Udacity AI Nanodegree: Project 2

Introduction

The aim of this project is to demonstrate the differences in performance of several search algorithms in the context of a graph search using PDDL.

The performance of both informed and uniformed search algorithms is assessed as the complexity of the problem grows.

Problems

There are 4 problems all variations of an air cargo problem where there are a number of planes, airports and cargos. The goal is for specific cargos to be delivered to specific airports.

- Problem 1 has 2 planes, 2 cargos and 2 airports.
- Problem 2 has 3 planes, 3 cargos and 3 airports.
- Problem 3 has 2 planes, 4 cargos and 4 airports.
- Problem 4 has 2 planes, 5 cargos and 4 airports.

These problems increase in complexity as shown in the table below.

Problem	Number of fluents	Size of state space (2^number of fluents)	Total number of actions
Air cargo 1	12	4096	20
Air cargo 2	18	262144	72
Air cargo 3	16	65536	88
Air cargo 4	18	262144	104

Search algorithms

Three uniformed algorithms are used:

- Breath first search (BFS): Complete and optimal in this case
- Depth first graph search (DFGS): Not complete and not optimal
- Uniform cost search (UCS): Complete and optimal

Additionally, two informed algorithms are used:

- Greedy best first graph search (GBFGS): Not complete and not optimal
- A * search: Complete for finite search space and optimal for admissible heuristic function (might not be optimal with level sum heuristic)

These were given a choice of four heuristics to inform them:

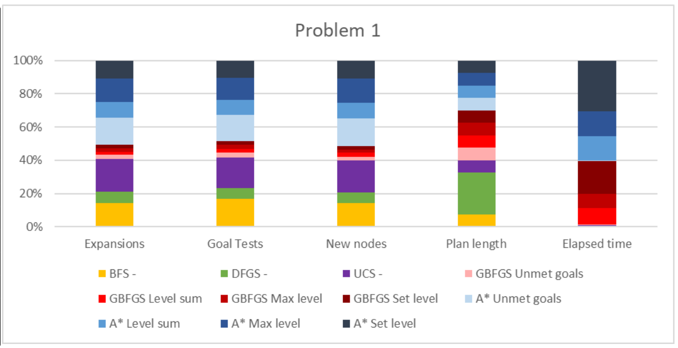
- Unmet goals: Number of goals unmet at each level.
- Level sum: Sum of level costs of all the goals. Assumes sub-goal independence so not always admissible.
- Max level: maximum of all level costs. Admissible.
- Set level: Level where all literals in conjunctive goal appear without any pair being mutex. Admissible.

Problem 1

This is the least complex problem.

The summary statistics are gathered from run_search.py. The raw results are displayed in a table as well as a 100% stack bar graph intended to show how each algorithm compares to the others in each performance measure. These are showed here.

Algorithm	Heuristic	Actions	Expansions	Goal Tests	New nodes	Plan length	Elapsed time
BFS	-	20	43	56	178	6	0.020682
DFGS	-	20	21	22	84	20	0.008601
UCS	-	20	60	62	240	6	0.033004
GBFGS	Unmet goals	20	7	9	29	6	0.005257
GBFGS	Level sum	20	6	8	28	6	0.531255
GBFGS	Max level	20	6	8	24	6	0.470613
GBFGS	Set level	20	6	8	28	6	1.049193
A*	Unmet goals	20	50	52	206	6	0.033009
A*	Level sum	20	28	30	122	6	0.787512
A*	Max level	20	43	45	180	6	0.804734
A*	Set level	20	33	35	138	6	1.643347



All the algorithms are optimal apart from DFGS and GBFGS. However, they all solve the problem in the optimal plan length of 6 except for DFGS which takes many more actions.

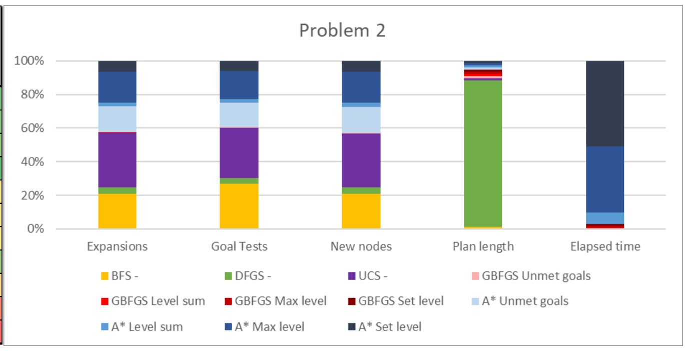
The GBFGS algorithms (reds in bar graph) all expand the least number of nodes. The A* searches (blues) have the most nodes expanded. The uninformed algorithms also expand a larger number of nodes (apart from DFGS).

