Planning Report

Filipe Reis – Udacity AIND Student

Optimal Sequences

In this section we provide an optimal sequence of action for each proposed problem.

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

Heuristic Analysis

In this analysis, we chose to focus on time spent to complete the process, number of node expansion and number of resulting actions. The first parameter was chosen as ideally the best algorithms should run faster, while the second provides information regarding the operation of the algorithm. Finally, number of resulting actions is closely related to the quality of the solution, as each action has a cost, which should be kept to a minimum. The analysis first considers individual results for each problem and then summarizes global algorithm performance.

Problem 1

This is a rather simple problem, which allowed for all the available heuristics to converge in an acceptable timeframe, with the fastest being Greedy Best First Graph Search h1, completing in only 0.01 seconds and the slowest, Recursive Best Fit Search h1 which took 3.42 seconds to complete. For this problem, the ideal action set had 6 actions and all algorithms except for Depth First Graph Search and Depth Limited Search resulted in a set with this size. The results are presented on Figure 1.

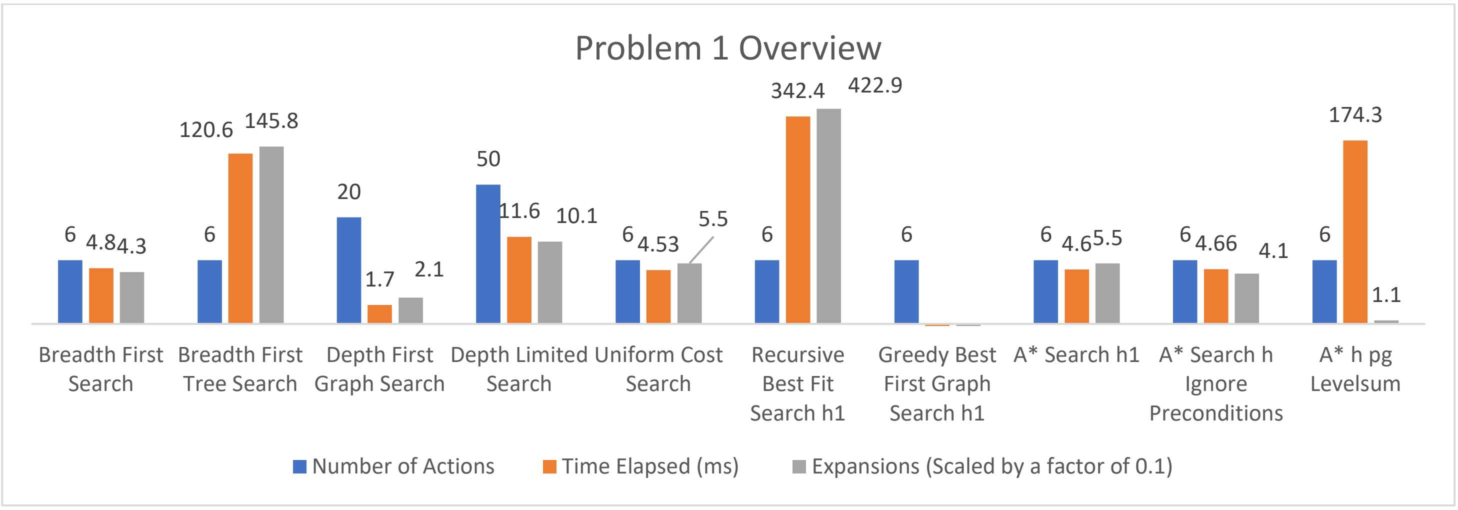

Figure 1 - Problem 1 Overview

Problem 2

Problem 2 was more difficult to solve than the first one. Breadth First Tree Search, Depth Limited Search and Recursive Best Fit Search h1 were unable to achieve results in under 10 minutes, what excluded them from the analysis. The best solution for this problem had a set of 9 actions and was achieved by all considered algorithms, except for Depth First Graph Search and Greedy Best First Graph Search h1. The fastest algorithm to solve this problem was Depth First Graph Search, achieving a solution in only 0.4 seconds. Figure 2 presents the results.

