Fundamental of AI Coursework 2

Improved Method for 15-puzzle-problem

Introduction

In this coursework I choose to try to improve the searching method used in coursework 1, the aim is to get the optimal solution with the least searching nodes. The general idea is to try some other Heuristic searching methods and improve it. So what I chose first is A-star method, then from A-start method, I applied a bidirectional A-star. After that, I got the idea how to use a learning method to do the searching, however I couldn’t realize it.

Heuristic searching

When it comes to improve the previous method, there are two ways appears in mind.

First one, improve the evaluation function. Greedy and A-star are using Manhattan distance in their evaluation function, but Manhattan distance is not accurate in this situation, the steps the face block use to move all characters is not the same as the distance between the goal and initial state. So if I want to prove the performance, I may need to calculate the real distance to goal state.

Second ideal is to change the way to use the evaluation function. The biggest problem of heuristic searching is the solution may not be optimal, so if I can apply the evaluation twice or searching several solutions then compare them and get the optimal one.

According to these two ideas I start my improvement.

A-star search

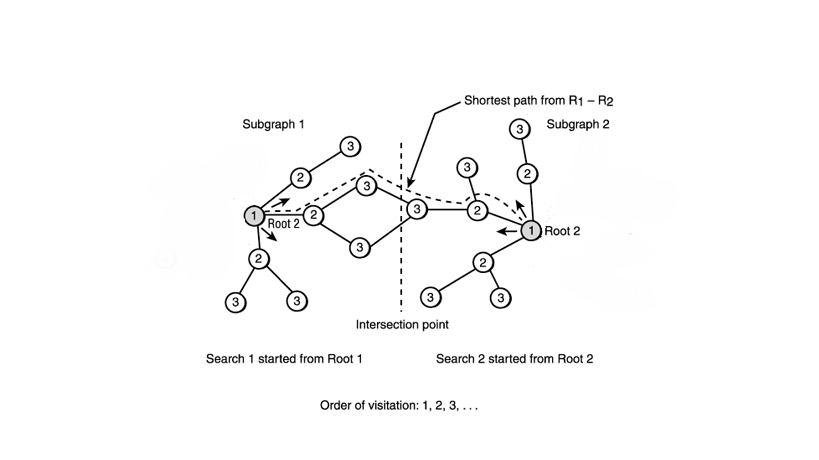
A-star searching generally is a improve method based on greedy search. Greedy search only calculate the distance between current state and goal state, A-star will add the cost from the initial state to current state, the reason to do so, is to make the solution more optimal. But this will cost more time to search states, I will show the result soon after.

Bidirectional A-star search

Bidirectional A-star is to start the search from both the initial state and the goal state (like Figure-1 shows), the aim is to make the two search tree meet in same state, and then we can get an entire route from the initial state to the goal state.

The general flow is, I expand a node, then chose the shortest node towards the current goal, then this node will became the goal of the bidirectional search and so on, when compare the distance, not only compare the current depth but all unexpanded states, and the distance should be recomputed towards the current goal. About re-calculate the distance, I found sometimes, it much possible to reach the optimal solution without re-calculate, this maybe because the wrong pick avoid the local maximum solution, but I can’t prove it, so it’s just my opinion.

The key point in this method is how to set the goals for both direction searches. When picking the state from depth 1, the Manhattan distance is compute between current states to the last goal state. Then the goal for the bidirectional search is the state chosen from depth 1, which just been chose to be expand, and the next goal for state in depth 2 is the new goal just been expand. And finally they will meet some where in around the middle.



Bidirectional A star with local search

This method is designed to find the optimal solution. Local search is an improvement when one solution has been found, it start the search again from another state, this state can be the neighbor state to the current solution or just pick a random state, trying to find another solution, and compare these two solutions, if the new solution is better, then start the search from this state, until it can’t find any better solution, that means, I have got the best one.

So to my situation, I need to make the search keep going, and search for another several times, I believe this could be a solution to find an optimal solution, however it means I need to search much more nodes than current state, which is not I prefer, so finally this method been abandon.

The last method I come to thought is using the learning method to search the puzzle. The principle is to using the Neural Networks to learn the true distance between the current states to the goal state, and then the face block can pick the shortest path to the goal.

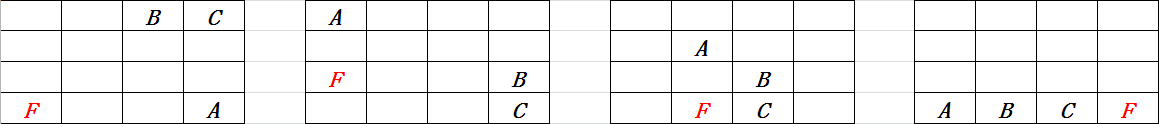
The first problem is how to treat the neural network. This requires lots of data, I need to know what the true distance is, how much steps it needs to take from this state to the goal. One way is to using the BFS to do once, and then I can know the true steps. So a 4\*4 board have 43680 potential combinations of states. Using BFS to compute the true steps from each state to goal state, this will became the training data of the neural network, then apply the trained neural network to the evaluation function, and done, the function will lead to the shortest path.