The uninformed searches take the least amount of time to find a solution. The informed searches in general take longer. For the informed searches, the unmet goals heuristic is the fastest and the set level heuristic is the slowest. Level sum and max level are similar. The GBFGS algorithm was always faster than the A*.

Problem 2

The second most complex problem.

Algorithm	Heuristic	Actions	Expansions	Goal Tests	New nodes	Plan length	Elapsed time
BFS	-	72	3343	4609	30503	9	0.359787
DFGS	-	72	624	625	5602	619	0.58614
UCS	-	72	5154	5156	46618	9	0.683337
GBFGS	Unmet goals	72	17	19	170	9	0.038218
GBFGS	Level sum	72	9	11	86	9	2.188374
GBFGS	Max level	72	27	29	249	9	3.32135
GBFGS	Set level	72	9	11	84	9	4.070123
A*	Unmet goals	72	2467	2469	22522	9	0.866988
A*	Level sum	72	357	359	3426	9	28.15604
A*	Max level	72	2887	2889	26594	9	162.144
A*	Set level	72	1037	1039	9605	9	211.7263



Again, only DFGS doesn't solve the problem in the optimal plan length. In this more complex problem, it results in a much less optimal plan than in the simpler case.

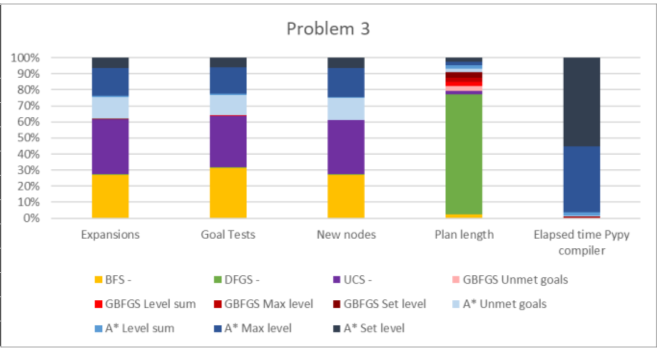
The same patterns for nodes expanded apply here. GBFGS with any heuristic expanded **much** fewer new nodes than A* searches or any of the uniformed searches. DFGS expands the least new nodes of the uninformed searches. The level sum and set level heuristics for both types of informed algorithms expanded fewer new nodes than the other heuristics.

The same general trends in elapsed time are seen as before. They are however much more exaggerated. The A* (bar graph dominated by blue) searches take **much** longer than any of the other algorism to find a solution (14-50x longer than the corresponding GBFGS algorithms and even worse for the fast running uninformed searches.

Problem 3

While the state space in this problem is smaller than problem 2, the increase in number of possible actions makes this problem more complex.

Algorithm	Heuristic	Actions	Expansions	Goal Tests	New nodes	Plan length	Elapsed time Pypy compiler
BFS	-	88	14663	18098	129625	12	0.923018
DFGS	-	88	408	409	3364	392	0.280513
UCS	-	88	18510	18512	161936	12	1.498306
GBFGS	Unmet goals	88	25	27	230	15	0.045317
GBFGS	Level sum	88	14	16	126	14	3.406132
GBFGS	Max level	88	21	23	195	13	3.925262
GBFGS	Set level	88	35	37	345	17	15.3396
A*	Unmet goals	88	7388	7390	65711	12	1.414474
A*	Level sum	88	369	371	3403	12	45.9704
A*	Max level	88	9580	9582	86312	12	799.726
A*	Set level	88	3423	3425	31596	12	1081.851



This is the first problem where GBFGS also shows its non-optimality and generates a plan length longer than the optimal 12 but quite close to optimal. DFGS on the other hand generates a far longer plan than the others again. This effect is not as exaggerated as in the previous problem, this presumably stems from the state space in this problem being smaller than that in the last problem.

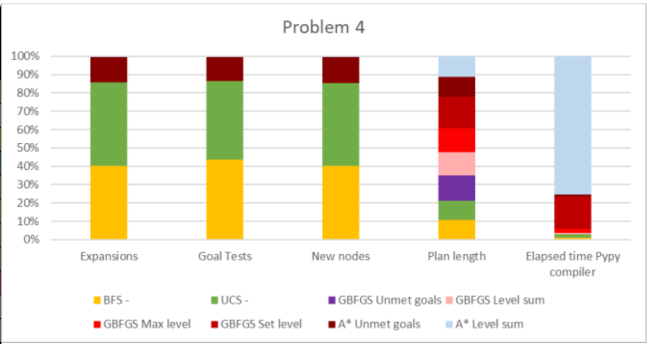
In terms of new nodes expanded we see the same story as in the problem 2. The GBFGS expands much fewer new nodes than any other algorithm with the A* search, BFS and UCS all expanding considerably more new nodes than the rest. The A* algorithm with unmet goals heuristic takes more than 200 times longer than the corresponding GBFGS algorithm with the same heuristic.

