Deaglan Diderich 101001565

COMP 4106 FINAL PROJECT

Introduction:

The topic that I chose to address in my project is decision making by AI in a multiple agent environment. I did this by building off our original assignment 1, where an agent must find the optimal path from a start square to a goal square in a grid of squares represented by a CSV file. Each entry in the CSV represents a node that can potentially be traversed; the value of the node is either A, B, C, X, or a number. A, B, and C are all start/goal nodes; X nodes are untraversable, and nodes with numbers in them are traversable with a path cost equivalent to their numbers. Each agent A, B, and C must find a path from its start node (A, B, and C, respectively) to a different start node. The agents cannot explore any squares that have been explored by other agents, however. Therefore, they are in direct competition with each other as the first agent to explore a cheaply traversed square gets to claim it before any other, forcing the other agents to take potentially more costly paths. The agents move one at a time, from A to B to C, taking turns and getting a chance to move once per round. My objective in this project is to see how the A* search algorithm performs in direct competition with other algorithms on the same field.

Motivation:

My motivation for studying this topic was to model how different agents would compete against each other for the best pathways to each other given a complex network of nodes with varying path costs, or walls. This modelling could be applied to games of capture the flag, or other strategic scenarios where multiple agents compete to get to each others' bases as quickly and efficiently as possible. It could even be applied to military skirmishes where each side aims to reach the others' bases as quickly and efficiently as possible.

Description of Methods of Artificial Intelligence Used:

The methods of A.I. I used were search algorithms, namely the A* search, as well as the greedy heuristic search, and Dijkstra's algorithm. Agent A uses A* search, meaning it considers both the heuristic which determines distance from a goal node (B or C) along with path cost, when searching for the optimal paths to a goal. Agent B uses greedy heuristic search, meaning it just looks for whichever node in the current path has the smallest heuristic value, because that node will be closest to a goal state. Finally, C uses Dijkstra's algorithm, which is like A* but only considers total path length when prioritizing, making it somewhat less optimal. I decided to use different search algorithms for different agents in order to compare their efficiency, and show that there are different situations in which each strategy excels, and situations where they falter.

Results:

Originally when I was programming the project, I gave every agent A* search. This ended up in a negative outcome, as their search paths were too similar, and they constantly ended in stalemates. Upon reflection, I realized that giving them different search algorithms would produce better results, and allow me to compare different algorithms' efficiency in different environments. Along the same line of thought, I created two extra variant cases for each of my four test cases. In the first variant, the position of A is replaced by C, B replaced by A, and C replaced by B. Similarly, in variant 2, A is replaced by B, B by C, and C by A. This allowed me to see how changing the algorithms but not the goal positions affected the outcomes of the searches.

What I found was that depending on the situation, A* search can perform worse than greedy heuristic or Dijkstra's algorithm. When goals are close to each other and the direct pathway between B and another goal is short but costly, depending on how many other viable options there are, A* can occasionally spend too much time searching other options before B reaches a goal. This results in B finding the shortest path first, because it charges through high-cost nodes without a second thought, claiming them for its own and reaching the goal quickly. Meanwhile, agent A will not look at high-cost nodes until later on in its search, resulting in it missing out on the opportunity to find a shorter pathway once B has claimed it, and forcing it to go the long way around, even though at first the longer path seemed more viable because it was full of low path cost nodes.

Interestingly, C's search algorithm operates similarly to A's except the lack of heuristic causes it to focus more on exploring nodes close to it than finding a goal. This results in it protecting the area around itself by exploring all possible options in its vicinity before branching out. This is much more costly in terms of time and nodes explored, but resulted in it almost never being reached by other agents. It would eventually find a decently short path to a goal, but only after exploring many nodes.

In order to more effectively evaluate the efficacy of each algorithm, I assigned a score to each algorithm, which was equal to its total path cost plus 1/5 the number of nodes it explored. I weighted explored nodes much less than path cost because I felt like the speed that a result is found is much less important than the resulting path having a short length. That said, depending on how important speed is compared to short paths (resource conservation), one could change the coefficient to closer to one.

Comparing scores with this new measurement, I found that often A would outperform both B and C, with B in second because of its generally very small list of explored nodes, and C in last due to the usually large volume of explored nodes. That said, C occasionally outperformed B due to the terrain; the more costly B's path was, the more likely it was that C would outperform it by taking its time.

Discussion of the Implications of the Work:

The results I achieved show that A* algorithm is not always the best algorithm to use in any given situation. When different agents are competing against each other for ground in a field, there are times when greedy heuristic search or Dijkstra's algorithm can find the most efficient solution faster than A* can. While eventually A* would have found the solutions that they did since A* always finds the most efficient solutions, when time is of the essence, it is possible that the time A* spends searching other routes is time wasted. That said, A* search's heuristic could be adapted to counter other algorithms like greedy heuristic search, if it was known that it would be facing off against greedy heuristic search.

Directions for Future Work:

As I was programming this, I thought about how machine learning algorithms could be implemented to adjust each algorithm's search methods. Each test case could be analyzed and used to create a policy and values associated with different nodes on the CSV, which would lead the algorithms to change their strategies as they were tested more and more.

These pathfinding algorithms could also be tested in more complex environments, for example, when programming AI in games or simulations. They would probably need to be able to balance resources, if they were in a strategy game like Civilization, leading them to be implemented in unique and complex ways.

User Manual:

To use my program, simply open the folder that it is in, and run it. The accompanying folders will already have files created corresponding to the input, output, and results of agent A, B, and C's searches in separate folders marked Test1, Test1_Alt1, Test1_Alt2, etc. As previously mentioned, the alternative versions are just the same as the original but with the agents in each others' places. Simply edit the CSV file marked input in each of the Test folders in order to change the node configuration of a given test. There should only be one A, B, and C in each CSV. The files marked a_data, b_data, and c_data have the results of each of the agent's searches: The optimal path, list of explored nodes, optimal path cost, and final score which takes number of explored nodes into account. The output file is a representation of which nodes each agent explored, with the lowercase a, b, and c's representing the nodes explored by A, B, and C respectively.

Contributions:

This whole project was done alone by me, Deaglan Diderich.