Ms Pacman vs. Ghost

Neha Patwardhan
Masters in Computer Science
Patwardhan.n@husky.neu.edu

Abstract:

In this report, I have explained nine algorithms by implementing them in the benchmark, Ms Pacman vs. Ghosts. Each algorithm is implemented in form of Pacman controller. I have explained performance of each controller and why it performed well or poorly by comparing and contrasting it with other controllers.

Depth First Search (DFS):

- **Working**
  I noticed that when applied DFS, Ms Pacman always moves upwards. Here by upwards I meant deeper in depth. In DFS, we explore nodes depth wise using Stack (Last In First Out) data structure. Because of this, Pacman always pops last node from the stack and then explore its children. So if the depth of node is d, depth of its children will be d+1. There is no backtracking because of which Ms Pacman tries to go upwards.
- **Analysis**
  Run Time: 24 sec
  **For run time, I first tried to write a code to find difference between start time and end time. But because of the threading, I never got end time from my code. So then, I ran algorithm several times and measured time using stop watch. So this run time is average of all the times I recorded.*
  High score: 560
- **Advantages**
  It runs faster than BFS (Breath First Search). The position of power pills is in such a way that Ms Pacman manages to eat power pills (around 2) and hence it gets better high score than BFS.
- **Disadvantages**
  Ms Pacman doesn't consider other environmental factors like ghosts or power pills before taking a path. So Pacman won't change its path even if a nonedible ghost is in his way or won't eat a power pill even if it's just besides the Pacman. This solution is incomplete as Pacman often gets stuck to the wall trying to move in one particular direction over and over again.

Iterative Deepening:

- **Working**
  When applied Iterative Deepening, I observed that, Ms Pacman first moves one level upwards depth wise (depth increments by 1) and then explores the path horizontally on that level and keeps on repeating this behavior.
  This is because, in Iterative Deepening, we explore nodes depth wise using Stack (Last In First Out) data structure but depth cut-off is incremental. Here, I started with max depth as 1 initially then incremented it by 1 on every iteration till it becomes 7.
  Pacman visits the nodes in the search tree in the same order as DFS, but the cumulative order in which nodes are first visited is effectively breadth-first.

Because of this, when reached the max depth cut off (basically on each level), Pacman starts exploring the path breadth wise.

- **Analysis**
  Run Time: 28 sec
  High score: 510
- **Advantages**
  Pacman runs faster than BFS (Breath First Search). This solution is complete and hence, Pacman doesn't get stuck to one particular position and stays alive for a longer time than DFS.
- **Disadvantages**
  The high score is lesser than DFS because Pacman often gets killed before eating any power pill. This is because Ms Pacman is oblivious to the environmental factors like ghosts or pills. So Pacman won't change its path even if a nonedible ghost is in his way or won't eat a power pill even if it's just besides the Pacman.

<u>**A*:**</u>

- **Working**
  In A* algorithm, I have considered two factors for calculating heuristics – edible Ghosts and Power Pills. If there are any edible ghosts available, I calculated heuristics based on the nearest ghost to the Pacman. If there are no edible ghosts then I calculated heuristics based on the nearest power pill from the Pacman.
  I have considered single cost associated with all the moves and that's 1. So, the actual cost associated with any move would be 1.
  Pacman takes the path where cost of move and heuristic is the least. She often moves greedily towards the nearest power pill and once she eats the power pill she moves greedily towards edible ghosts and eat a lot of them (3 on an average).
- **Analysis**
  Run Time: 36 sec
  High score: 2460
- **Advantages**
  Ms Pacman considers the environmental factors such as Ghosts, Pills and Power Pills (depending on heuristics) before selecting the path. If the heuristics are good, Pacman will always take the best possible path.
- **Disadvantages:**
  Similar to BFS, A* stores each function state in memory and hence requires a lot of space. This also slows down the Game in long run.

<u>**Simulated Annealing:**</u>

- **Working**
  Ms Pacman takes different and quite random paths each time when applied Simulated Annealing. Now which path it will take for sure, can't be predicted but I will discuss what influences the path selection process.
  Pacman will mostly select the move which is better than the current move. By better I meant whichever move leads to the better score. That means - suppose Pacman's current move is

'LEFT' and the next move is 'UP'. Then Pacman will select 'UP' if there is a pill, eating which Pacman will get a better score.
Now what if the next move 'UP' doesn't lead to a better score than the 'LEFT' move? Well, even then Pacman might take 'UP' move. This decision is made by probability of $e^{\Delta E/T}$.

- **Analysis**
  Run Time: 25 sec
  High score: 720
- **Advantages**
  This is a good option for optimization problems where heuristic methods are not available. This is better than hill climbing algorithm where Pacman often gets stuck trying to do the same move repeatedly just because the next move is not better than the current move (local maxima).
  Pacman mostly doesn't get stuck when implemented Simulated Annealing and is pretty fast.
  Because Pacman selects mostly the moves leading to the better score, Pacman achieves better high score than most of the algorithms which don't consider heuristics.
- **Disadvantages**
  Pacman won't work as good as the algorithms like A* that have heuristic methods, which are problem-specific or take advantage of extra information about the system.

