Heuristic Analysis (Planning and Search)

Part1 Comparison of Non-Heuristic Planning Search

Table 1 Comparison Metrics for Different Search Algorithm

Algorithm	Expansions	Goal_Tests	New_Nodes	Time(s)	#Path
BFS	43	56	180	0.15	6
BFS	3343	4609	30509	35.32	9
BFS	14663	18098	129631	200.58	12
DFS	21	22	84	0.06	20
DFS	624	625	5602	8.68	619
DFS	408	409	3364	6.29	392
Uniform Search	55	57	224	0.18	6
Uniform Search	4853	4855	44041	55.32	9
Uniform Search	18223	18225	159618	252.48	12

Table[1] and Figure[1,2,3,4] below illustrate the performance metrics for three different search algorithms on air cargo problems with three different size. Overall, DFS (depth first search) completed the tasks at fastest speed because of $O(D), D = depth$ frontier space/expansion rather than $O(2^{D-1}), D = depth$ frontier space for BFS. However, DFS is not considered optimal because of its long path length. In the real situation, long path length means much higher execution cost. It makes no sense of choosing path of a sequence of 619 executions rather than a sequence of only 9. Besides, DFS will even fail when the depth becoming infinite but goal hiding in the middle.

BFS(breadth first search) and Uniform Search show the similar growth of time and space complexity when increasing problem size. However, uniform search appears to have a larger frontier of nodes and more goal tested. Since the best first search (uniform search with default path cost calculation in program) used depth as default of path cost, as such uniform cost search becomes BFS indeed. However, difference can be attributed to the time when doing goal test. Implementation of BFS is checking the goal during expanding frontier nodes, whereas uniform search is after popping from Queue (added frontier nodes). This explains why BFS has a relatively large gap between #expansion and #goal tests (early stop).

In the real case, uniform search should have a better G function defined to best estimate the path cost. Speaking of problems in this project, DFS shows advantage of space consumption and faster speed of achieving goals, yet it generates a real bad path to be utilized.

