PacMan AI Controller

[For the pacman vs ghosts league]

Rodrigo Castro
Trinity College Dublin
MSc Interactive Entertainment Technology
cs7032: AI and Agents
28/01/2013
12324964
castrodr@tcd.ie

ABSTRACT
This paper provides a simple algorithm for maximizing the score of a PacMan AI controller in the Ms PacMan vs Ghosts Java-based environment. It includes two different strategies and a discussion of the results. Finally, a comparison is made between this and other well-known agent algorithms.

Keywords
AI, Agents

1. INTRODUCTION
PacMan is an arcade game of the 80's where the player controls the agent through a maze filled with pills and ghosts. Recently, the game was ported to Java, and a competition of agents was organised. This document describes our PacMan agent for this competition.

2. RULES OF THE GAME (AS OF JANUARY 2013)
The goal of Ms PacMan is to eat as many pills, power pills or edible ghosts as possible. Four different ghosts pursue Ms PacMan all over the maze. There are four different mazes. A life is lost if they touch Ms PacMan. The game is over after there are no more lives available. The game starts with 3 lives. One additional life is given at 10 000 points.

When Ms PacMan eats a powerpill, the speed of the ghosts is reduced in half and their direction is reversed. When Ms PacMan eats one ghost it goes into the lair for a while.

When all the pills have been eaten, the game moves on to the next level.

3. FORMALIZING THE PROBLEM
The Ms Pacman problem can be seen as a maximization problem. The bigger the score, the better it is.

$Max(Scores)$

To maximize the score, Ms PacMan should eat as many pills and ghosts and should survive for as long as possible.

4. OUR APPROACH
We have proposed the following behaviour in order to maximize the score:

Ms Pacman chooses a movement that will lead her to the closest and safest region having the greatest amounts of pills.

The behaviour can be summarized by the following algorithm:

Choose the best target node per direction:

$$Target_{t+1,direction} = \underset{t}{argmax} f_t(Pills_{region}, distance_{region}, dangerousnes$$

In some cases the targets will not lead to all possible directions. In that case, add the nearest junction as target within that direction. Then choose the best direction:

$$direction_{t+1} = \underset{direction}{argmax} Target_{t+1,direction}$$

4.1 Pills region
The pills are clustered by connected components. [1] The algorithm to find the clusters is as follows:

```
Init
Cs = List clusters, initialize with a single
  cluster containing all pills

Each time a pill P is eaten do:
  C = findClusterContainingP( Cs, P )
  Remove C from Cs
  For every possible neighbour Ni of P do:
    Create a new cluster Ci,
    Split C at Ni for Ci
    Add Ci to Cs
  End
End
```

[1] http://en.wikipedia.org/wiki/Connected_component_(graph_theory)

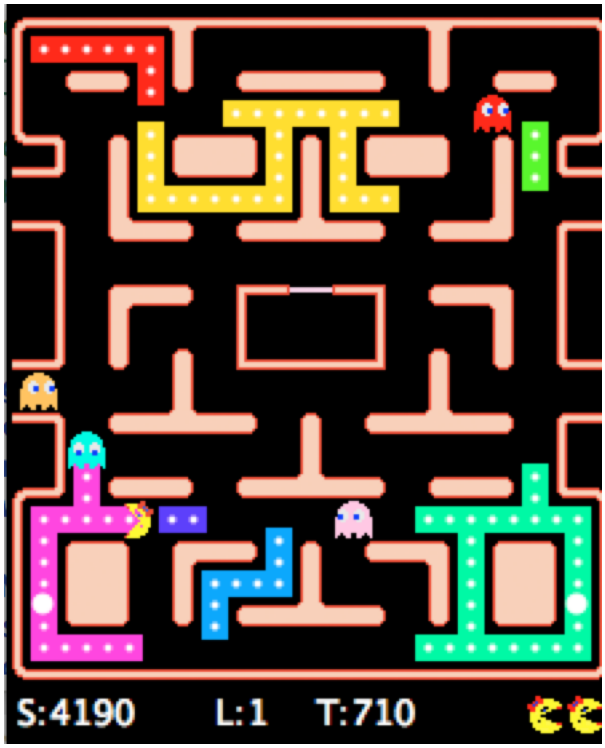

Figure 1: A sample of the clustering algorithm

```
Split C at Ni for Ci
  Remove Ni from C
  Add Ni to Ci
  For every possible neighbour Nj of P do:
    Split C at Nj for Ci
  End
End
```

It can be seen that the splitting algorithm is linear with respect to the number of nodes (When a graph is used as data structure). Testing if a node belongs to a cluster has a worst-case complexity of N. To make this function faster, a different data structure needs to be used. When using an ordered set as the main data structure for a cluster, the findClusterContainingP becomes a binary search (logarithmic complexity) and thus, splitting the cluster becomes linearithmic.