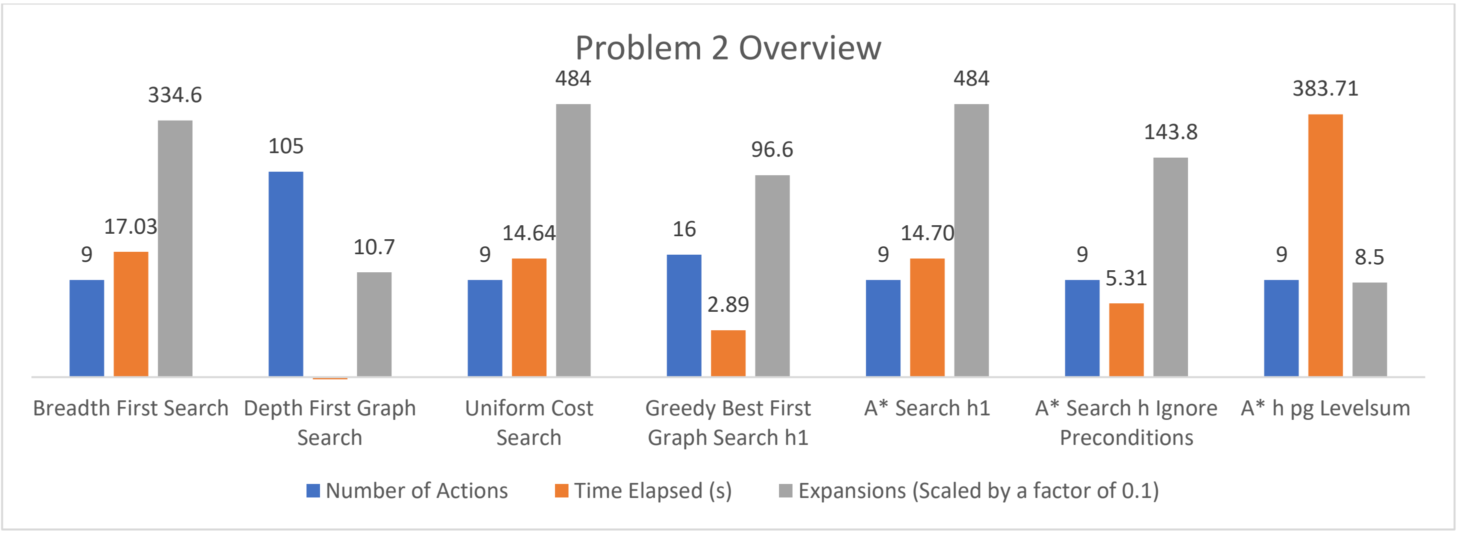

Figure 2 - Problem 2 Overview

Problem 3

The last problem was the toughest one. Only 6 algorithms were able to provide a result in under 10 minutes. The ideal set for this problem had 12 actions and, from those which completed the task, only Depth First Graph Search and Greedy Best First Graph Search h1 could find a solution with this size. The fastest algorithm was Depth First Graph Search, completing in only 1.41 seconds a solution with 288 actions. The fastest algorithm to provide the optimal solution was A* Search h Ignore Preconditions, taking 21.37 seconds to complete.

