Rational Ms Pac-Man Agent based on Cellular Automata with Dynamic Update Value Parameter

Erenay Dayanık
Department of Computer Engineering,Middle East Technical University,Ankara,Turkey
e1819192@ceng.metu.edu.tr

Abstract

Throughout the decades, video games such as Super Mario[1], Starcraft[2] and TORCS [3] has been used as a test-bed for new AI techniques. Similar to these games, Ms. Pac-Man[4] video game also has been used to test such bleeding edge algorithms. The reason behind this fact arises from the nature of the video games which provides challenging and non-deterministic environment. In this paper, we propose a new technique to create yet another rational agent for Ms. Pac-Man video game. We call it *Pac-Man-CA0561*. Our proposal is the modification over the existed cellular automaton based controller technique by using dynamic update parameter for Ms. Pac-Man video game.

1 Introduction

Designing rational agents for video game controllers, has always been one of the active sub-areas of AI because, it makes game much more appealing and agents designed for such games can be easily adapted to other usage areas. By means of rational agent, we refer to agents that always chooses to perform an action with the optimal expected outcome for itself from among the all feasible actions. The biggest goal is to create an agent that cannot be distinguished from the real users. However, it is quite complex problem and for the time being, even the most strongest AI players are outperformed by the real human players. Therefore, many people still try to improve performance of their video game agents and try to find new techniques. Our design basically depends on Cellular Automaton to build a representation of the Ms. Pac-Man environment at each time at which the agent decides its next move. It uses Darer's algorithm[5] as a basis block. Basically, when agent determines its next move, it selects the best promising available direction in the environment by looking the values of agent's neighbour cells. These values are modified during the update procedure of the Cellular Automaton. The number of updates in Cellular Automaton could be either static, i.e. constant number of updates at each time step or it could be dynamic. Unlike Darer's approach, in our implementation we selected to use the latter one, since dynamic update parameter may increase the efficiency of the agent.

2 Related Work

The previous works on designing rational agents for Ms. Pac-Man can be categorised under two groups that are based on human-defined rules and evolutionary computation.[6]

In the first category, the agent will get high scores if the designer know the subject very well. The advantage of this approach is its simplicity. However, it brings another complexity with it. The rules which will be implemented can be very expensive and complex since all of the situations that may appear during the game should be handled with these rules. The complexity of the rules depends on the specific goals which want to be achieved.This fact is one of the biggest disadvantages of rule based agents. Furthermore, if the designer want to change goals at later, he or she have to rewrite the rules.

In the second category, there are several promising techniques such as the one developed by Alhejali and Lucas.[7] Unlike rule-based systems, the techniques in this category are capable of handling uncertain environments however, these algorithms consumes too much times before starting to obtain high scores.

In the algorithm, all of these disadvantages has been overcome. Firstly, we do not need to write all possible situations as a rule and thus complexity does not depend on the type of goals. Consequently, in case of any change in the goals, we do not have to rewrite the rules. Secondly, the time required for the algorithm is not too much. Specifically, the agent is able to make a decision at every 40 milliseconds which is quite small.

3 Problem Definition and Algorithm

3.1 Task Definition

Pac-Man-CA0561 is proposed to use as an rational agent for Ms. Pac-Man character in the Ms.Pac-Man game. The algorithm takes environment as an input from the simulator and produces a direction, in which the Ms. Pac-Man will move at the next time step, as an output. Formally:

Input: 2D Maze with the information of content of each cell
Output: Direction which is one of the Left,Right,Down,Up

3.2 Ms. Pac-Man Game Rules

In the game, player (Ms. Pac-man) have to move around the maze and collect all of the Pills(small dots), Power Pills(large dots) while avoiding the ghosts at all times. Once all of the pills have been collected the game moves forward to next level however if at any point you come into contact with a ghost, you will loose a life. If you loose all of your lives the game will end. When you consume a Pill, your score will increase, when a Power-Pill is consumed the ghosts will turn edible allowing you to consume them.[8]

3.3 Algorithm Definition

The algorithm in *Pac-Man-CA0561* utilises the *Cellular Automaton* system. *Pac-Man-CA0561* uses the Darer's algorithm[5] as a basis and determines iteration numbers dynamically. In this system, the environment is perceived as a 2D cells and each cell contains a discrete value. The rules of the automata utilises these discrete cells' values to generate complex behaviours.[9]

In the algorithm, while representing the environment, we have used several symbols to represent each of the elements of the game. The following table (table-1) summarises these symbols.