5. DISTANCE REGION

The distance to a region is computed as follows

$D_r = min(D_{shortest}(P_c, P_i))$

$D_{shortest}$ is the length of the shortest path between P_c, Ms PacMan's current position and P_i, an element of the pills region (cluster) i.

The shortest path is pre-computed using $a*$. The complexity of getting the shortest distance value becomes constant. Therefore finding the distance to a cluster of pills is linear.

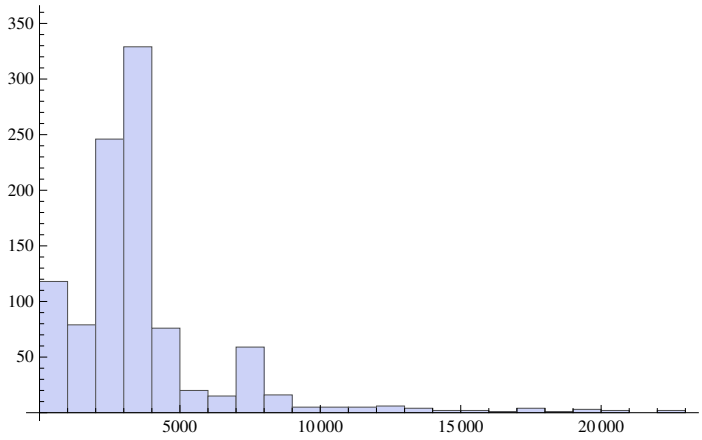

Figure 2: Score histogram for the Aggressive Ghosts controller using 1 strategy, 1000 iterations.

6. DANGEROUSNESS

The concept of dangerousness is the probability of finding a ghost on the same path as Ms PacMan. It is defined between 0 and 1.

In practice, the dangerousness has been simplified to the following cases:

- 1 if a ghost is headed to the same junction as Ms PacMan and the ghost is closer to that junction.
- 1 if a ghost is on the same path and opposite direction as Ms PacMan.
- 1 if the next junction has no ghost-free paths.
- 0 Otherwise.

7. RESULTS

The game was played a thousand times with different ghost controllers. For each controller we present the mean, the median, the standard deviation, the min, the max and the histogram of the scores.

7.1 First approach

The approach used here is the one described in the section "Our approach"

7.1.1 Aggressive Ghosts

Mean 3571.91

Median 3020

Standard Deviation 2956

Min 70

Max 22 670

This ghost controller shows the best average result for our Ms PacMan controller. Unfortunately, scores tend to vary a lot as it can be seen from the standard deviation. Some scores are above 20 000 and there are quite a few having very low scores. Running the experiment multiple times gives similar results.

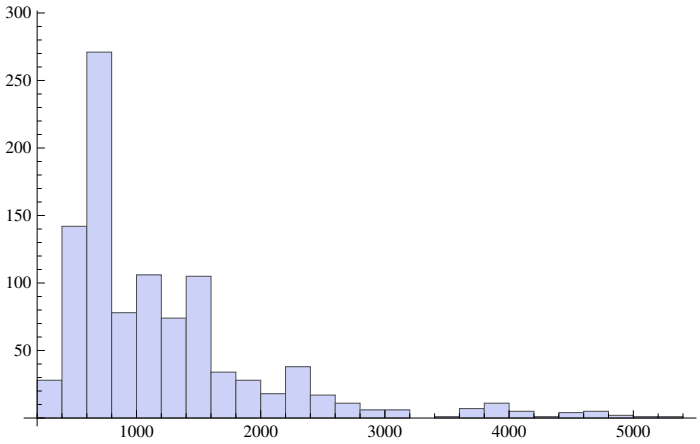

Figure 3: Score histogram for the Starter Ghosts controller using 1 strategy, 1000 iterations.

