Pacman Capture the Flag

Robin Critten

April 29, 2023

1 Introduction

Both the Pacman and Ghost agents use an implementation of the Dijkstra's algorithm, a form of greedy algorithm [1]. This allows both agent types to find the shortest path through the maze to their desired location. The algorithm itself was implemented by modeling the patches as a network, each patch a node on the network and the edges being the connection between adjacent nodes. Since the distance between adjacent nodes can be traversed in a single tick each edge in the network was assumed to have a length of one.

From the context of the Pacman agent the final node, T, is chosen based on a heuristic function that determines the 'state' of the Pacman. If the outcome of the heuristic is positive, then the Pacman will input the coordinates of the nearest pellet into the Dijkstra function. Else, if the outcome is negative the Pacman will input the location of it's home position into the Dijkstra function. Positive values are added to the heuristic based on the proximity of pellets, and negative values are given based on the proximity of enemy ghost agent, multiplied by the number of pellets the Pacman is currently carrying.

Ghost in their base state will patrol the currently non-captured pellets in a Hamiltonian cycle, they use the first half of the Dijkstra's algorithm to construct a sub-network of the maze using the patches containing pellets as the vertices of sub-network. This is possible due to the first half of Dijkstra's Algorithm being able to find the distance between two specified points, S and T. A separate algorithm can the use the information of distance between vertices to generate an optimised, but not optimal Hamiltonian cycle [2]. The Ghosts state of patrol is broken upon entering line of sight of the Pacman, it will chase the Pacman until line of sight is broken. If line of sight is broken, it will search for the Pacman, but upon failing to do so will re-enter it's patrol state.

(Word Count 329)

2 Justification

A consideration when planning was to have the Pacman to navigate the maze by moving in the closest direction possible towards a given point of interest. This could be achieved by first determining the direction of a given point from

the Pacman, then what directions the Pacman agent can move from its current location, and then the most optimal direction can be calculated. However this approach does not consider the walls of the maze in that the most optimal direction may be into a dead end. Dijkstra's algorithm does not suffer from this however as its approach uses a form of breadth first search to find the goal state, T, before working backwards from T to find the optimal path back to S. This approach by design will filter out dead ends, as well as any deviations from the optimal path.

Another consideration was to have the Pacman consider the locations of Ghosts when using Dijkstra's, however, this was deemed unnecessary, largely due to how the game functions. When the Pacman is not carrying any pellets there is very little need for the Pacman to care about Ghosts, as dying to them will only hinder the Pacman by a single tick. On the other hand the Ghosts will be hindered by multiple ticks, depending on how far they are from their home position. This mean it works out in the Pacman's favor to run into enemy Ghost agents whilst trying to capture a pellet. Furthermore due to the path being taken by the Pacman being optimal there is no need to worry about Ghosts chasing them catching up, as the speed of one patch per tick is shared by both agent types. Furthermore is Ghosts were modeled the same as walls within the implementation of Dijkstra's algorithm there would be many scenarios in which finding the goal state would be impossible, as Ghosts have blocked all routes. Having to make exceptions would further complicate the algorithm as well as potentially making it incompatible for Ghost agents to use. Therefore the algorithm does not take account for the position of Ghost agents.

(Word Count 351)

3 Critical Evaluation

My approach comes with limitations in that large amounts of processing power is needed for the creation of Hamiltonian cycles for the Ghost agents. This comes in the form of large stutters in tick rate whenever a Pacman agent captures a pellet, as well as when first starting the simulation. This effect is worsened upon the increase the number of pellets, the number of Ghost agents or the size of the environment. However I have only tested the AI on my laptop with lower specifications, this issue may be resolved running the script on a device with higher specifications.

As to why the Hamiltonian cycles require a large amount of processing power if because in order to create a representation of the network every vertex is first compared to every other vertex to find the distance between the two, using the first half of the Dijkstra's algorithm. An optimised path is then generated by picking a random starting point and finding the nearest vertex to this point, saving that as the next vertex in the cycle and setting it as the current vertex to travel from. This is repeated until every vertex has been visited. By repeating this for every Ghost agent in parallel, the result is a loss in tick rate.

There is no solution to this in that there is no algorithm to guarantee the

finding of an optimal Hamiltonian cycle [2], other than creating every possible permutation of Hamiltonian cycles for the given network and choosing the smallest one, which would only further exacerbate the problems with processing power.

A challenge faced when creating Hamiltonian cycles was that they update every time the score changes for the opposing team. This caused an issue that when a Ghost agent was not in the Patrol state when an opposing Pacman agent captures a pellet, and was still not in patrol state when the Pacman agent picks up the last remaining pellet on the Ghost agents side of the map. The Hamiltonian cycle function would attempt to create a cycle for a network with no vertices causing the script to crash. Since the Ghost AI functions of of having pellets to travel to an additional state had to be created such that in the circumstances no pellets were available for the Ghost to travel to they would instead traverse the environment randomly, attempting to find the Pacman before they capture the last remaining pellet.

(Word Counter 403)

References

[1] Kevin Hartanto Ade Candra, Mohammad Andri Budiman. Dijkstra's and a-star in finding the shortest path: a tutorial. *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, 2020.

[2] Manfred Padberg Karla L. Hoffman and Giovanni Rinaldi. Traveling salesman problem. *Kluwer Academic Publishers*, 2001.