CSL 309 LAB 2 REPORT

UNEET PATEL 2015CSB1038 JATIN GOYAL 2015CSB1014

PART 1

COAL BLOCKS AUCTION

We had to formulate this problem as a state space search for a single agent and implement hill climbing with random restarts. We had to define the state and its neighbours ourselves.

Definition of State Space and Neighbours of a State

We defined each state as the collection of some bids (1 or more) which do not conflict each other. By conflict we mean no two bids in the same state are from the same company and there is no common block between them. Now to get neighbour of any state we selected the bid which do not conflict to any of the bids present in the state. Taking the best neighbour among all neighbours and adding that to the current state to get new state.

Algorithm

The algorithm is hill climbing with random restarts in which we have done random restarts in the local search again and again till the time permits. When the time is over the best solution obtained from all the random restarts till that point is returned.
In each random restart we randomly generated a starting state. The initial state contains only one bid. To obtain the initial state first we make a 2D array in which all the bids are stored. In this array each row represents a bid which is stored same way as given in the input. So, at the 0^{th} index of each row the company id is stored. Now a random number is generated and then its modulus is taken with the number of bids. This number is considered as the first bid and taken as initial state.
Now to get the best neighbour of this state, first we find all the neighbours of this state. All the neighbours of a state are found such that no bid conflict with this state. We stored whether a conflict occurs between any two bids or not in a 2D array. Now, after finding all the neighbours we found the best neighbour. For this, the bid which has the highest revenue per block is considered as the best neighbour. We applied the hill climbing approach and climbed the hill greedily. Now, this bid is added to the current state and we got a new state. Then again we listed all the neighbours of this state as mentioned above and selected the best neighbour among all. While listing the neighbours, if a bid get conflicted with the state then we leave that bid and checked out the next one. This loop ends when all the bids have been checked. We did this process until we reach the state which has no neighbour left or not included in the state.
We did the whole process until time out. At the end, we return the max revenue obtained and the corresponding bids.
There were some modifications also made but not resulted in the best result. First, for the best neighbour we selected the neighbour having the maximum revenue overall. But not resulted in highest revenue.
Second we allowed some random moves in which it will take any one neighbour with 0.25 probability and actually best neighbour according to maximum revenue per block with 0.75

probability. But this modification resulted in better results that we obtained with just selecting the best neighbour according to maximum revenue per block. The reason behind this is that, with allowing some random moves the algorithm which get stuck in some hill jump to some other hill which leads to better result. It might happen that the algorithm jumps to that hill which is worse than the current hill but we allowed only 25% moves to be random and due to random restart it is very likely to go to better hill. So finally, we implemented this algorithm in our code.

PART 2

MULTI-AGENT PACMAN

QUESTION-1)

In this question we have implemented better evaluation function for the simple reflex agent. The function considers both food locations and the ghost locations to measure the state evaluation value. It takes the game-state and action to be performed as input. It avoids pacman from staying in the same position by returning very less evaluation value. Also if the ghost is within the distance of one step(manhattan distance) it again returns a high negative value. Now the the reciprocals of the sum of Manhattan distance of food particles from the pacman and number of food items remaining is taken. The final value returned by evaluation function is the weighted sum of both the reciprocals as described above. After tuning weights we observed that It is more inclined towards the reduced amount of food rather than the sum of the distance of food particles.

OBSERVATION:

We observed that the pacman wins all the games with score>=1000 on openClassic layout. Also we can summarise the behaviour of Pacman as mainly focusing on eating the food particles when ghost is not around.

QUESTION-2)

In this question we have implemented the minimax search agent where the Pac Man tries to maximize its utility while the multiple ghosts which behaves optimally tries to reduce the utility value of the pacman and avoid it from reaching the goal. Minimax function uses recursion where the search agent(Pac man) tries to maximize the utility by selecting the maximum valued successor while enemy ghosts selects minimum valued successor thereby trying to reducing the utility value for the search agent. Min function is implemented considering the given depth and the number of ghosts(as they can be more than 1). It recursively calls itself until all the ghosts have taken their turn and the search agent is ready to take turn. The function terminates by returning the utiltiy value of the corresponding state when it reaches at the required depth or there are no further moves possible.

CSL 309 LAB 2 REPORT

UNEET PATEL 2015CSB1038 JATIN GOYAL 2015CSB1014

OBSERVATION:

Although the Pac man loses most of the games it correctly explores the required number of states.

QUESTION-3)

In this question we have implemented the alpha-beta pruning on the game tree with same goal state to reduce the time complexity. We take the ranges for values at each step and prune branches of the game tree which are out of range. It results in the benefit of expanding the unnecessary states and thereby saving the time to reach goal state in game trees with greater depths.

OBSERVATION:

Although the Pac man loses most of the games it correctly explores the required number of states.

QUESTION-4)

In this question we have implemented the expectimax function different from the above functions. The above functions assumed that the ghosts are playing optimally and the pac man made the best move available deterministically. But the movements of the ghosts are random and may not be optimal hence we consider probability of the ghost taking a certain step and find the expected utility value using the probability and the utility values of each possible step. The function returns the final utility value as the expected value if all the moves of the ghost are equally likely.

OBSERVATION:

We observed that pac man fails to reach goal state most of the times. But this function may perform better than the above two functions because the movements of the ghost are random not optimal so there may be a chance that pacman makes the correct move to win. In the above two functions pacman make moves giving no chance of victory to the ghosts and ends up maximising the utility instead of reaching the goal state.

QUESTION-5)

In this question we implemented an evaluation function which which evaluates states rather than actions as in the reflex agent. The addition to the previous code is the manhattan

distance of the ghost in the scared state. Also the final value returned by the evaluation function includes the sum of the distance of all the food particles from the current position only if the ghosts are far away, along with the number of food particles
remaining and distances from the scared ghost.

OBSERVATION:

We have observed that our evaluation function allows the pacman to win all 10 games on the small Classic map with an average score greater than 1000. Pacman is allowed to chase ghost when they are in scared state.