A Simple Tree Search Method for Playing Ms. Pac-Man

David Robles and Simon M. Lucas, *Senior Member, IEEE*

***Abstract*— Ms. Pac-Man is a challenging game for software agents that has been the focus of a significant amount of research. This paper describes the current state of a tree-search software agent that will be entered into the IEEE CIG 2009 screen-capture based Ms. Pac-Man software agent competition. While game-tree search is a staple technique for many games, this paper is, perhaps surprisingly, the first attempt we know of to apply it to Ms. Pac-Man. The approach we take is to expand a route-tree based on possible moves that the Ms. Pac-Man agent can take to depth 40, and evaluate which path is best using hand-coded heuristics. On a simulator of the game our agent has achieved a high score of 40,000, but only around 15,000 on the original game using a screen-capture interface. Our next steps are focussed on using an improved screen-capture system, and on using evolutionary algorithms to tune the parameters of the agent.**

I. Introduction

Games have long been used as a test bed for Artificial Intelligence (AI) and Machine Learning (ML) algorithms. Traditionally, most successful game playing strategies have been achieved through clever programming an effective strategies, but machine learning techniques such as evolutionary algorithms are now often competitive, and in some cases, superior to hand-coded algorithms [1].

The most popular maze video game of all time is Pac-Man, and while it is no longer the newest or most advanced example of game development, it still provides a platform that is both simple enough for AI research and complex enough to require intelligent strategies for successful gameplay. In recent years, Ms. Pac-Man (a variant of the original game) has received attention from the Computational Intelligence (CI) community, and one of the reasons for this is the competitions that have been held in conferences such as the IEEE Symposium on Computational Intelligence and Games (CIG). These provide a level playing field to establish which CI techniques work best.

Several techniques from all CI paradigms have been applied to Pac-Man agents, such as Genetic Algorithms, Reinforcement Learning, Neural Networks, Fuzzy Systems, etc. However, hand-coded rule-based systems have proven to produce the highest scoring agents. Therefore, in contrast to previous research carried out in Pac-Man, where the main goal is exclusively to use the game as a platform for research with intelligence being an emergent property, the aim of this work is producing apparently intelligent behaviour using whatever techniques are appropriate to produce a high-scoring agent to participate in the 2009 IEEE Symposium on Computational Intelligence and Games (CIG) Ms. Pac-Man competition[1].

This paper describes the application of a tree search strategy for path finding to play Ms. Pac-Man in a simulator and in the original game (via screen capture). The rest of this paper is organised as follows: in Section II we review the previous research in Pac-Man, Section III provides details of our Ms. Pac-Man simulator and the screen capture adapter, Section IV describes our agent and the use of the tree for path finding, Section V presents details and results of our experiments and a comparison to successful agents from previous Ms. Pac-Man competitions, and Section VI provides a summary and conclusions.

II. Previous Work

Previous research in Pac-Man has been carried out in different versions of the game (e.g. in the original game, Ms. Pac-Man and customised simulators), hence an exact comparison is not possible, but we can have a general idea of the performances. Most of these works can be divided in two areas: agents that use CI techniques partially or completely, and controllers with hand-coded approaches used in previous competitions. We will briefly discuss the most relevant works in both areas.

A. Computational Intelligence approaches

One of the earliest studies with Pac-Man was conducted by Koza [2] to investigate the effectiveness of genetic programming for task prioritisation. This work utilised a modified version of the game, using different score values for the items and also a different maze. According to Szita and Lorincz [3], the only score reported on Koza's implementation would have been equivalent to approximately 5,000 points in their Pac-Man version.

Bonet and Stauffer [4] proposed a reinforcement learning technique for the player, using a very simple Pac-Man implementation. They used a neural network and temporal difference learning (TDL) in a 10 x 10 centred window, but using simple mazes with only one ghost and no power pills. Using complex learning tasks, they showed basic ghost avoidance.

Gallagher and Ryan [5] used a Pac-Man agent based on a simple finite-state machine model with a set of rules to control the movement of Pac-Man. The rules contained weight parameters which were evolved using the Population-Based Incremental Learning (PBIL) algorithm. They ran a simplified version of Pac-Man with only one ghost and no power pills, which takes away scoring opportunities in the

The authors are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom. (phone: +44-1206-872048; fax: +44-1206-872788; email: darobl@essex.ac.uk, sml@essex.ac.uk).

[1] http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html

978-1-4244-4815-9/09/$25.00 ©2009 IEEE