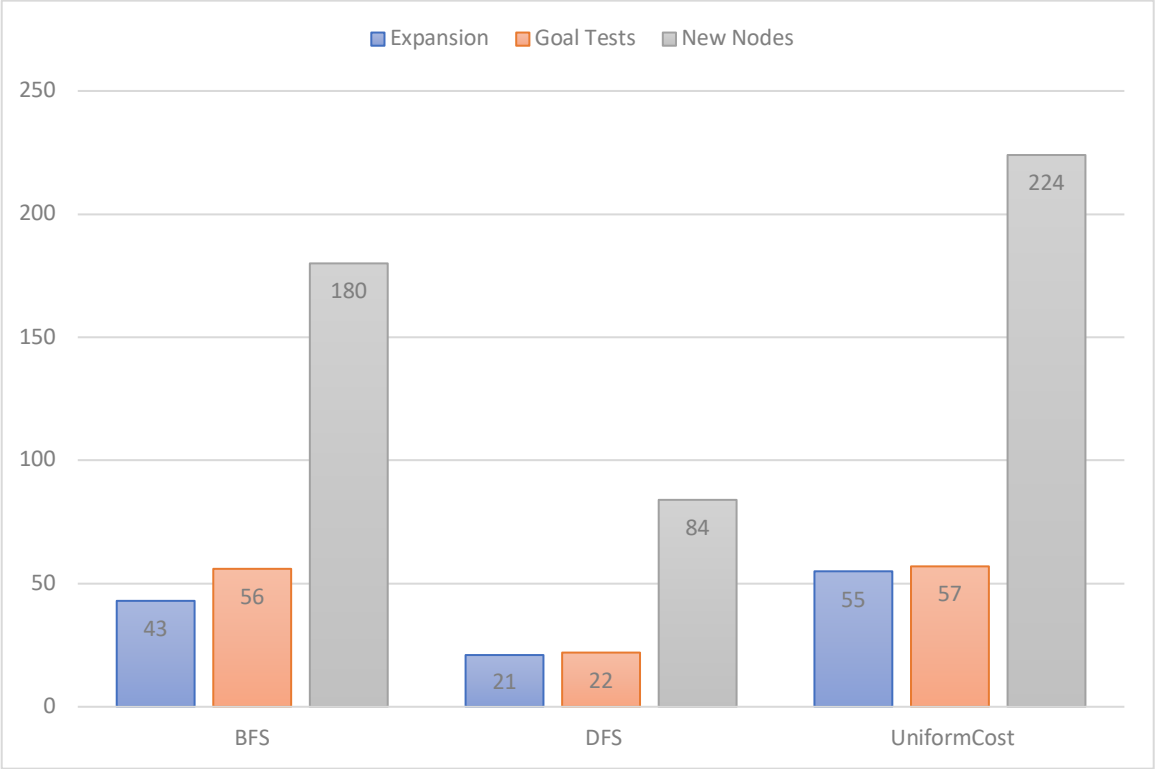

Figure 1 Metric Comparison on Problem1

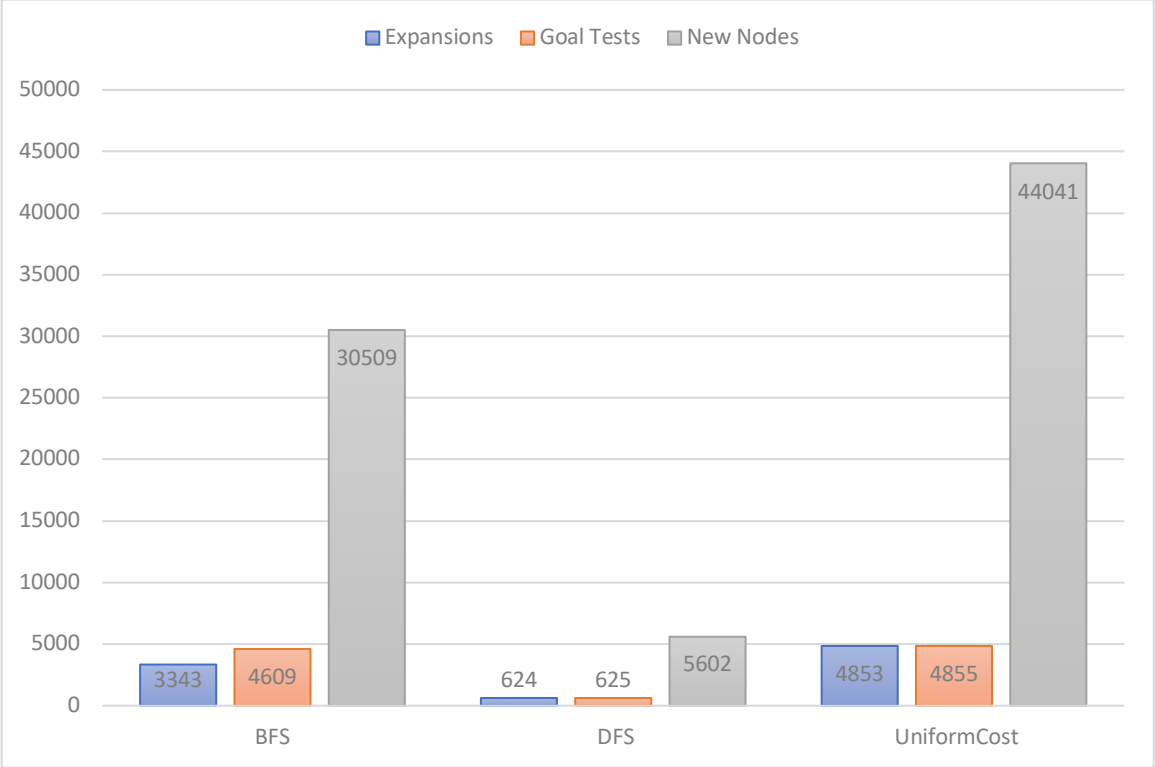

Figure 2Metric Comparison on Problem2

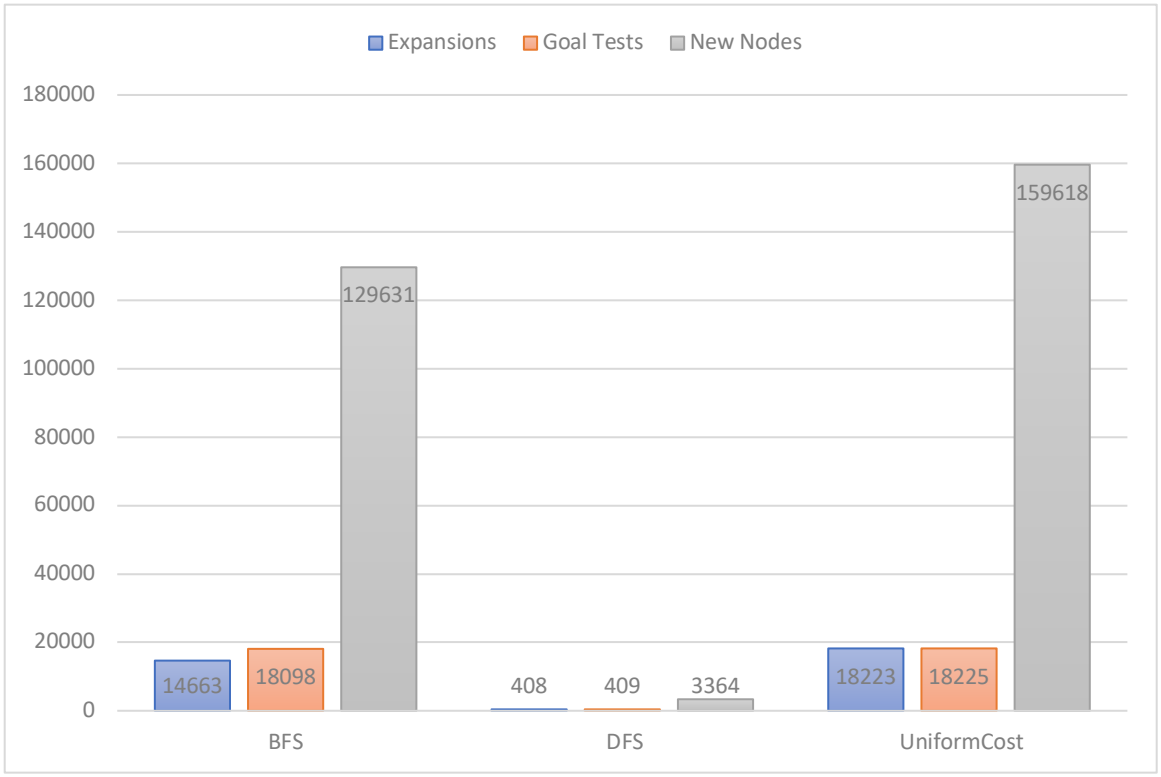

Figure 3 Metric Comparison on Problem3

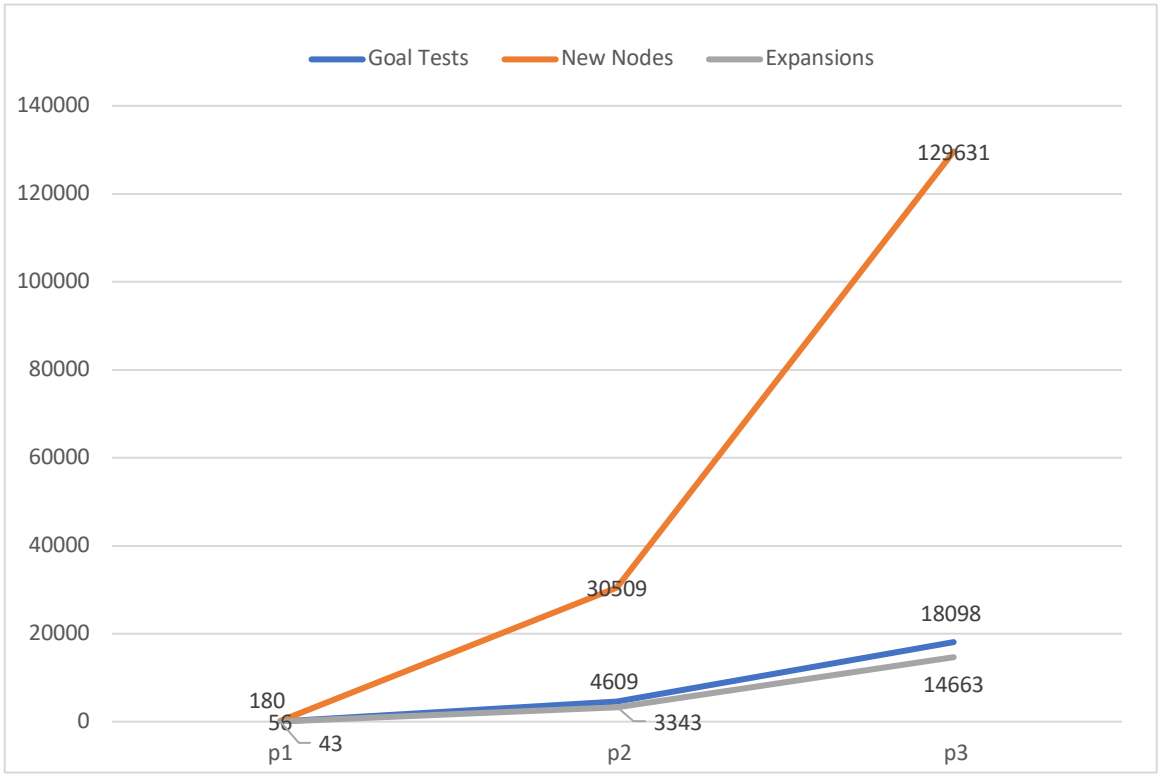

Figure 4 BFS on P1, P2, P3

Part2 Comparison of Heuristics on A star

Table 2 Comparison Metrics for Different Heuristics

Heuristic	Expansions	Goal_Tests	New_Nodes	Time(s)	#Path
h1	55	57	224	0.16	6
h1	4853	4855	44041	54.06	9
h1	18223	18225	159618	252.55	12
ignore precondition	41	43	170	0.16	6
ignore precondition	1450	1452	13303	17.34	9
ignore precondition	5040	5042	44944	69.74	12
pg levelsum	11	13	50	1	6
pg levelsum	86	88	841	72.3	9
pg levelsum	316	318	2916	364.52	12

Table[2] and Figure[5,6,7] below illustrate comparison of different heuristics on A star search algorithm. From the perspective of pure space complexity (state nodes rather than planning graph expression nodes), heuristic of level sum completely beats the other two. This reflects the strong capability of automating best heuristic by applying planning graph data structure. The search space of $O(2^n), n = \#fluent\ variable/literals$ can be greatly reduced because of prioritizing nodes of less cost. For the best case of perfect available actions/edges and best heuristic, the complexity could be reduced to linear if there is a straight path from root to goal.

However, from perspective of wall time consumption, level sum on planning graph suffered to get top because of more compute of calculating H term. We need to build an entirely leveled planning graph each time when expanding neighbors. It took an extra $O(n(a+l)^2), a = \#actions, l = \#fluents$ time complexity layer. However, such metric comparison does show how closer to true cost of admissible heuristic can shrink the searching space. In a nut shell, ignore precondition will be considered as optimal solution for air cargo problems, because the time consumption really beats others and also generated a path of the shortest length.

When comparing to non-heuristic search algorithms described earlier, not all heuristic ones win the game, i.e. h1 with BFS. However, A* with a good heuristic estimate should have a more stable performance than BFS.