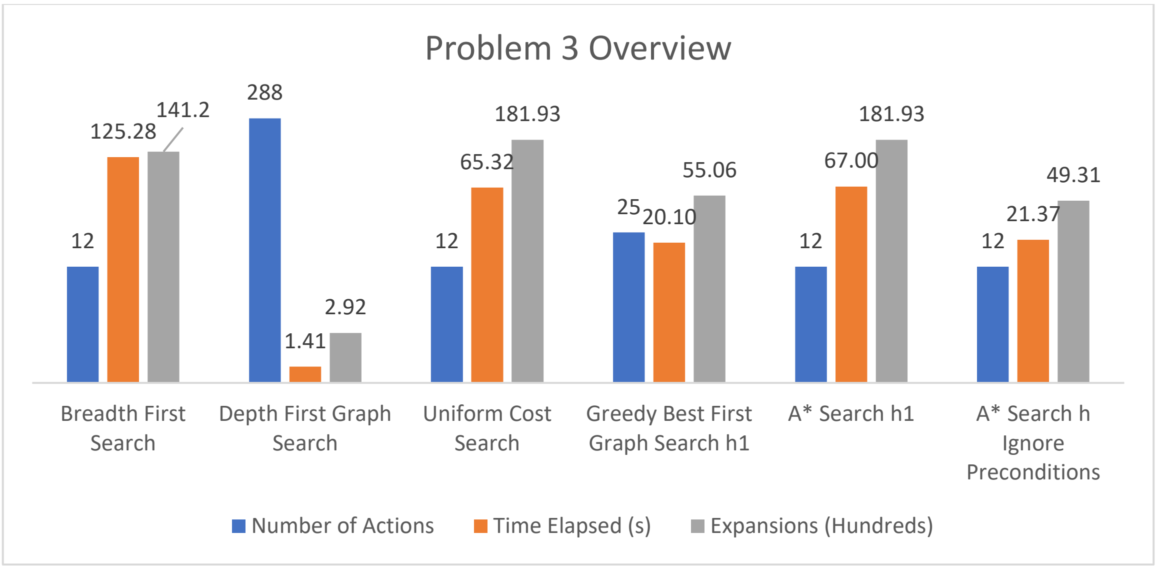

Figure 3 - Problem 3 overview

Overview

To determine the best algorithm, it is necessary to evaluate their performance across all problems. O the number of actions point of view, A* Search h1, A* Search h ignore, Uniform Cost Search and Breadth First Search obtained the best results, as shown in Figure 4.

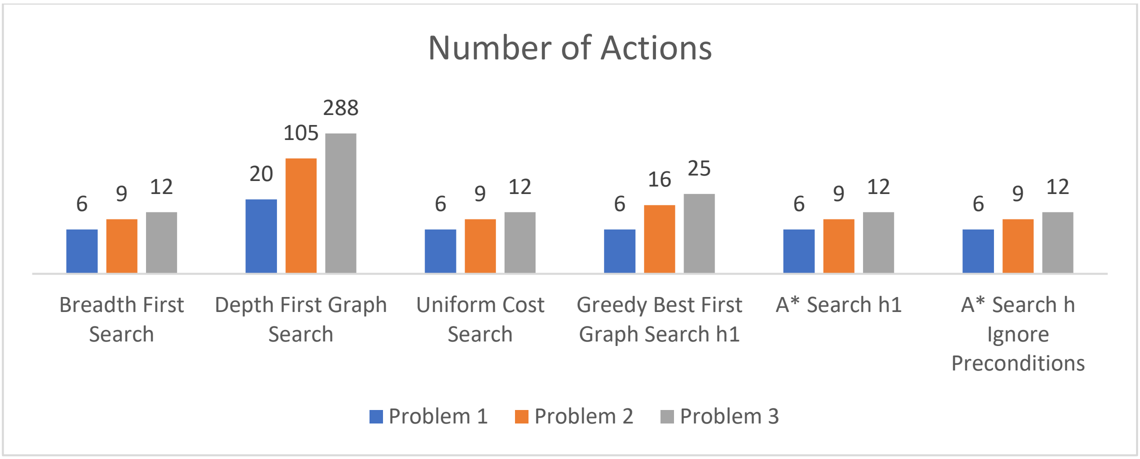

Figure 4 - Number of actions analysis

Additionally, the fastest algorithm across all problems was Depth First Graph Search, while the best performer which achieved an optimal plan was A* Search h Ignore Preconditions. Figure 5 shows the performance for each problem.

