	-	-	-	-	-
	depth first graph search	NO	575	3.37790923	582	583
	depth limited search	-	-	-	-	-
	uniform cost search	YES	9	13.1957629	4844	4846
p3	**breath first search**	YES	12	129.374836	14663	18098
	breath first tree search	-	-	-	-	-
	depth first graph search	NO	596	3.663312	627	628
	depth limited search	-	-	-	-	-
	uniform cost search	YES	12	64.4290743	18225	18227

The Uninformed planning search was experimented with Breadth First Search (BFS), Depth First Search (DFS) and Uniform Cost Search (UCS). As Breath First Tree Search and Depth Limited Search were skipped from comparison as they took more than 10 minutes to run on Problem 2 and Problem 3.

1. Execution Time

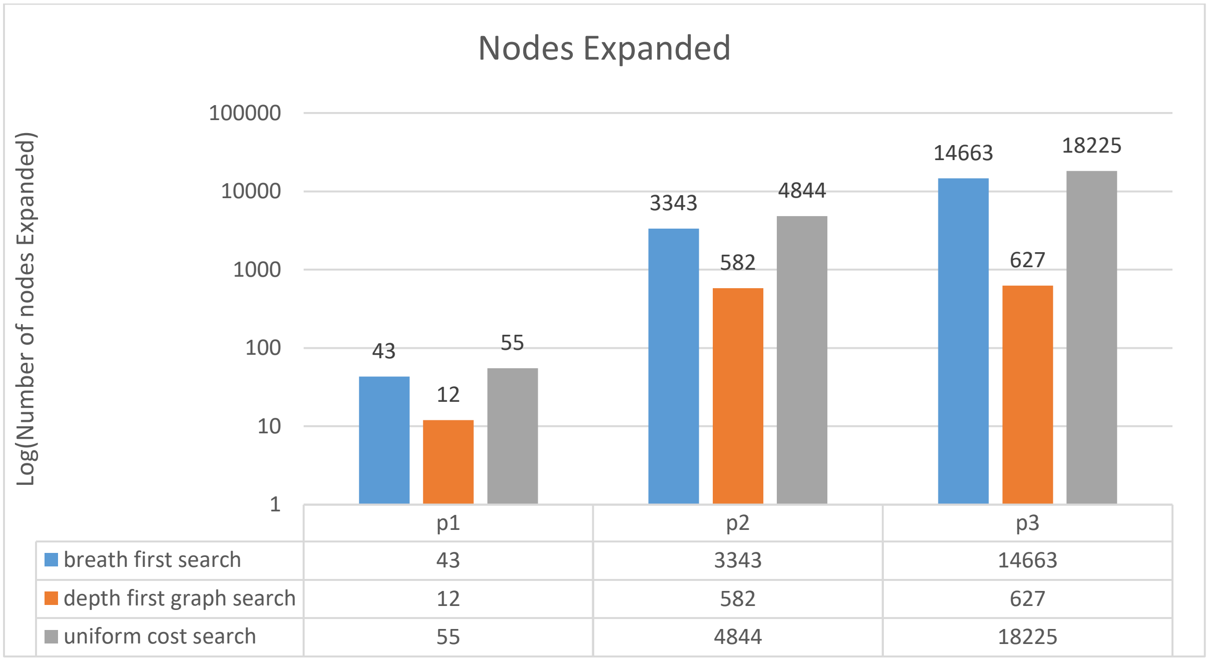


	p1	p2	p3
🟦 breath first search	32.45	14877.7	117289.13
🟧 depth first graph search	9.32	3377.91	3663.31
⬜ uniform cost search	39.77	13195.76	58289.81

Figure 1 Execution Time for Non-Heuristic Search Algorithms

From the above results it is clear that **Depth First Graph Search** takes much less time to execute and reach its goal state in comparison to Breath First Search and Uniform Cost search.

2. Number of Nodes Expanded



	p1	p2	p3
🟦 breath first search	43	3343	14663
🟧 depth first graph search	12	582	627
⬜ uniform cost search	55	4844	18225

Figure 2 No. of Nodes Expanded for Non-Heuristic Search Algorithms

From the above figure we can see that **Depth First Graph Search** expands less amount of nodes than Breath First Search and Uniform Cost Search algorithms, which also means it needs less amount of memory to execute the planning problems.

3. Plan Length and Optimality

From **Table 1** we can clearly see that Breath First Search and Uniform Cost Search have the least number of plan lengths that are 6, 9 and 12 for problems 1, 2 and 3 respectively. Although Depth First Search Algorithm performs the best in Execution Time and Node Expansion, but its plan is quite high in comparison to the other two. So both Breath First Search and Uniform Cost Search can be called Optimal plans.

