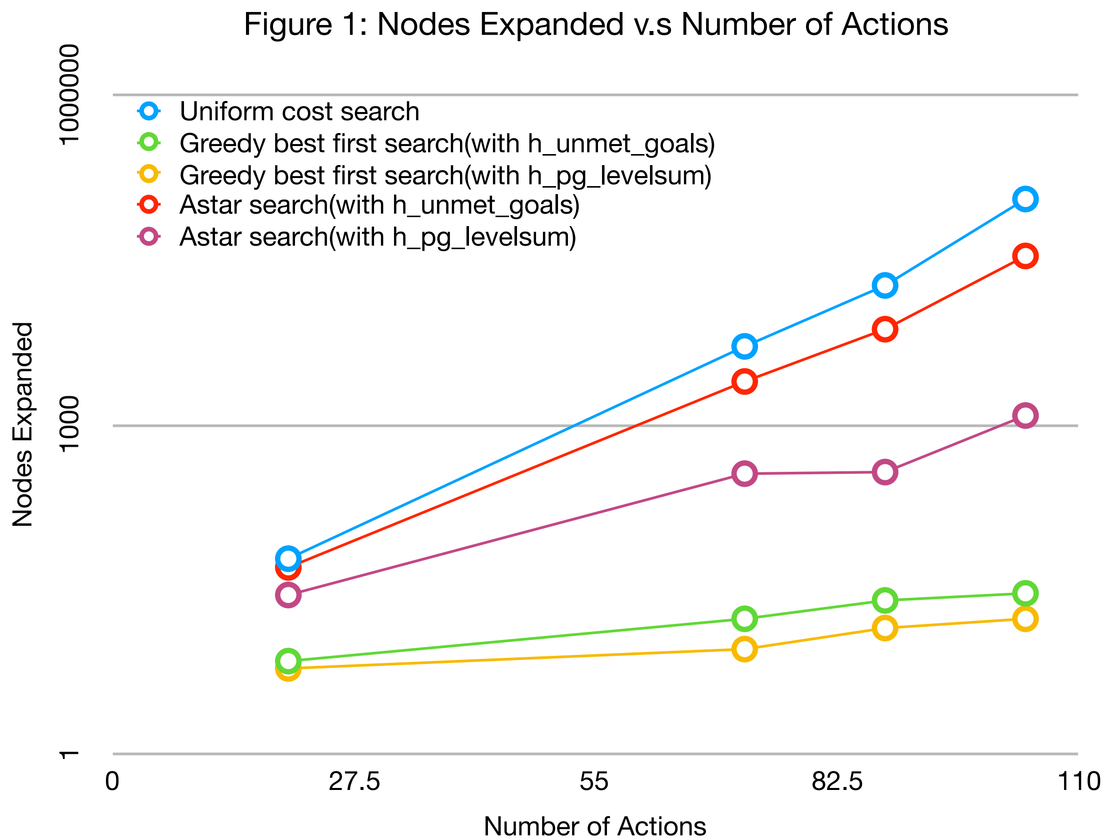
Number of Nodes

Table 1: Number of Nodes Expanded

Nodes Expanded	P1(#action 20)	P2(#action 72)	P3(#action 88)	P4(#action 104)
Breadth first search	43	3343	14663	99736
Depth first search	21	624	408	25174
Uniform cost search	60	5154	18510	113339
Greedy best first search with h_unmet_goals	7	17	25	29
Greedy best first search with h_pg_levelsum	6	9	14	17
Greedy best first search with h_pg_maxlevel	6	27	21	56
Greedy best first search with h_pg_setlevel	6	9	35	107
Astar search with h_unmet_goals	50	2467	7388	34330
Astar search with h_pg_levelsum	28	357	369	1208
Astar search with h_pg_maxlevel	43	2887	9580	N/A
Astar search with h_pg_setlevel	33	1037	3423	N/A

As shown in Table 1 and Figure 1, as the number of action increase, the number of nodes expanded increase exponentially, with rates varying among different algorithms and heuristic functions.

Among all algorithms in this experiment, greedy best first searches with heuristic functions have the smallest rate. Uninformed searches(Breath first search, Depth first search, Uniform cost search) have the largest rate while Astar searches are in the middle. This makes sense as the informed searches(greedy and Astar with heuristic estimations) tend to have less nodes expanded because many nodes are skipped. Greedy best first search performs better at the number of nodes expanded simply because greedy method don’t need to find the optimal solution.



