

IEEE WCCI 2008 Ms Pac-Man Competition

Neural Net based Controller for Ms Pac-Man

Nico Piatkowski, Georg Neugebauer, and Boris Naujoks¹

May 23, 2008

1 Introduction

MsNeuro is a hybrid neural network (NN) approach developed for playing Ms Pac-Man. It consists of a global decision making strategy based on NN (cf. Engelbrecht [Eng02]) to identify the right game situation and local heuristics to behave well in such. The NN continuously identifies one of three game situation. Based on this decision, a deterministic behaviour of Ms Pac-Man is encoded to solve the task coupled to the identified situation.

The main code of MsNeuro is written in Java using JDK 1.6. The movement code is written in C++. For connecting both components we used the Java Native Interface (JNI)². The java code simply loads the movement code supplied in a Dynamic Linked Library (DLL).

The MsNeuro project is based on a prior one introducing different computational intelligence (CI) methods to control the ghost within a Pacman game (cf. Danielsiek et al. [DEH⁺07, NEH⁺08]). Therefore, we developed and compared the performance of a NN and evolutionary algorithm approach (cf. Beume et al. [PHN⁺08]). Thus, a NN implementation was available for the competition's application. The source code was written in C++ and we decided to reuse this code leading to the solution described above.

Before this final work we tried several different approaches. For example, we deployed an evolutionary algorithm and a neural net just for moving Ms Pac-Man around. This approach did not provide the expected great success. We also tried a neuro-evolution approach. This failed as well, due to an increased need to learn special situations and the implied slowness during the learning stage of the NN.

2 Screen capture process

For Image Processing and Controlling Ms Pac-Man, we used the code base from the kit mentioned on the competitions web site³. This way, we receive the array of integers filled with the pixel map. Next to the functions from the original tool, we added some functions to improve the screen capture capabilities:

- New wall detection: the pixels near Ms Pac-Man are analyzed and an improved wall detection software is invoked.
- Closest ghost: like the existing "closestpill" function, the position of closest ghost is calculated during image processing. Based on the positions of the closest ghost and Ms Pac-Man, the Euclidian distance is determined.
- Closest edible ghost (like closest ghost).
- Closest powerpill (like closest ghost).
- Fruit detection: a fruit is recognized during image processing, if one is available on the screen. Again, he positions are derived and the Euclidean distance is calculated.
- Farthest pill (like closest pill), but this parameter in not used in the current implementation.

¹Chair for Algorithm Engineering, Faculty of Computer Science, Dortmund University of Technology, Germany.
Email: {forenames.surname}@tu-dortmund.de

²see <http://java.sun.com/j2se/1.5.0/docs/guide/jni/index.html> for details

³see <http://cswwww.essex.ac.uk/staff/sml/pacman/PacManContest.html> for details

Unfortunately, the screen capture process is not perfect. This sometimes leads to unpredictable behavior (see section 4).

In the following, the available parameters for decision making are listed:

- ghosts: positions (X,Y coordinates), edible flag,
- Ms Pac-Man: position , powerpill flag,
- closest pill position, farthest pill position, closest powerpill position
- closest ghost position, closest edible position, fruit position
- pill count, powerpill count
- wall code, laststep

From these parameters, we determined six to be the input values for our NN. This is described in more detail in the following section.

3 Controller for movement and Performance

The movement of Ms Pac-Man is depending on the current game situation. The movement in each of these situations is coded in heuristics. We identified the following situations:

- `eatClosestPill`
This is the general state of Ms Pac-Man, where she tries to minimize the distance to the closest pill. This is performed by some simple algorithm that also allows for degradation, if the direction towards this pill is blocked. Furthermore, ghosts are handled like walls, the direction towards them is blocked allowing only feasible directions maximizing the distances to the ghosts.
- `eatGhosts`
As far as this situation is recognized by the NN and the power pill is not active, Ms Pac-Man minimize the distance to the closest power pill. If the power pill is active, the closest edible ghost is chased. The algorithm from above is incorporated again.
- `eatFruit`
Recognizing this situation, Ms Pac-Man tries to eat the available fruit.

Summing up, we encoded one special distance minimization algorithm that is called with different objectives provided by the NN. For different game conditions, special strategies for avoiding collisions are invoked additionally.