Game Element	Symbol
Ms.Pac-Man	@
Pill	p
Power-Pill	P
Ghost	G
Edible Ghost	E
Empty Cell	e

Table-1: Used symbols in the algorithm

In *Pac-Man-CA0561*, unlike the general cellular automaton system, each cell contains two different values. These are normal *cell values* and *decay values*. At the beginning of each decision, the cell values shows which of the six elements in table-1 occurs in that cell. Once, this has been accomplished, the algorithm iterates N times (N is predefined number) over the environment and with each iteration these values are spread through the neighbour cells if they pass the domination rules. After these N iteration occurs, agent decides its next move based on the cell values of its neighbour cells. The pseudocode for the algorithm is given in figure-1. This process are similar to idea of general cellular automaton technique in which at each iteration, cells try to modify value of its neighbour cells based on several domination rules. The domination rules of the *Pac-Man-CA0561* are shown in *table-2*.

According to table, if a cell contains, for example, P as a cell value, which means there is a Power-Pill in that cell, the cell pass its cell value to its neighbour cells if these neighbours contain "*G*" or "*p*" or "*e*" as a value. Note that, the cell whose value is either "@" or "*e*" cannot pass its value to any neighbour cell.

```
// Nvalues: predetermined most promising N values
// maze : the representation of the maze at the time of decision.
function PacManNextMove(maze,Nvalues ):
    iterationNum=possibleNvalues.selectRandomly()
    while iterationNum is not 0:
        for each cell in maze:
            if cell is not empty:
                do cell.dominate( cell.neighbours)
        end for each
        iterationNum-=1
    return bestMove(Pacman.neighbours)
```

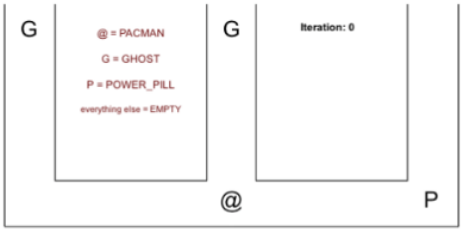
Figure-1: Pseudocode for the Ms. Pac-Man-CA0561

Element	Dominates
@	{}
p	{e}
P	{G,p,e}
G	{G,p,e}
E	{P,G,e,p}
e	{}

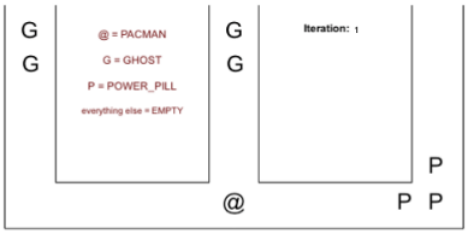
Table-2: Domination rules used in the algorithm

In our project, we use the simulator of the *Ms. Pac-Man vs Ghosts League*[10] as the simulator. It provides game engine,the game state and visual effects. However, it does not contain features of our approach such as cell value and decay value and therefore, we have added them on top of the simulator.

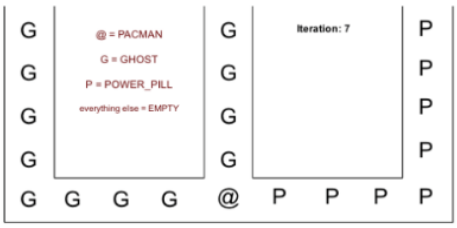
In figure-1, there is a toy example which represents the environment at the initial step, after first iteration and seventh iteration (N is equal to seven) of Cellular Automaton had been completed. Note that, internal iterations have no direct effect on the final decision of the agent but it is shown in figure-2 to trace algorithm easily. In this example, at initial step, it is assumed that two Ghost(non-edible), a single Power-Pill and the Ms. Pac-man exist in the game environment. Other than these, all cells are empty. After the first iteration, both of the cells with "G" cell value, dominates their neighbour cells' cell values. Similarly, cell with a "P" cell value also dominates its neighbours. (These are expected results since both "P" and "G" are capable to dominate "e" according to table-2). At the end of seventh iteration, the Ms. Pac-Man will decide to its next move, which will be moving right cell because the priority of "P" is higher than "G".This selection will move it away from the ghost and direct it to cell with Power-Pill. This decision is made based on the neighbour decision rules shown in *table-3*.



