Adroits - Team Description Paper

Vedant Tikku* and Somsubhra Bairi†

*201201002@daiict.ac.in
†201101056@daiict.ac.in

Abstract. This paper describes an approach to design an intelligent computer agent able to play the game of Angry Birds. The main aim of our strategy is to hit the maximum number of pigs and other objects using less number of birds to increase the overall score. We have defined a strategy which calculates the optimum trajectory based on the objects getting hit along the path of trajectory and few other factors.

Keywords: angrybirds, strategy, pigs, trajectory

INTRODUCTION

Angry Birds is an artillery game based on projectile motion where the agent controls a number of birds. The birds have lost their eggs to a group of pigs which seek protection in structures made up of variety of materials like wood, ice and stone. Using a slingshot the agent is supposed to shoot a bird towards the game scene and hit as many number of pigs and blocks as possible.

The game playing agent is based on a basic software making use of the Chrome version of Angry Birds and another application making use of computer vision, trajectory planning and a game interface component.

APPROACH TO THE PROBLEM

Introduction

For a well performing Angry Birds playing agent, it should be able to carry out the following tasks in decreasing order of priority:

1. Hit and destroy all the pigs in a level to qualify for the next level.
2. Use minimum number of birds to destroy all the pigs.
3. Cause maximum damage to the various objects occurring in the game environment like wood, stone or ice.

The first point is a necessary task to be carried out while the next two points will help in improving the score.

Methodology

First of all we determine the points lying within the dimensions of all the Angry Birds Game objects occurring in the environment including the pigs, wood, stone, ice, etc.

To satisfy the first and second points mentioned above, we calculate a trajectory which passes through maximum number of points lying within the area of the remaining pigs in each launch. In other words we take all the possible permutations of the pigs and calculate possible trajectories for each permutation.

We take into account the permutations for which the trajectory exists. Once this is obtained we rank the trajectories based on the number of pigs lying on their path.

We then filter the trajectory/trajectories which have maximum number of pigs from the rank list obtained above. Note that there may either be a single trajectory or multiple trajectories satisfying this property.

From the result obtained above, if there exists only one trajectory we choose that.

Calculating Rank of a Trajectory

In the previous step, if there exists multiple trajectories we assign a score to each trajectory based on the number of miscellaneous objects(other than pigs like ice, wood, stone, etc.) lying on that trajectory. This score is calculated based on weights assigned to each miscellaneous object.

The weights assigned to each miscellaneous object(ice, wood or stone) depends on the color of the current bird being shot. This is because a yellow colored bird has a different effect on stone than a red colored bird, a red colored bird has a different effect on ice than a blue colored bird, etc. Based on the effect a bird has on a type of block, weights are assigned to them on a scale of block having maximum impact to a block having minimum impact. In our previous strategy, we were not differentiating the effect that a particular colored bird has on a particular type of block.

Also the blocks may appear before the pig or after the pig on a particular trajectory. For a particular trajectory, if the block appears before the pig there are high chances of deflection of the bird and the pig not getting hit. While if the block appears after the pig then there is a chance of the pig as well as the blocks getting hit. Therefore more priority/score should be given to the paths for the trajectories in which more blocks occur after the pig. To do this, we assign a lesser weight to the blocks occurring before the pig in the trajectory while we assign a greater weight to the blocks occurring after the pig in the trajectory.

The rank of a trajectory is calculated as:

$$f(p) = \omega_1 + \omega_2 + \omega_3 + \omega_4 + ... = \sum_{i=1}^{n} \omega_i$$

In total, there are 3(wood, stone, ice) * 2(before, after pig) * 5(red, blue, yellow, white or black bird) = 30 possible weights. n is the number of objects on the trajectory. The value of n is restricted till 6 for better results.

Once the weighted score is calculated we rank the trajectories based on the score and choose the trajectory with maximum weighted score.

Discounting

As it is only guaranteed that the bird will hit the first object of the sequence associated with the trajectory and the effect of the objects following the first contact is unknown, we use a discount factor to evaluate objects i > 1. The ranking function with discounting is:

$$f(p) = d^0\omega_1 + d^1\omega_2 + d^2\omega_3 + d^3\omega_4 + ... = \sum_{i=1}^{n} d^{i-1}\omega_i$$

where d is the discount factor.

Reinforcement learning

So far, our agent will behave the same if a given stage is played again and again. If our agent loses on a particular stage, then it will lose again if it plays the same stage again.

So we need to ensure that the agent learns from the last time it played the stage so that it does not repeat the same mistakes again. We have to ensure that the agent performs differently the next time it plays the stage so that it does not lose again.

For this, the agent needs to store an image of the entire game play during the last play of the stage. Our agent will store the following things in memory:

1. A set of environment states S
2. A set of actions A
3. The reward set for every transition associated with corresponding actions R

Our agent then follows a stochastic model to determine the expected reward(re) if we are to follow the same action a again given an environment state s.

$$r_e(s,a) = r(s,a) * K$$

Here K is a binary random variable. If the expected reward is positive then we follow the action.

RELATED WORK

The strategy described in this paper is inspired by the strategy followed by team Angry Concepts who stood second in the AIBirds 2013 competition. Their work can be found here: http://aibirds.org/2013-Papers/Team-Descriptions/angryconcepts.pdf.