Among all heuristic functions that are admissible, “set-level” usually provides the lest number of nodes expanded as its estimation of the cost from current state to the goal is the closest to the ground truth. “unmet-goals” and “max-level” are tied very closely in the performance regarding number of nodes expanded. One thing interesting is about “level-sum” heuristic function. It gives the smallest number of nodes expanded as it seems that its estimation of cost come very close to the ground truth. However “level-sum” is not admissible, which means it does not guarantee that Astar search finds the optimal solution as we could see later.

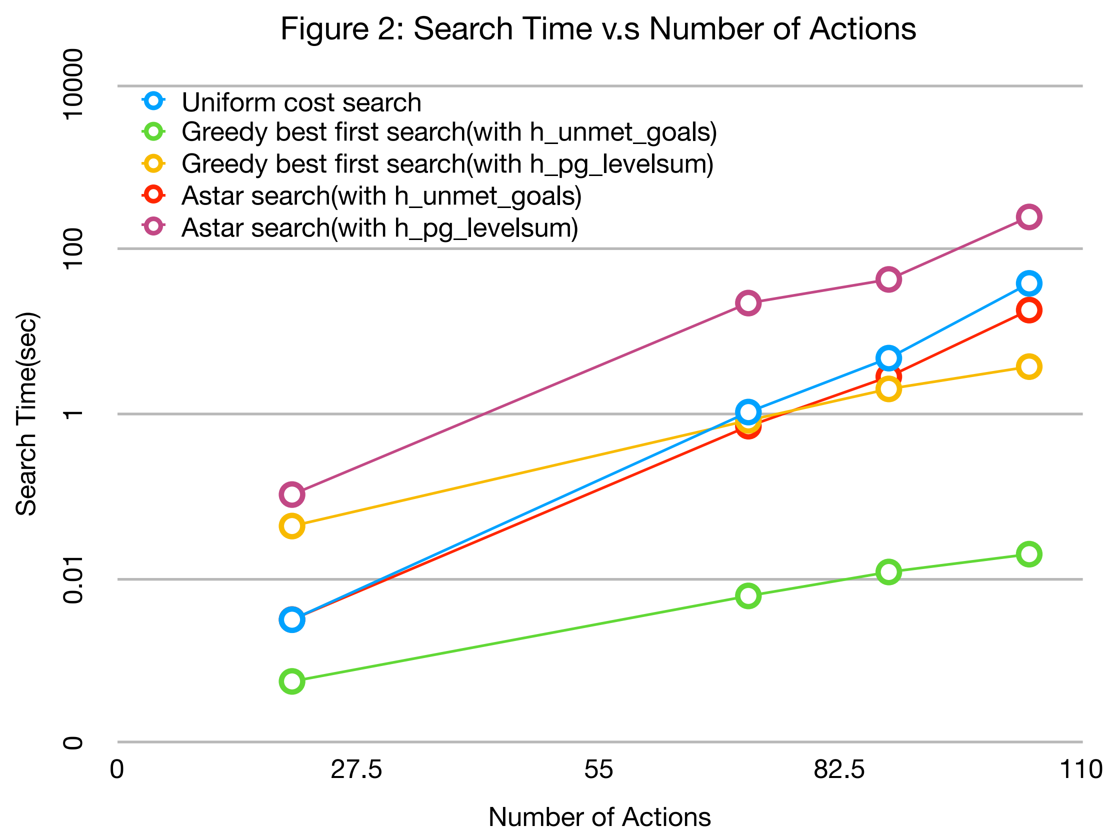
Search Time

Table 2: Search Time

Search time(sec)	P1(#action 20)	P2(#action 72)	P3(#action)	P4(#action 104)
Breadth first search	0.00199284398695454	0.6327979550114833	3.49896235804772	31.632155402039643
Depth first search	0.0011115919915027916	0.9164393020328134	0.3631585740367882	1255.4929340649978
Uniform cost search	0.0031590540311299264	1.0542622620123439	4.770671224978287	38.65481398900738
Greedy best first search with h_unmet_goals	0.000562014989554882	0.006148003973066807	0.0120090129785 2397	0.019773103995248675
Greedy best first search with h_pg_levelsum	0.04342343402095139	0.8359332629479468	2.015552362019662	3.7609174629906192
Greedy best first search with h_pg_maxlevel	0.029970114992465824	1.3208192640449852	2.1198573240544647	6.372067677031737
Greedy best first search with h_pg_setlevel	0.12786401499761268	3.043137119966559	16.919008856988512	77.78098958299961
Astar search with h_unmet_goals	0.0031719940016046166	0.7105693609919399	2.8379849590128288	18.296345016977284
Astar search with h_pg_levelsum	0.10481709300074726	22.267187930003274	43.158625721989665	247.9038546369993
Astar search with h_pg_maxlevel	0.10849602799862623	129.7763412820059	780.7069767760113	N/A
Astar search with h_pg_setlevel	0.3022687429911457	272.49078862101305	1486.5060756680323	N/A

As shown in table 2 and Figure 2, the search time increases exponentially as the number of actions increases. Meanwhile we could observe substantial differences among algorithms and heuristic functions.

Uninformed searches(Breath first search, Depth first search, Uniform cost search) tends to run fast when the scale of problem is small, but the search time will increase exponentially at a faster rate, compared to informed searches, when it deals with a higher number of actions. This makes sense as in a problem of large scale, the trimmed number of nodes expanded in an informed search will help to drive down the computational time significantly. Greedy best first searches generally produces search time that increases at a smaller rate compared to Astar search, because greedy searches don't try to find the optimal solution, have fewer nodes expanded, and therefore less time-costly, compared to Astar searches.