Heuristics Planning Search Results

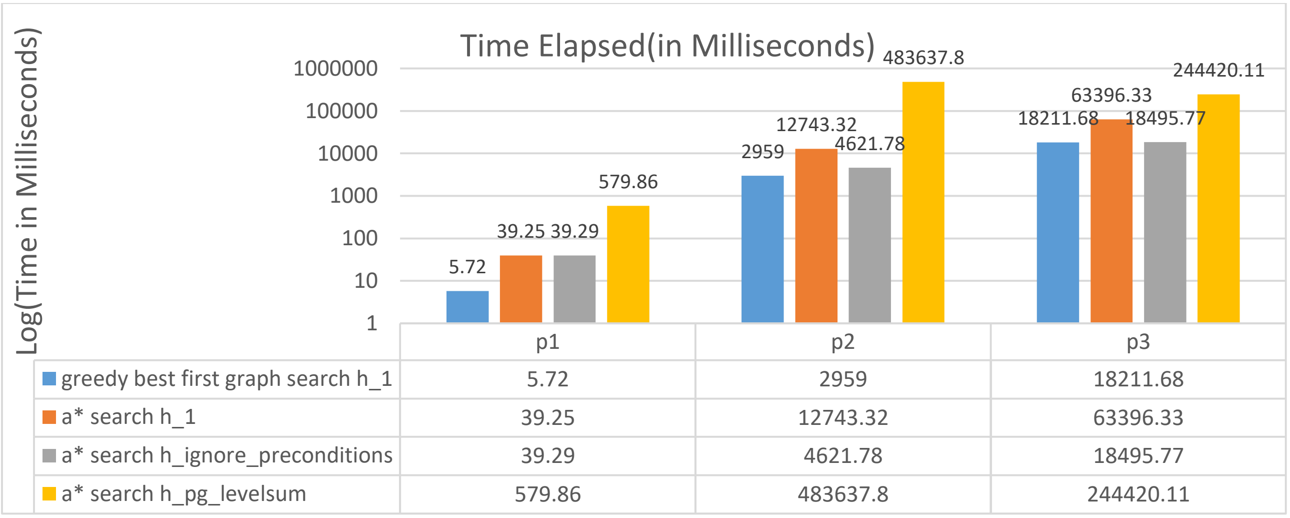
Table 1 Metrics for Heuristic Search

Problem	Search Algorithm	Optimal	Plan Length	Time	Expansions	Goal Tests
p1	recursive best first search h_1	YES	6	2.93581114	4229	4230
	greedy best first graph search h_1	YES	6	0.00571988	7	9
	a* search h_1	YES	6	0.03925145	55	57
	a* search h_ignore_preconditions	YES	6	0.03928695	41	43
	a* search h_pg_levelsum	YES	6	0.57986455	11	13
p2	recursive best first search h_1	-	-	-	-	-
	greedy best first graph search h_1	NO	23	2.95900372	974	976
	a* search h_1	YES	9	12.7433177	4844	4846
	a* search h_ignore_preconditions	YES	9	4.62178243	1445	1447
	a* search h_pg_levelsum	YES	9	48.3637799	86	88
p3	recursive best first search h_1	-	-	-	-	-
	greedy best first graph search h_1	NO	28	18.2116789	5598	5600
	a* search h_1	YES	12	63.396333	18225	18227
	a* search h_ignore_preconditions	YES	12	18.4957694	4995	4997
	a* search h_pg_levelsum	YES	12	244.420113	296	298

The Heuristic Planning Search was experimented with Greedy Best First Graph Search H_1, A* Search H_1, A* Search H_ignore_preconditions and A* Search H_pg_levelsum. Whereas Recursive Best First Search H_1 was eliminated from experiment as it took more than 10 minutes in execution for Problem 2 and 3.