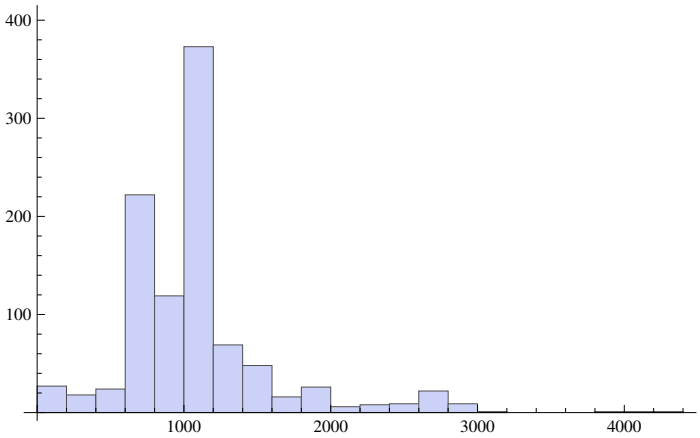

Figure 4: Score histogram for the Legacy 2 The Reckoning Ghosts controller using 1 strategy, 1000 iterations.

7.1.2 Starter Ghosts

Mean 1211

Median 935

Standard Deviation 831

Min 230

Max 5370

This ghost opponent gave the second best result for Ms Pac-Man. The average score is around 1211, with a peak near the 800 points. The standard deviation, which is quite large, is reflected on the histogram pattern. Running the experiment multiple times gives similar results.

7.1.3 Legacy 2 The Reckoning

Mean 1075

Median 1070

Standard Deviation 510

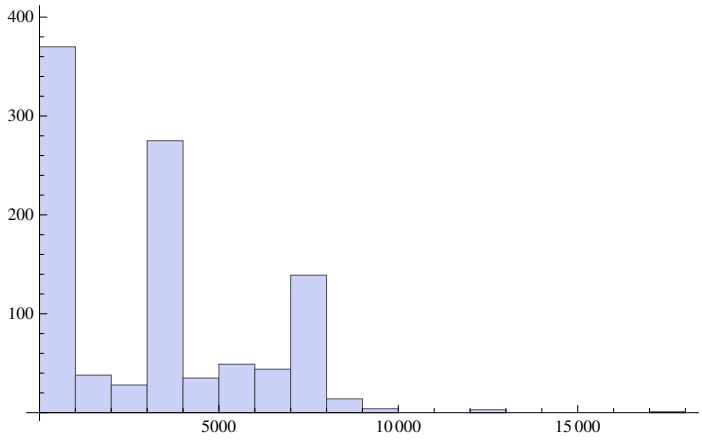

Figure 5: Score histogram for the Aggressive Ghosts controller using 2 strategies, 1000 iterations.

Min 70

Max 4310

This ghosts controller was probably the hardest for Ms Pac-Man. The average score being slightly lower than the rest of the experiments. The standard deviation is smaller, which can be observed on the previous histogram. Running the experiment multiple times gives similar results.

7.2 Second approach

In the second approach, a second strategy was added: if an edible ghost is within reach and it is safe to go there then go and eat it.

7.2.1 Aggressive Ghosts

Mean 3177

Median 3710

Standard Deviation 2744

Min 70

Max 17980

This time, three peaks appear probably due to the fact that ghosts where too close to each other when Ms PacMan had eaten a power pill and started chasing the ghosts.

Using 2 strategies for this controller resulted in a slightly worse average score than when using a single strategy.

7.2.2 Starter Ghosts

By using 2 different strategies against the Starter Ghosts controller, we managed to significantly improve the average score. We observe the values tend to concentrate around the average score.

Mean 2585

Median 1950

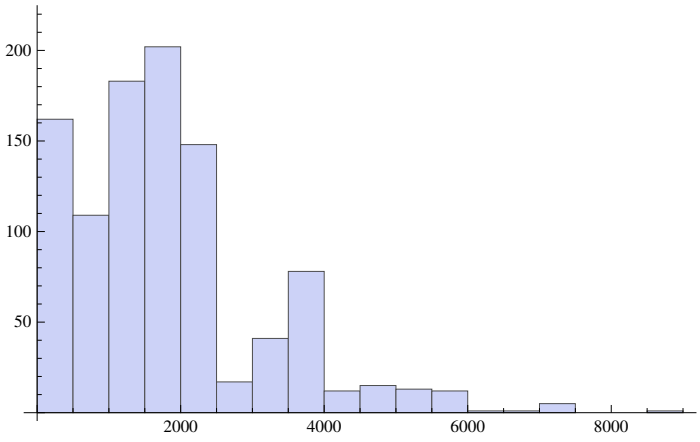

Figure 6: Score histogram for the Starter Ghosts controller using 2 strategies, 1000 iterations.