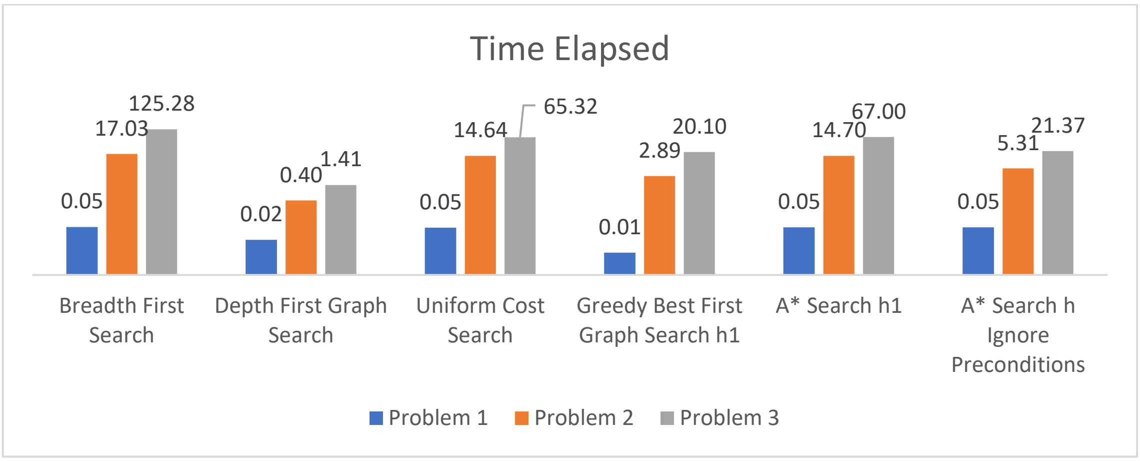

Figure 5 - Time elapsed across all problems and algorithms

The number of expansions for each algorithm have a close relation to the time elapsed, with the smallest group being achieved on Depth First Graph Search and the smallest between the optimal achiever is A* Search h Ignore Preconditions.

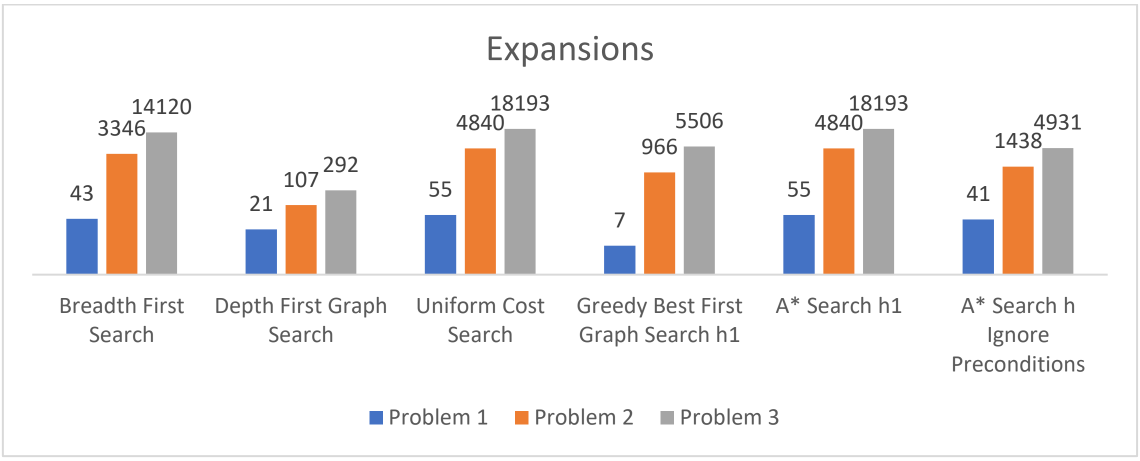

Figure 6 - Overall results for number of Expansion Nodes

Therefore, the best algorithm overall was A* Search ignoring preconditions, which is related to the fact that this algorithm focuses on finding the best solution, hence getting the optimal set, and ignoring the preconditions makes it faster and with a reduced number of expansions, at least for simple problems.