Assignment: 2 Adversarial Search

Santosh Kumar Ghosh SBU ID: 10977062

Problem Statement:

It is classic adversarial game which can be modelled as a search problem using the game tree. Here the objective is for the Pacman to eat as much food as possible by avoiding the ghosts.

Reflex Agent:

For the first problem on reflex agent I used a combination of 4 factors to evaluate the desirability or utility of a successor state. The higher the value of a successor state the more desirable it is from the pacman's point of view. The four factors are

1. The distance from the closest ghost

The pacman would want to go to a state where the distance from the ghost is maximum so that its possibility of being eaten is less. So the position of each ghost was calculated and the distance from them was found out. Next the inverse was taken of that distance. This inverse value was multiplied by a negative weight. The negative weighted inverse tells us that the less the distance the less the score and hence less desirable the state.

2. The distance from the closest food

Eating food would give the pacman more points. So for each of the possible legal moves for the pacman the distance from the closest food was calculated. Then inverse of this score called foodScore in the code was taken. The inverse signifies that less the distance more desirable is the state as the next state.

3. The number of food left.

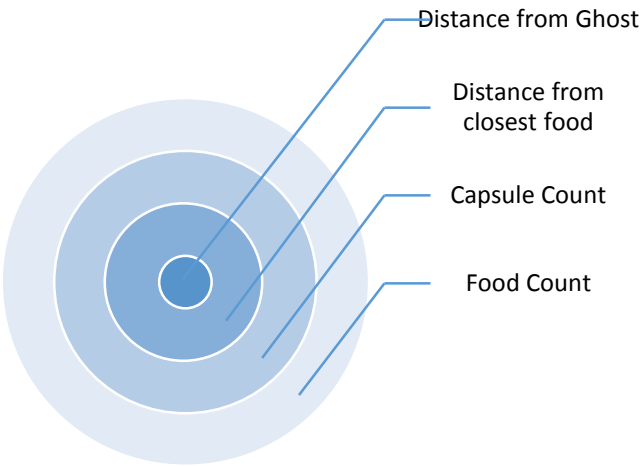
The motivation behind this parameter was to motivate the pacman to eat up food. So the number of food in each successor state was calculated and it was given a negetive weight, meaning the more number of foods left on the state the less desirable the state is.

4. The number of capsules left

The motivation behind this parameter was to motivate the pacman to eat up capsules. So the number of capsules in each successor state was calculated and it was given a negetive weight, meaning the more number of foods left on the state the less desirable the state is. Capsules earn the pacman a greater point. So we gave a greater penalty (negetive weight to the number of capsules) in this case.

Analysis of the parameters for the Reflex Agent:

The evaluation function for the reflex agent was tested with different combinations. It was observed that as we incrementally added the different factors to the function the scores of the auto grader improved substantially. This improvement was in terms of the time needed and the percentage pf the wins. The most dominating contributing factor in the performance was the number of distance to the closest ghost and the distance to the closest food. The other two factors marginally increased the efficiency of the algorithm.



Parameter	Average Win Ratio	Average Time Needed
Distance from ghost	**0.8**	**10.54**
Distance from food	**0.85**	**10.23**
Capsule count	**.95**	**10.01**
Food Count	**0.9**	**9.85**

Algorithms used for solving Pacman:

Two algorithms were used to search the game tree for finding a solution to the pacman problem. They are Minimax and AlphaBeta pruning.

Minimax Strategy:

The minimax pruning algorithm uses a recursive strategy to find out the next move of the pacman. It is a depth first strategy to find the solution. The minimax algorithm was run for the three board configurations of open classic and small classic.