Time to solve the problem is even more skewed here than previously. The time taken to solve the problem in other algorithms is negligible in comparison to the time taken by the A* search (especially with max level or set level heuristics). The GBFGS with the set level heuristic is more than 72 times faster than A* with the same heuristic. The goals unmet heuristic with either informed search and uninformed searches, were much faster with UCS being the slowest of the uninformed searches.

Problem 4

The most complicate problem.

Algorithm	Heuristic	Actions	Expansions	Goal Tests	New nodes	Plan length	Elapsed time Pypy compiler
BFS	-	104	99736	114953	944130	14	4.936673
DFGS	-	104					
UCS	-	104	113339	113341	1066413	14	8.191043
GBFGS	Unmet goals	104	29	31	280	18	0.0545
GBFGS	Level sum	104	17	19	165	17	5.195576
GBFGS	Max level	104	56	58	580	17	10.69128
GBFGS	Set level	104	107	109	1164	23	83.38853
A*	Unmet goals	104	34330	34332	328509	14	5.045815
A*	Level sum	104	1208	1210	12210	15	360.4537
A*	Max level	104					
A*	Set level	104					



The DFGS for this problem generated a plan too long to run and the A* algorithm with max level or set level heuristics took far, far too long to run and so they were terminated. This is to be expected, we have seen that as complexity increases, the A* algorithm is especially adversely affected in run-time.

As, expected the plan length found by BFS, UCS and the A* search with unmet goals is the optimal plan length of 14. For the first time we see that the A* search with level sum heuristic does not find an optimal solution. This will be because the sub-goals in this problems goal state are not completely independent.

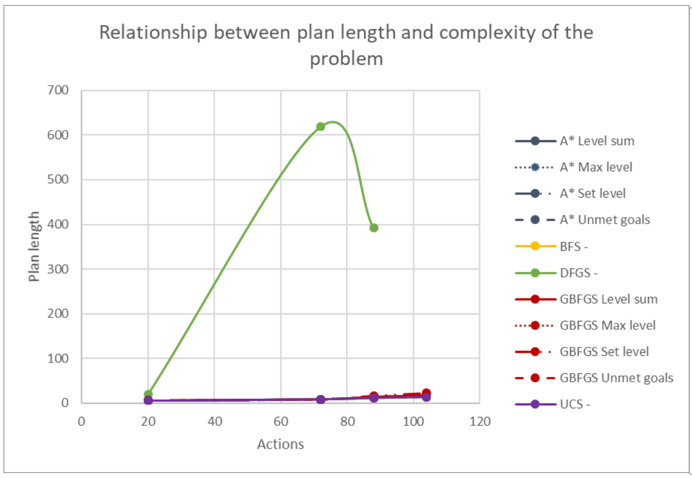
The number of new nodes expanded are again much less for the GBFGS and since this problem is even more complex, the uniformed algorithms and A* algorithms expand even more node in comparison than before. The A* algorithm with unmet goals heuristic takes more than 1000 times longer than the corresponding GBFGS algorithm with the same heuristic.

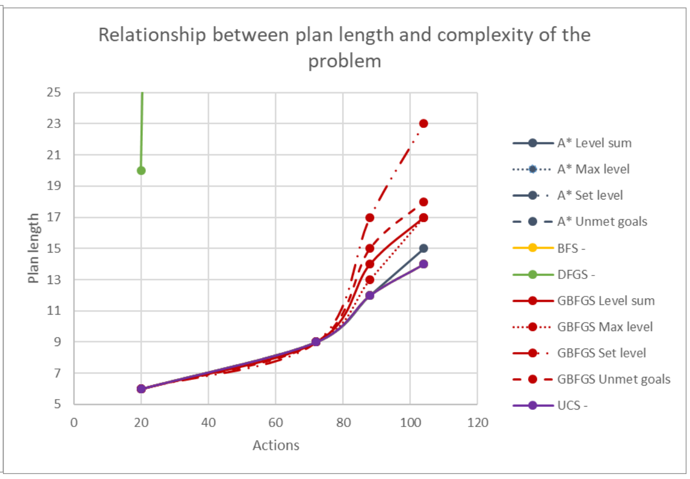
The A* algorithms which took too long to run obviously took the longest. The A* with level sum heuristic, however, still takes many times longer than any other algorithm to run. The GBFGS with set level heuristic took the second longest to run and the others were all negligible in comparison.