Minimax:

- **Working**
  In Minimax, Ms Pacman is our max player and always tries to select the move that leads her to the maximum possible score. On the other hand, ghosts will always try to make sure that Pacman selects the move that will leads her to the minimum possible score. So the Pacman's path is completely depending on the position of the pills which is static and the moves of the ghosts which are random.
  So if nonedible ghosts starts coming towards the Ms Pacman, she tries to run away from them as that will lead her to the better score. If there are edible ghosts, Pacman tries to move towards them because that will lead her to the better score.
- **Analysis**
  Run Time: 50 sec
  High score: 2900
- **Advantages**
  Pacman works very well when applied Minimax. She achieves highest scores as compared to all other algorithms and stays alive for the longest time. Pacman selects that path which leads her to the maximum score.
  The minimax algorithm describes perfect play for deterministic, perfect-information games. The goal of the algorithm is to achieve the best payoff against best play.
- **Disadvantages**
  This is highly exhaustive algorithm. Thinking ahead for the rest of the game is impractical except for very simple games (like tic-tac-toe). This is because there can be a very large number of moves.
  In practice, I can only think ahead a few moves. This still generates a tree of moves and board positions, but one where the leaves are not necessarily finished games. Instead, I need to use my heuristics to estimate the values of these positions.

KNN:

- **Working**
  For KNN, Ms Pacman is provided with 3 features - Power Pills, Active Pills and Ghosts.
  The game copies are then advanced according to the moves influenced by these features for a certain depth. The top K closest neighbors are retrieved with their respective moves. The MOVE which is present maximum number of times is selected, and the Pacman takes this MOVE. Pacman first moves downwards as the closest power pill is in downwards direction and then she makes her way up (trying to move away from the nonedible ghosts) as the ghosts begin to approach her. The DOWN move seems to win the majority label for some time until the ghosts begin approaching and then, the UP move begins to win since the features in the game copies now push UP as a better move.
- **Analysis**
  Run Time: 20 sec
  High score: 510
- **Advantages**
  KNN is a very simple and easy algorithm to implement.
- **Disadvantages**
  KNN is not accurate in classifying and dependent on the value of K. It is very expensive to find the top K neighbors in a game state space. You never know, what the best value is for K.

ID3:

- **Working**
  The features considered for ID3 are to move Ms Pacman towards the closest Power pills, Active pills and edible Ghosts. The gain for each feature is to move Pacman in the direction which would lead to the highest score. Considering these features, Pacman first moves towards the closest power pill in the right corner in the down direction and then moves up because now the ghosts are edible. She then tracks down all the power pills and tries to consume as many edible ghosts as possible. However once the power pills are consumed, she then moves along the path of the active pills staying away from the ghosts.
- **Analysis**
  Run Time: 41 sec
  High score: 2010
- **Advantage**
  Ms Pacman receives a high score with ID3 since the gain selects the best possible feature.
- **Disadvantage**
  If the features are not chosen carefully, she does not play the game successfully.

Genetic Algorithm:

- **Working**
  In the Genetic algorithm, Pacman has an initial population of 4 individuals. These 4 individuals are set of randomly selected moves. There is a fitness function which evaluates the best of the 4

individuals and then generates the off springs by mutating that individual. The path taken by Ms Pacman is random as the initial population is very random. She doesn't reach far since the individual population moves are not based on any concrete attribute. It doesn't even consider environmental factors.

- **Analysis**
  Run Time: 16 sec
  High score: 410

<u>My Own Algorithm:</u>
I have created an algorithm which is a combination of A* and ID3 algorithms as Ms Pacman received very good score with these 2 algorithms. My motive behind developing this algorithm was to achieve highest score with Ms Pacman.

- **Description:**
  In A* we select any random factor for calculating heuristics. We don't know actually which feature will give us the best heuristics. I had faced this problem of selecting most efficient heuristic when I was working on A* algorithm.
  So I applied a part of ID3 algorithm to determine which attribute will give me the best heuristics. Whichever attribute is going to give me the highest gain will be selected for calculating heuristics. Here gain is high score as my goal is to achieve highest score.
  Once we get the most efficient feature, I will calculate my heuristics based on that and will apply A* algorithm further.
  I have considered edible ghosts, nonedible ghosts, power pill and active pills as attributes.
- **Working:**
  In this modified algorithm, each time, I select the feature which leads to the best score for calculating heuristic. Pacman takes the path where cost of the move and heuristic is the least. She moves greedily towards the power pill or edible ghost or active pill depending on which will lead her to the best score. So if the edible ghost is present nearby, Pacman will move greedily towards him rather than moving towards the power pill which is far or eats active pills which are nearby rather than running for a power pill which is far.
  As I have considered nonedible ghost as part of heuristic, Pacman tries to chase the ghosts. So, it won't try to go to power pill if there is ghost in between.
  This works very well, as I almost always get a possibly most suitable move. Pacman selects different paths based on positions and availability of pills/power pills and ghost movements (which is random).
- **Analysis**
  Run Time: 47 sec
  High score: 3280
- **Advantages**
  Ms Pacman considers the environmental factors such as Ghosts, Pills and Power Pills (depending on heuristics) before selecting the path. Also, heuristics are mostly good because it will lead to the maximum score so Pacman will always take the best possible path.
- **Disadvantages:**
  Similar to A*, this algorithm stores each function state in memory and hence requires a lot of space. This also slows down the Game in long run because of lots of computations before selecting the move.