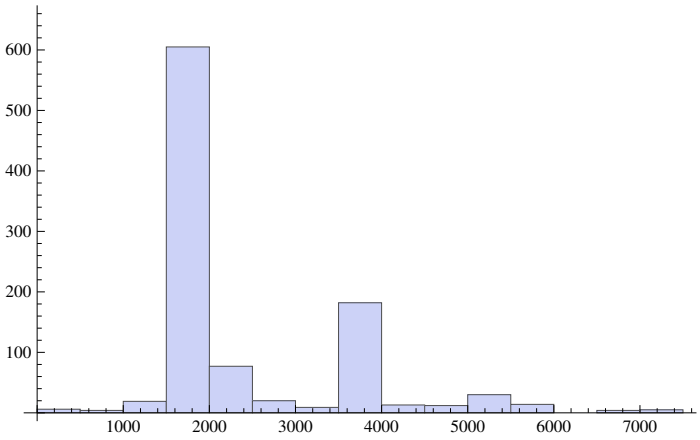

Figure 7: Score histogram for the Legacy 2 The Reckoning Ghosts controller using 2 strategies, 1000 iterations.

Standard Deviation 1178

Min 70

Max 7210

7.2.3 Legacy 2 The Reckoning

Mean 2555

Median 1950

Standard Deviation 1095

Min 330

Max 7250

Again, we observe three different peak scores as with the Aggressive Ghosts controller. And again, the average score and, more generally, the distribution of the scores is better than using a single strategy.

7.3 Discussion of results

7.3.1 First approach, one strategy

The Ms PacMan controller seems to behave correctly in the following simple conditions:

- When no ghosts are present
- When a ghost follows PacMan
- When a ghost is in front of PacMan
- When a ghost will get to the same junctions as Ms PacMan, either before or after Ms PacMan

In practice, Ms PacMan gets cornered very often. Something that could have been prevented if the dangerousness was more realistic. Furthermore, in some mazes, there are many junctions very close to each other which seems to be problematic as the current algorithm code explores adjacent nodes to Ms PacMan and adjacent paths to this nodes only. The code fails to detect nearby ghosts when at least two junctions are very close to each other. Sometimes, this forces Ms PacMan to oscillate and get eaten, this is why a run away strategy was implemented in case the ghosts get too close to Ms PacMan.

7.3.2 Second approach

Using multiple strategies seems to achieve better results in the general case, but may achieve poorly in some cases. We could certainly have used a single strategy and yet achieved the same results as when using two strategies, by simply adding a moving cluster for every edible ghost.

7.4 Further work

This AI controller is not particularly intelligent. It fails to provide an accurate overview of the maze and thus fails to avoid being cornered by ghosts. By simply modelling the dangerousness function more appropriately and with a deeper knowledge of the maze and the ghosts, we may achieve better results without changing the fundamentals of the algorithm. This work is yet to be done.

8. COMMENTS

The latest implementation of Ms PacMan controller can be found online at github [2]: src/pacman/entries/pacman/ MyPacMan.java There seems to be a bug in the dangerousness function where it wrongly computes a dangerousness value forcing Ms PacMan to go directly to a ghost (when both are very close to a junction). The run away from ghosts strategy is a work around for this problem.

9. DISCUSSION WITH RESPECT TO AI

Our Ms PacMan final implementation controller uses different strategies and agent architectures. It is a mixture between a reactive agent [3] and a reinforcement learning approach [4]. It is a reactive agent as it checks whether a ghost

[2]https://github.com/recastrodiaz/MsPacMan
[3]urlhttps://www.scss.tcd.ie/ luzs/t/cs7032/reactive-notes.pdf
[4]urlhttps://www.scss.tcd.ie/ luzs/t/cs7032/efeedback-notes.pdf

is near Ms PacMan and then locks its target to that ghost or runs away if it is too close. It is an evaluative feedback agent as it compares different alternatives by greedily choosing a target and then a direction. Sometimes, when a target does not evaluates all possible directions, other safe directions might be chosen instead. This last approach is very similar to the epsilon-greedy method used in the n-armbed bandit example.

10. CONCLUSIONS

Developing an effective intelligent and autonomous agent is quite complex. Abstract architectures certainly help defining the agent's behaviour, but in practice, these need a lot of tweaking to make them work as expected. Fortunately, creating an autonomous agent is a very rewarding experience.