1. Execution Time

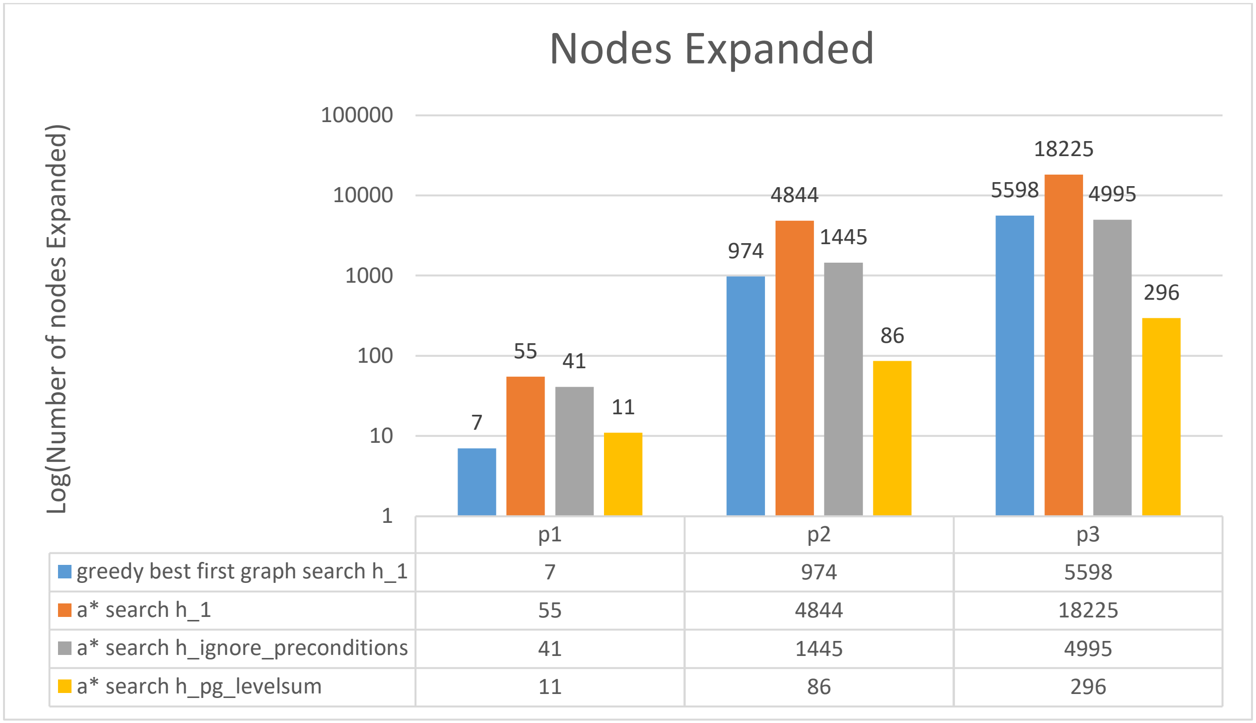
Figure 3 Execution Time for Heuristic Search Algorithms



	p1	p2	p3
🟦 greedy best first graph search h_1	5.72	2959	18211.68
🟧 a* search h_1	39.25	12743.32	63396.33
⬜ a* search h_ignore_preconditions	39.29	4621.78	18495.77
🟨 a* search h_pg_levelsum	579.86	483637.8	244420.11

From the above results it is clear that Greedy Best First Graph Search H_1 takes much less time to execute and reach its goal state in comparison to all A* Searches. But we can see that as the problem goes more complex the execution time gets reduced on A* Search H_ignore_preconditions.

2. Number of Nodes Expanded



	p1	p2	p3
🟦 greedy best first graph search h_1	7	974	5598
🟧 a* search h_1	55	4844	18225
⬜ a* search h_ignore_preconditions	41	1445	4995
🟨 a* search h_pg_levelsum	11	86	296

Figure 4 No. of Nodes Expanded for Heuristic Search Algorithms

From the above figure we can see that Greedy Best First Graph Search H_1 expands less amount of nodes than all A* Search algorithms in Problem 1, but on Problems 2 and 3 A* Search H_pg_levelsum performs the best out of four as it's the most memory efficient.

3. Plan Length and Optimality

From **Table 2** we can clearly see that all four Heuristic Search Algorithms have the least number of plan lengths that are 6, 9 and 12 for problem 1, 2 and 3 respectively, except for Greedy Best First Graph Search H_1. The plan length of all the A* Searches is 6, 9 and 12 for problems 1, 2 and 3 respectively. So A* Search H_1, A* Search H_ignore_preconditions and A* Search H_pg_levelsum are optimal to solve the above problems.

Best Overall Planning Search

From the above analysis we know that Breath First Search, Uniform Cost Search and all three A* heuristic searches are optimal because they give the minimum plan lengths for the stated problems. But if we would have only considered execution time and memory consumption but not the plan length for optimality, Depth First Graph Search is the best trade-off between execution time and memory consumed. But as we are also taking plan length into consideration we will be looking at Breath First Search, Uniform Cost Search and all three A* Heuristic Searches. A* Search H_pg_levelsum performs the best in memory usage as it expands the least amount of nodes, but it also performs the worst in execution time among all 5 optimal Search Algorithms. On the other side A* Search H_pg_levelsum takes the least time to execute the algorithm in fact it takes almost 1/3th of the time taken to execute Problem 2 and 3 by A* Search H_1 and Uniform Cost Search both of which perform about the same in execution time and are faster than Breath First Search as they take almost ½ the time taken by Breath First Search.

So we can conclude that A* Search with Ignore Preconditions Heuristic is the best among all the planning algorithms taking in account the plan length, execution time and memory usage of these algorithms. Ignore Precondition drops all preconditions that relaxes the actions. So Every action becomes applicable in every state, and any single goal fluent can be achieved in one step i.e. steps required to solve a relaxed problem is number of unsatisfied goals as per explained in Norvig and Russell's textbook "Artificial Intelligence: A Modern Approach". So that gives Ignore Precondition Heuristic to give a plan length of what least number of goals are required to complete the job and makes it the optimal for solving a planning problem.

Norvig and Russell's textbook "Artificial Intelligence: A Modern Approach"