Complexity vs Plan length

Using the number of actions as a measure of complexity, we can now further investigate the trends for the algorithm's performances across the levels of increase complexity.

The graph below on the left shows how plan length varied for each algorithm as complexity increased. The graph on the right is the same graph but zoomed in to better see the trends for algorithms with more optimal solutions.



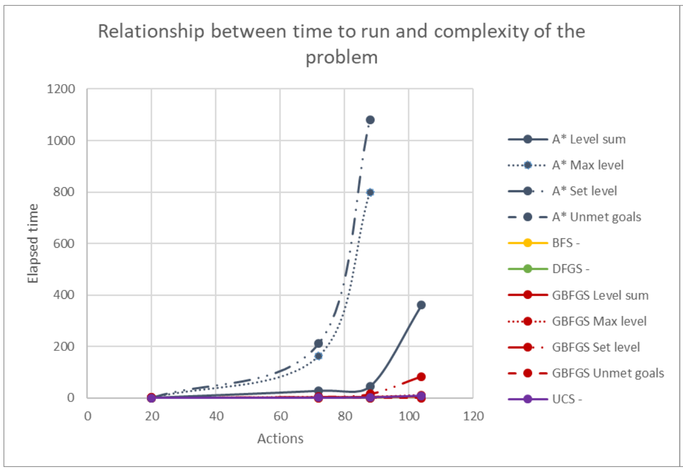


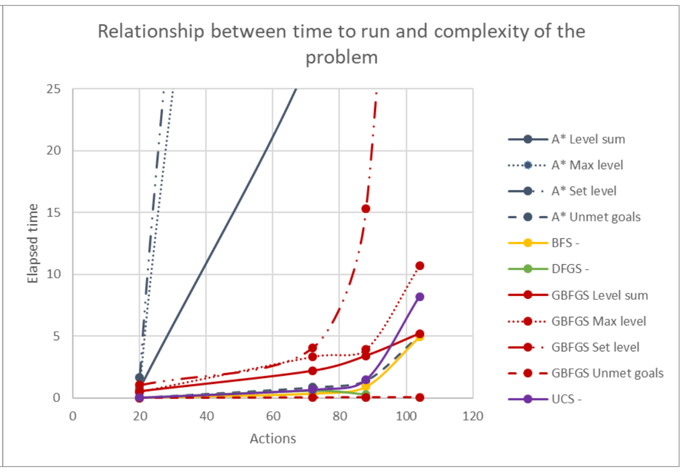
DFGS dominates the plan length over all complexities due to it being non-optimal. As the problem complexity increases, so does the plan length for all algorithms apart from the DFGS. This is due to it being non-optimal so that it takes many more actions than necessary and due to there being more states to search down into in the second problem than in the third. For optimal algorithms and those that are closer to optimality, plan length increases as complexity does.

While all the algorithms follow the same trend (apart from DFGS), the greedy searches which are non-optimal, always have longer plan lengths. So does the A* algorithm using level sum. The optimal solutions in the second graph always have the same plan lengths so they are all obscured under the purple UCS line.

Complexity vs Elapsed Time

The graph below on the left shows how elapsed time varied for each algorithm as complexity increased. The graph on the right is the same graph but zoomed in to better see the trends for algorithms which found solutions quickly.