Our strategy differs from their existing strategy in the following ways:

1. Our strategy always has a pig lying in the path of the trajectory among all the objects. In the above mentioned strategy it is not necessarily true.
2. We calculate the trajectory based on location of the objects rather than traversing the entire trajectory space thus saving computation time.
3. Our first priority is to have maximum number of pigs on the trajectory first rather than just any object. Once we ensure we have maximum number of pigs lying on the path, only then do we consider other blocks.
4. Since we always have a pig on the trajectory the number of possible weights have been reduced to 30 from 45, since we eliminate the case where there is no pig. The only cases possible are the pig occurs after or before the block.
5. Our strategy has a reinforcement learning module to learn from the past play of a particular stage.

TEAMMATE DESCRIPTION

Somsubhra Bairi is a student at DA-IICT, India, an aspiring data scientist and an open source software enthusiast who hacks on KDE Software. He is a two time Google Summer of Code participant and has also participated and won in Microsoft Imagine Cup, 2013 and Google Cloud Developer Challenge, 2013-14. In his free time, he tinkers with new software technologies, his or others open source projects and tries to learn data science.

Vedant Tikku is a third year student at DA-IICT, India. He takes great interest in learning new programming languages and loves solving different algorithmic problems. His areas of interest include web development. During his free time, he loves to work on a few open source projects related to his interest.

ACKNOWLEDGMENTS

We would like to express our gratitude to Prof. Sourish Dasgupta for allowing us to do this project. We would also like to thank our Teaching Assistants and friends for helping us in the project.

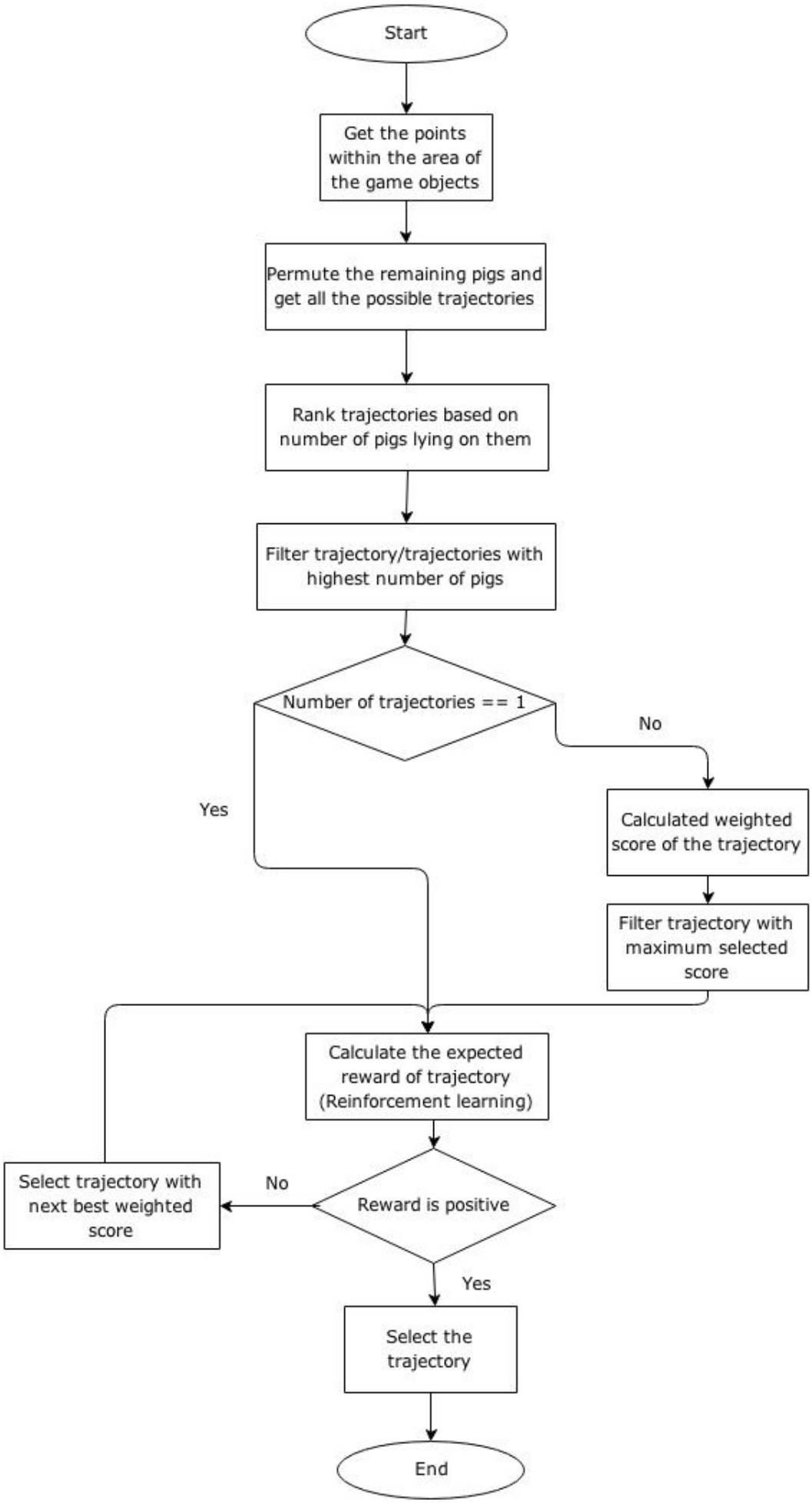

FIGURE 1. FLOWCHART