Among heuristic functions, “unmet-goals” is most time efficient although it expands the most number of nodes. This is because “unmet-goals” estimation is very efficient to compute. “level-sum” comes second as it balances the computational complexity of heuristic estimation and number of nodes expanded. “max-level” has a similar computational complexity of heuristic estimation with “level-sum”, but “max-level” has a much higher number of nodes expanded. “set-level”, while having the least number of nodes expanded, will cost too much in estimation computation.

The Length of Plan

Table 3: The Length of Plan

Length of Plan	P1(#action 20)	P2(#action 72)	P3(#action)	P4(#action 104)
Breadth first search	6	9	12	14
Depth first search	20	619	392	24132
Uniform cost search	6	9	12	14
Greedy best first search with h_unmet_goals	6	9	15	18
Greedy best first search with h_pg_levelsum	6	9	14	17
Greedy best first search with h_pg_maxlevel	6	9	13	17
Greedy best first search with h_pg_setlevel	6	9	17	23
Astar search with h_unmet_goals	6	9	12	14
Astar search with h_pg_levelsum	6	9	12	15
Astar search with h_pg_maxlevel	6	9	12	N/A
Astar search with h_pg_setlevel	6	9	12	N/A

As shown in table 3, Breath first search(BFS), Uniform cost search(UCS), and Astar search with admissible heuristic estimation(“unmet-goals”, “max-level”, “set-level”) give the optimal length of plan in all four problems. Depth first search(DFS) produces lengths of plan that could diverge a lot from the optimal ones. Greedy best first searches tends to find the optimal solution when the scope of problem is small, like in P1 and P2, but they fail to produce the minimal length of plan when problem get more complex, like in P3 and P4. But to be fair, In P3 and P4, greedy best first searches still provide sub-optimal solutions, which are much better than DFS. One out-lier is Astar search with “level-sum”. It tends to find the best solution at P1, P2, and P3, but when the scope of problem get larger, it only provide a sub-optimal solution as “level-sum” is not an admissible heuristic estimation.

One caveat behind BFS is that if the cost of each action is not equal, then BFS does not guarantee to find the optimal solution. UCS and Astar searches with admissible heuristic estimation always guarantee optimal solutions.

Q&A

Q: Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

Answer: Based on the previous analysis and charts, greedy best first search with "unmet-goals" would be the best solution for a real-time application in a very restricted domain. Because this algorithm is computationally least costly.

Q: Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)?

Answer: In a very large domains, being computationally fast and finding the optimal solution are equally important, because time and cost both matter. Base on this principle, Astar search with "unmet-goals" could be most appropriate for this application.

Q: Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Answer: Uniformed cost search(UCS) and Astar searches with admissible heuristic estimations("unmet-goals", "max-level", "set-level") will guarantee optimal plans. With computational time in concern, Astar search with "unmet-goals" stands out as a better solution compared to other candidates.