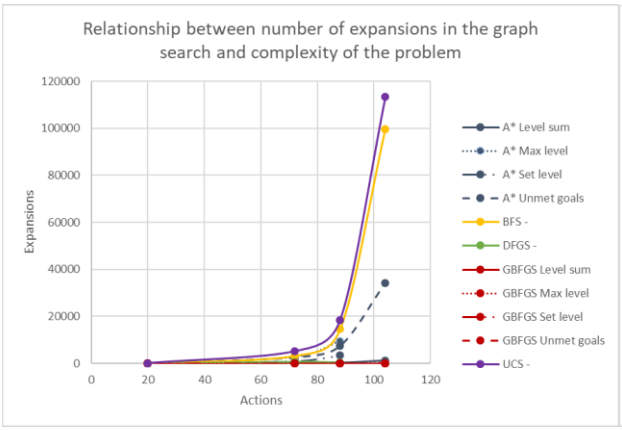


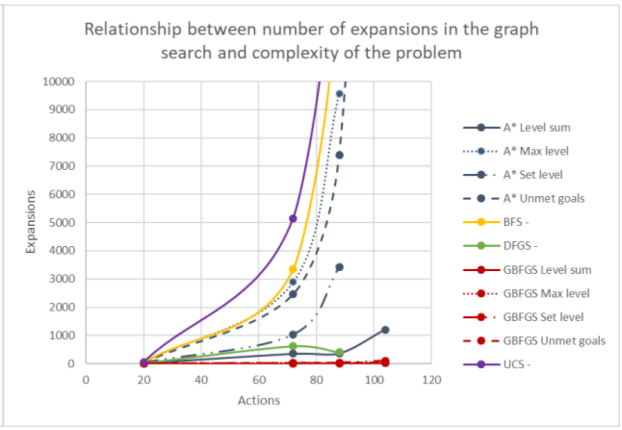
In general, all times increased as complexity of the problem increased. All variations of the A* algorithm (except unmet goals) and the GBFGS algorithm with set level heuristic took **much** longer to find a solution than any of the others over all complexity levels. These were followed by the remaining GBFGS algorithms, UCS, A* with unmet goals heuristic, BFS, DFGS and the fastest being the GBFGS with unmet goals heuristic.

For informed algorithms, the unmet goals heuristic consistently results in faster solutions and the max level and set level heuristics take far longer.

Complexity vs New Nodes

The graph below on the left shows how number of new nodes visted varied for each algorithm as complexity increased. The graph on the right is the same graph but zoomed in to better see the trends for algorithms which visited fewer new nodes.





As expected, number of new nodes visited increased as complexity of problem increased. Here, the informed algorithms tended to visit fewer new nodes than the uninformed ones since they have extra information with which to reject some nodes which wont lead to a solution.

The UCS and BFS always explored many more new nodes than the others. They were followed by the A* algorithms and DFGS. All versions of the greedy algorithm, explored far, far fewer new nodes than any of the other algorithms. This is due to its greedy nature considering only cost to the goal as compared with A* searches. The trade-off being that the same greedy nature is what causes it to be incomplete and may not find a solution at all for some problems.

Conclusions

In a restricted domain with few actions, in order to operate in real time, I would prefer to use uninformed algorithms. Since they don't have to do any of the extra computation that the informed ones do, in a domain where the state space is small enough to search most of it, uninformed algorithms perform very well. I would even include the DFGS in this category since, although it is far from optimal, in a restricted domain, it doesn't result in too many extra actions and is very fast.

Additionally, I would also investigate informed algorithms but only those with very simple heuristics which don't take a lot of time to compute for the same reasons as above. In fact, the fastest algorithm in problems 1 and 2 with fewer actions than the others, was the GBFGS using the unmet goals heuristic.

In very large domains on the other hand, when the state space (2^ total positive and negative fluents) is small, then uninformed searches still perform well (apart from DFGS) and are faster than then informed ones as in problem 3. In this situation the informed algorithms with the simple unmet goals heuristic were also quite fast.

In very large domains which also have large state spaces like problem 4, some of the informed algorithms catch up to the uniformed in terms of speed. The GBFGS using the unmet goals heuristic is the fastest here too and while using more complex heuristics does not affect performance as badly as in smaller domains, there is still a trade-off between strength of the heuristic and the amount of time it takes to compute. Additionally, the A* algorithms performed exceptionally poorly in terms of speed here. This is due to them evaluating backwards and forwards costs.

It would seem that in large domains the compromise of having an informed agent but one that does not evaluate too much information is best in terms of performance (GBFGS algorithms). They also consistently explore fewer nodes than the other algorithms. However, this does come with the disclaimer that they worked well in these problems. Since they are non-consistent and non-optimal, they might not find a solution at all in other complex problems.

In larger domains optimality comes into play too. While the non-optimal DFGS was an option in smaller problems due to its speed, in large problems it can try to build a plan so large and cumbersome that it might not be worth using. This also begins to significantly increase its run times.

When it is important to only find optimal plans, out of the algorithms evaluated here, only the UCS and BFS are always optimal and produce the lowest cost plan. The A* search is also optimal but only when the heuristic provided is admissible. This was not the case with the level sum heuristic used in problem 4. In general, the A* algorithm struggles with long run times and so in this case, I'd prefer to give priority to trying the uninformed searches.

Acknowledgments

All information used to justify these views were obtained from the Udacity AI Nanodegree lecture videos and Artificial Intelligence A Modern Approach by Stuart J. Russell and Peter Norvig.

Appendix

PDDL: Planning domain definition language. Allows all actions to be expressed in one action schema

Complete: Will always find a solution

Optimal: Will find the solution with the least possible cost.