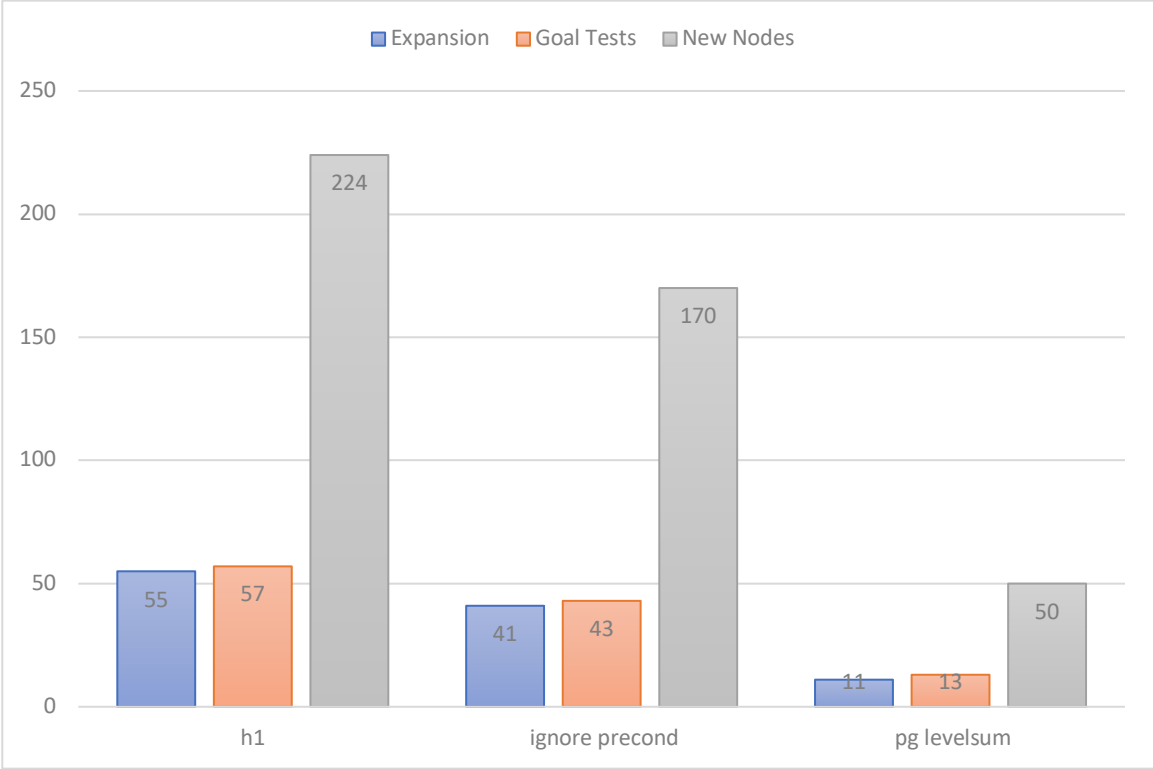

Figure 5 Metrics of heuristics on problem 1

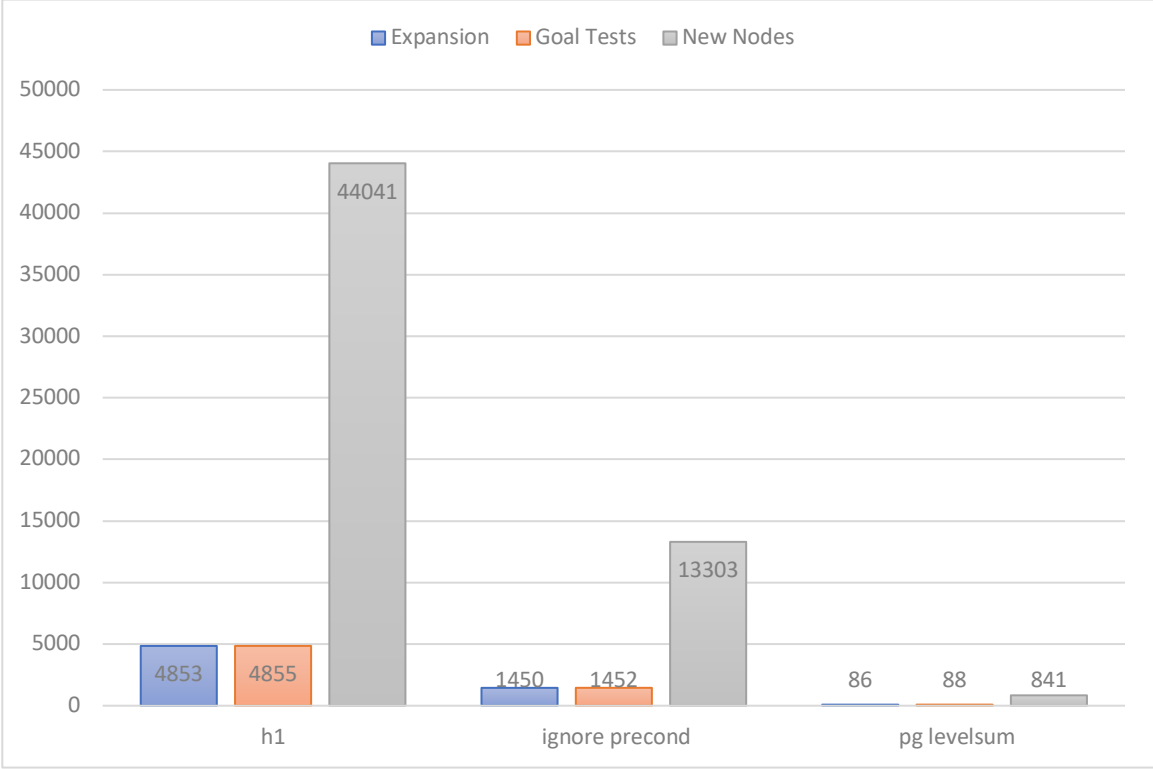

Figure 6 Metrics of heuristics on problem 2

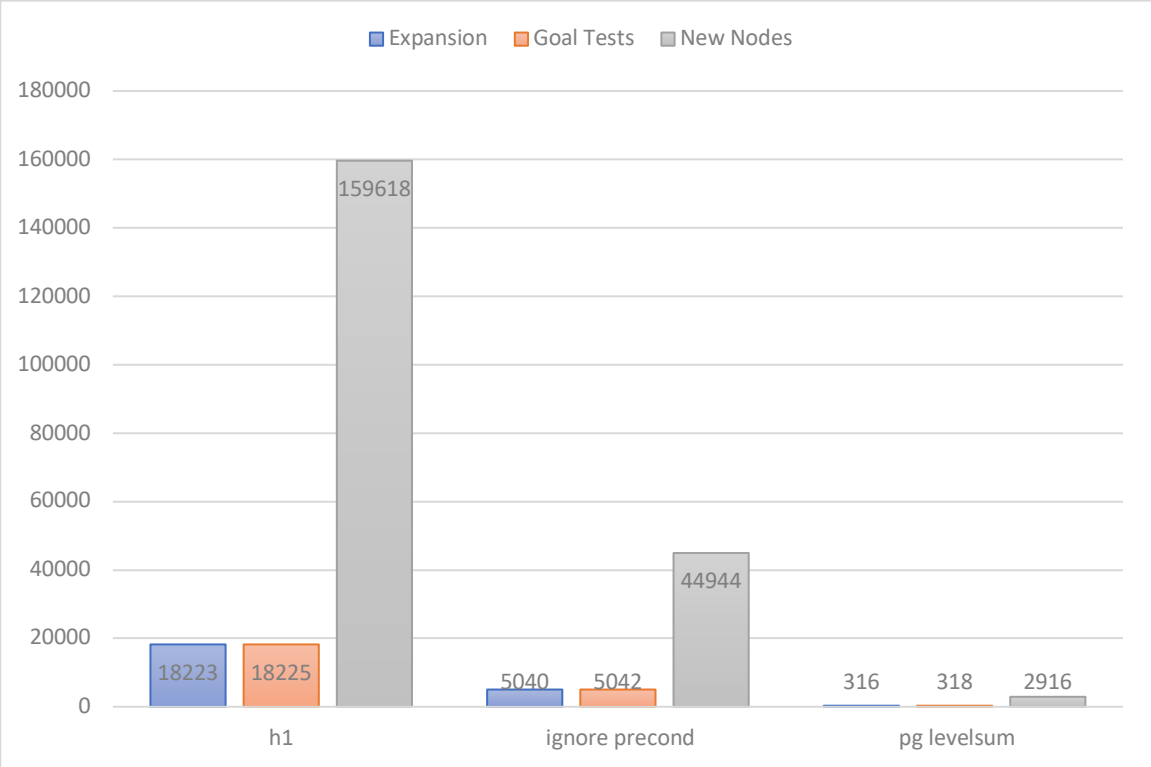

Figure 7 Metrics of heuristics on problem 3

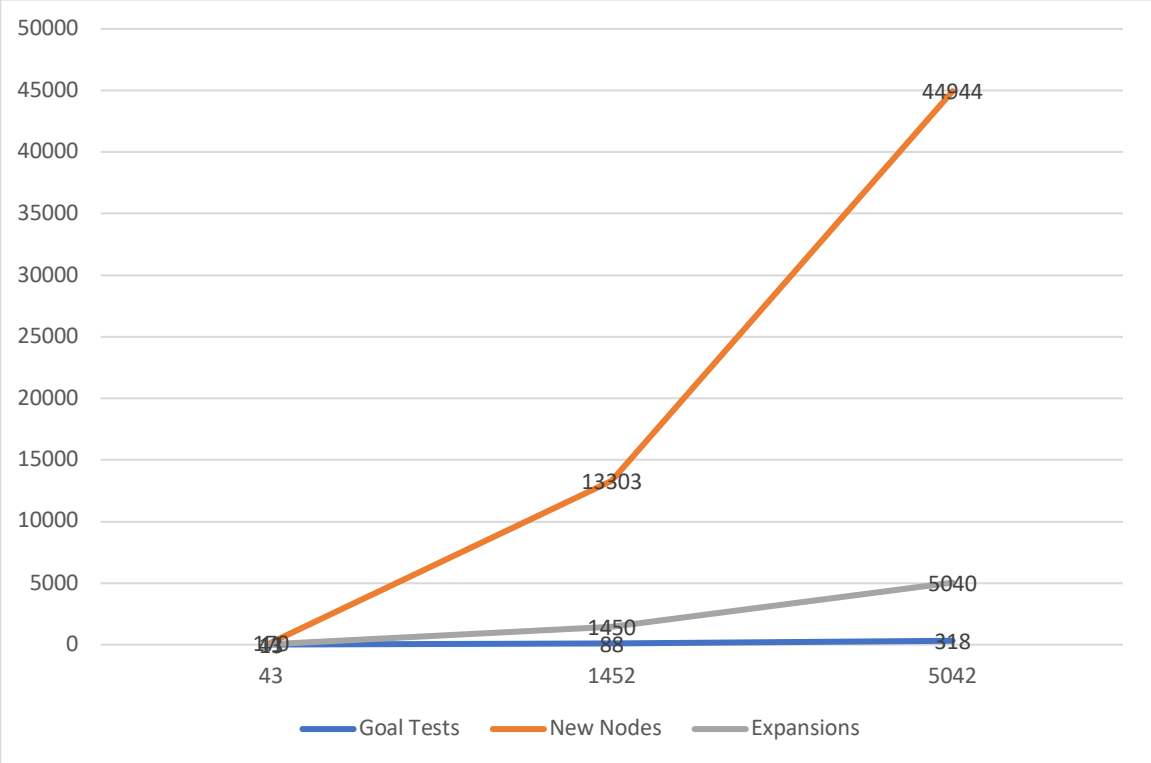

Figure 8 ignore condition on problems

Part 3

P1	P2	P3
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
Fly(P1, SFO, JFK)	Load(C2, P2, JFK)	Fly(P2, JFK, ORD)
Unload(C1, P1, JFK)	Load(C3, P3, ATL)	Load(C4, P2, ORD)
Load(C2, P2, JFK)	Fly(P1, SFO, JFK)	Fly(P2, ORD, SFO)
Fly(P2, JFK, SFO)	Fly(P2, JFK, SFO)	Load(C1, P1, SFO)
Unload(C2, P2, SFO)	Fly(P3, ATL, SFO)	Fly(P1, SFO, ATL)
	Unload(C3, P3, SFO)	Load(C3, P1, ATL)
	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
	Unload(C2, P2, SFO)	Unload(C4, P2, SFO)
		Unload(C3, P1, JFK)
		Unload(C1, P1, JFK)
		Unload(C2, P2, SFO)

Figure 9 Optimal Path for problems

Figure [9] shows the optimal paths for each problem.