Ms. Pacman AI controller

Jonas Flensbak
Georgios N. Yannakakis
Center for Computer Games Research
IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S, Denmark
{flensbak; yannakakis}@itu.dk

Abstract

This document describes a rule-based heuristic-search controller for the original Ms. Pacman entering the Ms. Pacman competition of WCCI 2008. The controller is implemented in an efficient screen-capture framework, which manages a lot of the functionality that would otherwise lie in the controller, thus, speeding up the development of new controllers. First a brief presentation of the framework is given followed by a detailed description of the controller.

Framework

The framework has been developed in .NET 2.0 and 3.5 for the core functionality required in a Ms. Pacman controller. The main elements of the framework include.

- *Screen reader*
  Fast and relatively reliable software that captures a screenshot and transforms it into a game state.
- *Game state*
  This is a representation of a Ms. Pacman game; it includes the maze type, a discrete representation of the map elements such as pills, power-pills, walls, and empty corridors, as well as the position and movement direction of the ghosts and Ms. Pacman. Moreover, the state of the ghosts (e.g. *hunting*, *edible eaten*) is represented. A pre-calculated map that contains the shortest distance between any two discrete positions in the grid map is also included along with the direction the controller will have to choose to get there.
- *Control program*
  A program that makes sure the Ms. Pacman game is positioned correctly on the screen, loads a user selected controller, and provides statistics about the running game.
- *Visualizer / Simulator*
  A program that can either visualize a game state or simulate the Ms. Pacman game by imitating the ghosts' controllers existing in the original game. The advantage of the simulator is that the game can achieve at least 10 times higher real-time speed than the speed of the original Ms. Pacman game on a 2.0 GHz processor, thus, making the application of machine learning approaches possible for this game.

Controller

The controller presented here is primarily based on the concepts of pill-hunting and short- and medium-range ghost avoidance. Initially a very simple controller, called *SmartPac*, was implemented to test the

framework. The *SmartPac* controller, implemented in approximately 100 lines of code, worked surprisingly well by achieving 9500 points on average with a maximum score of 15070. *SmartPac* worked simply from short-range ghost avoidance, only "seeing" ghosts within 4 discrete grid-cells. In broad terms the short-range avoidance works by eliminating the possible directions Ms. Pacman can move towards, starting with the most dangerous. If none of the directions are considered safe (i.e. all have a ghost within a distance of 4 grid-cells), then the direction that corresponds to the longest distance to a ghost is considered the least dangerous and is chosen. In *SmartPac*, if more than one direction is safe then the path that corresponds to the shortest distance to a pill is chosen. Even though *SmartPac* is a very simple approach and the generated Ms. Pacman behavior demonstrates great difficulty escaping when cornered by 2 or more ghosts, it works well because of its efficient pill-eating behavior. Before the ghosts exit from their starting position, *SmartPac* has already eaten a large portion of the pills and therefore minimizes the time in which all the ghosts are out hunting.

Several approaches were applied to improve *SmartPac*, but many Computational Intelligence (CI) techniques failed primarily because they generated less efficient pill-eating Ms. Pacman behaviors. It appears that CI generated behaviors which proved (at times) very good at escaping complex situations would also frequently cower in a far corner because they would sense the ghosts from far away. Another observation was that the top row of pills in the first Ms. Pacman maze (level 1 and 2) would in many cases be considered so dangerous to enter that the controller would often avoid it completely.

The controller submitted for the competition, called *SmartDijkstraPac*, builds upon the speed and effective short-range avoidance of *SmartPac*. In addition to this, a medium-range avoidance algorithm has been implemented (see Figure 1). Medium-range avoidance works by predicting the possible routes that the ghosts may travel to for the next *n* discrete grid-cells. In *SmartDijkstraPac* the compromise for *n* was for 7 turns. The medium-range avoidance algorithm is what gives the Dijkstra part of the *SmartDijkstraPac* name, as the prediction algorithm uses a modified Dijkstra's algorithm (Cormen, 2001) that determines the shortest route based on the probability of a ghost appearing in a grid-cell at the same time as Ms. Pacman. Since the speed of Ms. Pacman and the ghosts is variable to some degree, the predicted routes are game state dependent and become less reliable the further the prediction is (i.e. the longer the route). The controller uses the predicted routes to decide upon which direction is the least dangerous. Additionally edible ghosts are also hunted when they are close to Ms. Pacman as this, on average, increases the score although the diversion of hunting ghosts also seems to make Ms. Pacman less likely to complete the first two mazes without being caught.

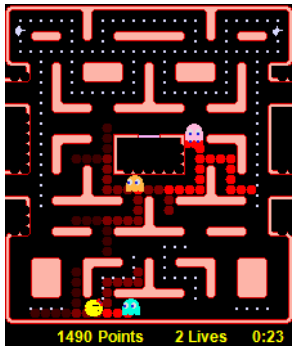

Figure 1: Predicted ghost routes; the more red a route is the higher the probability of a ghost being there.

The basic steps of the SmartDijkstraPac algorithm are as follows. If the algorithm chooses a direction at any step then no further steps are evaluated. When a direction is considered dangerous this is carried on into the next step.

1. Use short-range avoidance to eliminate dangerous directions:
   If only one direction is safe then choose that direction, or if no directions are safe then choose the least unsafe direction.
2. Use medium-range avoidance to eliminate dangerous directions:
   If only one direction is safe then choose that direction, or if no directions are safe then choose the least unsafe direction.
3. If any edible ghosts are relatively close, then choose a safe direction to the closest edible ghost.
4. Find the safe direction that has the shortest distance to a pill.

The additions over *SmartPac* have increased the average score of *SmartDijkstraPac* up approximately 2500 points, giving an average of 11832 points and a maximum score of 23780 as seen in the table below.

AI	Average	Min	Max	Games played	Lines of code
SmartPac	9470	4830	15070	64	~100
SmartDijkstraPac	11832	4600	23780	113	~400

Future Work

Relatively effective and reliable rule-based methods for short- and medium-range ghost avoidance have been implemented for controlling the Ms. Pacman agent. However, a reliable method for long-range ghost avoidance might provide an enhancement to the current algorithm. An effective long-avoidance method would ideally allow Ms. Pacman to move to parts of the maze that are less dangerous. A lot of work also remains on representing the game state in a way that takes the speed characteristics of the ghosts and Ms. Pacman better into account.

References

Cormen, L. e. (2001). *Introduction to Algorithms.* Cambridge: MIT Press.