For recognizing the correct situation, a neural network controller is used. The inputs are the euclidean distance regarding all four ghosts, the closest pill and the fruit. Overall there are six inputs and three outputs. We trained a neural net with six hidden neurons leading to a 6-6-3 architecture. The training set consists of 280 patterns derived from extremal situation within the game. Such situations can be:

fruit near and ghosts far away \Rightarrow `eatFruit`

all ghosts near Ms Pac-Man and power-pill near \Rightarrow `eatGhosts`

In Addition, we implemented a heuristic for avoiding collisions with walls or ghosts. This reduces the number of possible directions if ghosts or walls are in range. The performance of the controller framework is quite good, although not perfect due to the problems listed in the following section.

Unfortunately, the performance of MsNeuro holds a big variance. This leads to a number of points, which is not predictable beforehand. The maximum number of points reached until now is 14850. The highest level was level 3. Nevertheless, a minimal amount of approx. 3500 points was reached in every run. In the average case, MsNeuro achieves a score of 6000-9000 points. More precise results are not possible due to the mentioned variance in performance.

4 Known problems

Several unsolved problems are observed while working with the compound system:

- delays in the screen capture process leading to unexpected and unexplainable behaviour,
- failures in the screen capture process producing pathological behaviour of the whole system,
- different performances on different machines, here we tested on MS Windows and Mac, and
- the heuristic for avoiding collisions with respect to multiple ghosts does *not* work in a perfect way.

5 How to run

For MS Windows, the following steps are needed to run MsNeuro:

1. Extract the data from the msneuro.zip,
2. copy the proper DLL and adjust the system variable PATH pointing to the DLL folder,
3. start Ms Pac-Man and adjust the window to the position (6,46) (upper-left corner), and
4. finally start the main method of the class MsPacInterface:
cmd, go the folder “controller” from the extracted data msneuro.zip, then start with
java -cp bin pacman.MsPacInterface

6 Summary

Overall, we found an satisfying solution for the interesting competition at hand. Unfortunately, we have not been able to solve the recognition problem to identify each entity with the screen capture process to our expectations. Nevertheless, the results are encouraging and the research will be continued to overcome the drawbacks. Because the neural net controller works very good and gives a good approximation for choosing the correct situation, we thought of methods anticipating entities behaviour. This behaviour can be compared to the one received from screen capturing.

Unfortunately, we have not found a perfect solution for collision avoidance in the non-deterministic environment and we spend much time writing proper heuristics for each situation. These are also points, where the current implementation can possible be improved.

References

- [DEH⁺07] Holger Danielsiek, Christian Eichhorn, Tobias Hein, Edina Kurtić, Georg Neugebauer, Nico Piatkowski, Michael Puchowezki, Jan Quadflieg, Sebastian Schnelker, Raphael Stüer, Andreas Thom, and Simon Wessing. Zwischenbericht der PG 511. Technical Report of the Collaborative Research Centre 531 Computational Intelligence 236/07, Technische Universität of Dortmund, 2007. German Language, downloadable at <http://sfbc.uni-dortmund.de/Publications/Reference/Downloads/23607.pdf>.
- [Eng02] A. Petrus Engelbrecht. *Computational Intelligence An Introduction*. Wiley-VCH Verlag, Weinheim, 2002.
- [NEH⁺08] Georg Neugebauer, Christian Eichhorn, Tobias Hein, Edina Kurtić, Holger Danielsiek, Nico Piatkowski, Michael Puchowezki, Jan Quadflieg, Sebastian Schnelker, Raphael Stüer, Andreas Thom, and Simon Wessing. Final Report of PG 511: CI in Games. Technical report of the collaborative research centre 531 computational intelligence, Technische Universität of Dortmund, 2008. to appear, see <http://www.pg511.de/> for details.
- [PHN⁺08] Nico Piatkowski, Tobias Hein, Boris Naujoks, Georg Neugebauer, Nicola Beume, Mike Preuss, Raphael Stüer, and Andreas Thom. To Model or Not to Model: Controlling PacMan Ghosts Without Incorporating Global Knowledge. In *Proceedings of the 2008 IEEE World Congress on Computational Intelligence*, 2008. accepted for publication.