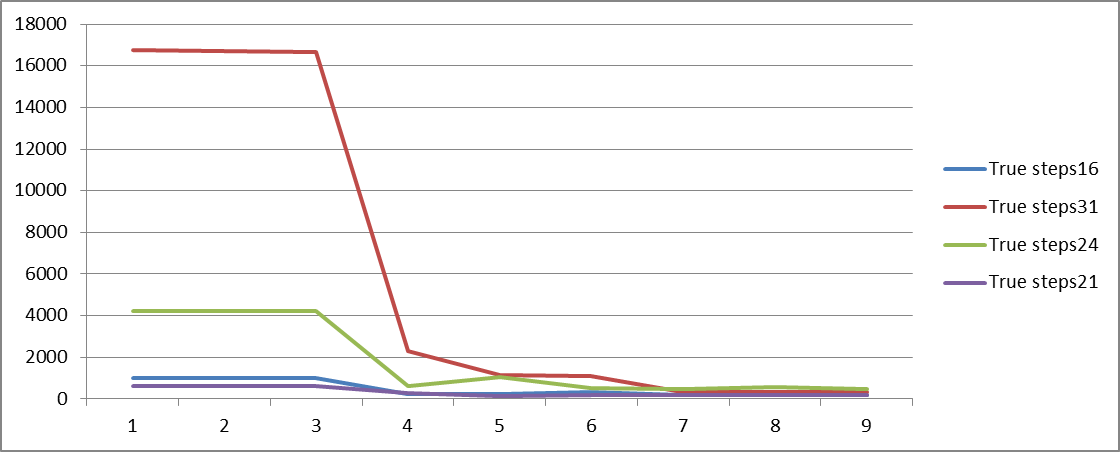
Result & Analysis

In this section, I will show the result of A-star and Bidirectional A-star in solving the 15-puzzle-problem and analysis the performance of these two searches. Greedy search will set as the contrast.

The performance is represented by the steps, how much steps it need to take towards the goal, and visited states, how much states it need to search. The best situation is to search the minimal states and get the shortest path.

I prepared 4 initial states this time (shown in figure-3), including the example state. Given the Manhattan distance can’t measure the difficulty well, so this time the difficulty is measured by how much steps it need to take from the start to the end, the steps is measured by A-star (running several times with the same initial and get the shortest path, I want to BFS, but my BFS take incredible long time when dealing long steps problem).here I don’t compare the time, the time also limited by the code structure, our aim Is just compare the searching methods.





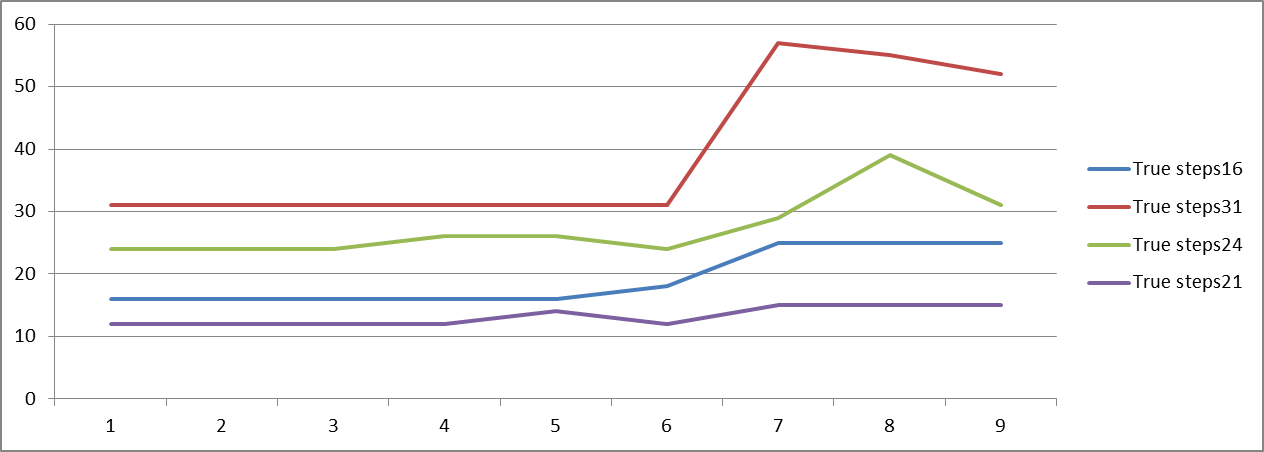


Figure-3 shows the number of visited nodes each method from different initial state. Figure-4 shows the steps each method gives.

1-3 represents the three run of A-star, 4-6 represents the three run of Bidirectional A-star and the last three is Greedy method.

From Figure-3 we can see, A-star tends to search more states than the other two, especially in long steps search, like 31 steps. However A-star can provide a stable and optimal solution, which can be observed from 1-3 in Figure-5.

Bidirectional A-star searches much less states than A-star, however it not always gives the optimal solution, from Figure-4 4-6 we can see the steps Bidirectional A-star provides is unstable, but the differences in steps is not big compared with A-star, the value is around 1-2 steps.

Greedy method always searches least nodes, but provides more steps than the other two, which is unacceptable.

Conclusion

Bidirectional A-star searches less than A-star, and provides a similar solution, compared with A-star, so I believe Bidirectional A-star is actually better than A-star. But we still can’t ignore the disadvantage of it; it can’t always provide an optimal solution. However if I try several times like I do in the test, I can still find an optimal solution. Last about the Neural network, I had a consider about that, but since I don’t have so much time left, so I can’t show the result about that.

**http://www.yaldex.com/games-programming/0672323699\_ch12lev1sec7.html**