Joe Sheppard

CSC 380

Prof. Jonathan Gemmel

8-Puzzle Solver 1-25-2015

UNDERGRAD WITH GRADUATE WORK

https://www.youtube.com/watch?v=kuD9R

fsxONs&feature=youtu.be

Performance Results

EASY

ALG	Length of Solution	Time (ms) (avg of 5 runs)	SPACE (max nodes on queue)
DFS	13	11.6	12
BFS	5	14.2	36
GBF	5	13.6	8
A* (h1)	5	13.8	8
A* (h2)	5	14.4	7

MEDIUM

ALG	Length of Solution	Time (ms) (avg of 5 runs)	SPACE (max nodes on queue)
DFS	729	903.6	59,458
BFS	9	22.4	204
GBF	9	21	46
A* (h1)	9	13.4	27
A* (h2)	9	11.8	17

HARD

ALG	Length of Solution	Time (ms) (avg of 5 runs)	SPACE (max nodes on queue)
DFS	21,118	1174	18,764
BFS	30	918	24,971
GBF	64	131.8	962
A*(h1)	30	110,000	25,953
A*(h2)	30	2,548	2,254

The results I obtained were interesting, eye-opening and unexpected to say the least. Some Algorithms failed miserably in certain situations, while being fantastic in others. No algorithm seemed outperform the others in all criteria and situations, and thus I don't feel I can reach a conclusion that any one of the Algorithms are "better" than the others. I can, however, comment on some of the strengths and weaknesses of each:

When looking at the easy problem, the differences seem to be negligible. Time wise, they all take nearly the same amount of time. In addition, all but DFS are able to come up with the same (optimal) solution. When considering space, BFS uses the most space, though it is not an extraordinary amount due to the ease of the problem. The informed (heuristic) algorithms are able to find the solution in about the same amount of time while using slightly less space.

Results from the medium problem seem to be similar, with a few very notable exceptions. The first, and most obvious solution is in the case of DFS. It requires over 400x the time and space of any of the other solutions, and comes up with an extremely long and far from optimal solution, unlike all the other methods. This is due to the nature of DFS and simply getting "unlucky". Perhaps if a different path was chosen to begin on, the results would be better. Another exception to the similarity to the easy problem can be seen when considering the two A* Algorithms. They take about half the time and space of the other solutions.

We get into some sticky territory when looking at the hard problem. I almost believe that this starting positon was designed in a way to make many of the Algorithms struggle with it; I manually entered a few other random starting states and couldn't reproduce these results. Interestingly, the greedy best-first search performed by far the best in this scenario in both time and space criteria. The negative is that it was not able to find the optimal solution. As far as optimal solutions, the best performing Algorithm as far as time is concerned is BFS, and the Algorithm that required the least amount of space is A*, using the second heuristic.

One thing that I think is very notable is the utter failure of A* with the first heuristic on the hard problem. Not only did it require even more space than BFS, but it took almost 2 minutes to find an answer!! I believe the reason for this stems from the stated major weakness of the A* algorithm, that it is possible for it to expand many, many states in search of the optimal solution, often causing it to run out of memory. In this case, it did not run out of memory, but as the queue grew, it became exponentially more and more costly to processes it, especially the repeated sorting of the large queue. The reason the Second A* performed so much better than the first, is that the first is heuristic, simply the number of displaced tiles, is quite bad, and led to an inefficient Algorithm in this hard scenario.

In many situations it seems the greedy best-first produces the best results if you are looking for an answer quickly and efficiently. If you are looking for an optimal solution. A* is a very good choice **only if it has a good heuristic**. If a good heuristic cannot be provided, it seems BFS may be a better option.

I don't believe my 8-puzzle solver is intelligent, by most standards. When given the problem of solving this particular puzzle, and this particular puzzle only, it can be safely said that it performs better than a human. That may make it appear intelligent, but a calculator can solve a certain problem better than a human as well. Efficiently finding a solution to a given problem, I would argue, does not make an agent intelligent.

That said, my opinion may be greatly skewed due to the fact that I programmed the Agent. I remember on the first day of class we mentioned that we often think of a problem as not requiring intelligence after we already solve it. That's an interesting statement, but I think it has some validity. We might consider an agent intelligent when we see it doing something we didn't think a computer could do. I think someone with no experience in programming or artificial intelligence, or I suppose more specifically, how the algorithms work, may be more "impressed" with the agent and consider it more intelligent than I do.

But does that mean nothing can be intelligent? Does it mean that when we are able to program a computer to solve a previously unsolvable problem, we won't see it as the machine's intelligence, but instead ours because we figured out how to make the machine do it? I think so, yes. I think that intelligence can be achieved only in the machine solving problems we did not think it could solve. When a computer is able to solve problems completely separate from what was programmed into it, learns and figures out how to solve the problem itself, I think maybe then it could be considered intelligent. That said, I have only been working with artificial intelligence for a few weeks, and as I learn new techniques, I feel it is very likely my opinion may change on what actually constitutes intelligence.