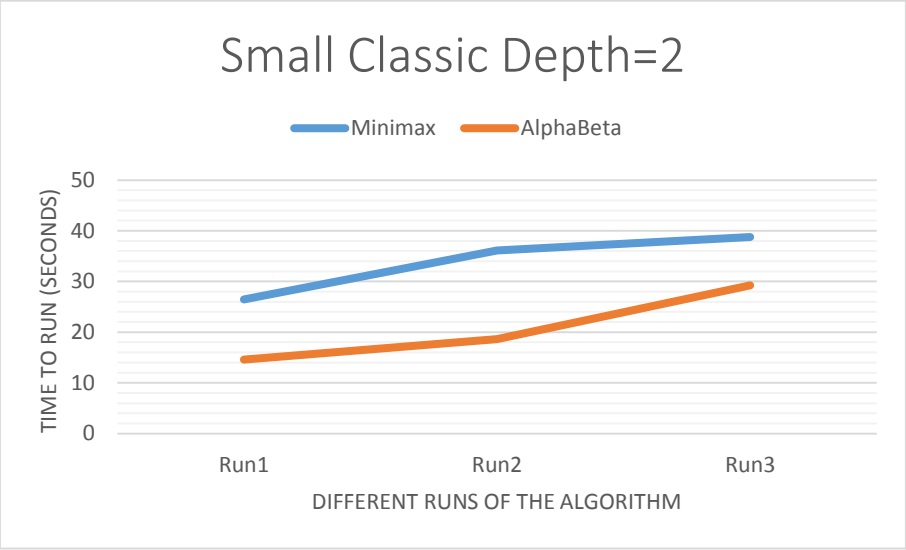
Alpha beta Pruning:

This is same as Minimax with an inbuilt optimizatuion optimization for reducing the search space. It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

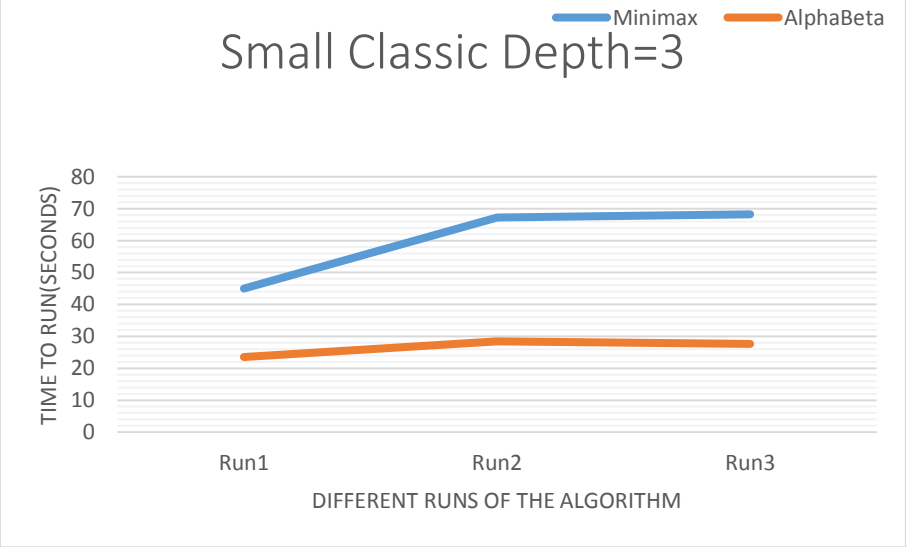
Comparative Study of the algorithms:

From the results obtained we can conclude that on the general the AlphaBeta Pruning algorithm gives better results in terms of running time and number of states expanded than Minimax algorithm.

The reason behind this is that alpha beta algorithm actively prunes many branches that are not needed. This leads to a lesser number of nodes expanded. This is true for all the depths. It is to be noted that although the time needed to complete the game is not hugely different in AlphaBeta and Minimax strategies the number of nodes expanded and hence the memory requirement is much more in Minimax.



ime Plot of different algorithms at depth 2



Time Plot of different algorithms at depth 3

The space required can be the difference between solving the problem or not. The space requirement increases drastically as increase the look ahead depth. This is intuitive as because the further down the game tree we go the more number of moves are possible.