2.a: Initial Step



2.b: After first iteration



2.c: After seventh iteration

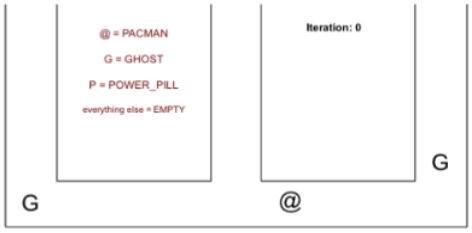
Figure-2: An example to environment of the Ms. Pac-Man game and process of the algorithm

Priority	Cell Value
1	E
2	P
3	p
4	e
5	G

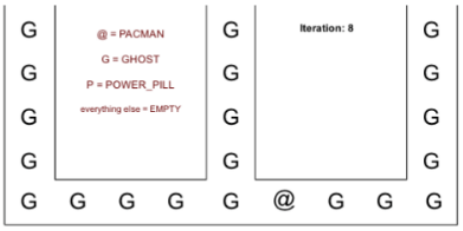
Table-3: Neighbour decision rules

Pac-Man-CA0561 is capable of handling situations like in figure-2 by only using cell value and neighbour decision rules, however, there are some exceptional cases in which, the Ms. Pac-Man cannot make a decision for its next move unless we use cell's decay value property. Decay value of a cell is the decimal number which represents the distance between the cell which contains this decay value and the cell whose cell value propagates to the cell. The environment in figure-3 represents the situation in which Ms. Pac-Man need to use decay value in order to select its next move. The initial representation of the environment is shown at figure-3(a). In figure-3(b), we represents the point of Ms. Pac-Man without any use of decay value property. In there, both neighbours of current Ms. Pac-Man cell contains "G" as a cell value and therefore, it cannot decide which of these neighbours will be its next place. However, we can see that(as observers) the ghost at the right side is much closer than the left side and hence, Ms. Pac-Man should move to left. Aim of the decay value is exactly giving this information to the agent. Since, the ghost at the right side of the maze are further to the agent than the other ghost, the right neighbour of the agent's current position will have less decay value than the left neighbour of it. As a result, the agent now knows which way is safer to go (neighbour with the least decay value if all neighbours are ghost or the cell with highest decay value if all of the neighbours are not ghost). After all, if all of the neighbours of the Ms. Pac-Man cell's contains same cell value, the next move will be determined based on neighbours' decay values.

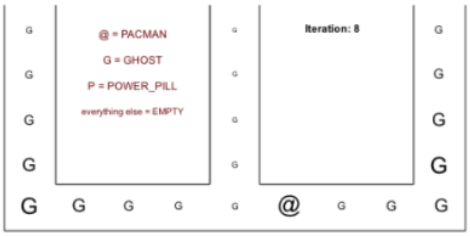
The number of iterations that are required to select the next move is determined dynamically. For this purpose, the algorithm is run at multiple times with different N values. We set the N to best promising value at the beginning of each game. After that, through the game, N is updated according to state of the game. For example, if there are edible ghosts in the maze, the update number is increased so that the agent can detect the existence of edible ghosts. Before the next move, N is again set to initial value and algorithm continues in this way until the game is finished.



3.a: Initial Step



3.b: View of environment from the point of Ms.Pac-Man after iterations complete without decay value



3.c: View of environment from the point of Ms.Pac-Man after iterations complete with decay value

Figure-3: An example of environment of the Ms. Pac-Man game for showing aim of the decay value property

4 Experimental Evaluation

4.1 Methodology

In order to evaluate the algorithm, we will run the game multiple times and the maximum, minimum and average scores along with the standard deviations will be calculated. By running multiple times, we are trying to prevent from the results of non-deterministic nature of the ghosts of Ms. Pac-Man video game.

Results will be compared with the other rational agents developed for the Ms. Pac-Man. One of these will be Darer's algorithm which is the basis of our algorithm. Furthermore, the behaviour of the agent will be monitored in certain scenarios to determine whether the agent reacted in the way we wanted to certain situations.

4.2 Results

We have obtain the following results by running Pac-Man-CA0561 30 times. At each time, the agent start with 2 extra lives and the maze contains 4 non-deterministic ghosts. The game consist of 3 different levels however; Pac-Man-CA0561 was able to move second level at most while we collected the results. In the problem we have dealt with, there was not a ground truth and therefore, we need to evaluate performance of Pac-Man-CA0561 by comparing it with other techniques that were created for the same purpose. In this manner, we have compared the results of our algorithm with the Darer's solution which is the foundation of Pac-Man-CA0561, nearest pill algorithm in which the agent always move towards the nearest pill and the random move algorithm in which directions determined randomly. Below in *table-4* , we have represented the results of all mentioned algorithms. The results are compared in terms of average score, minimum and maximum scores and standard deviations that are obtained over 20 distinct trials. Furthermore, in figure-4 we have showed the results of each algorithm for each experiment in a chart.

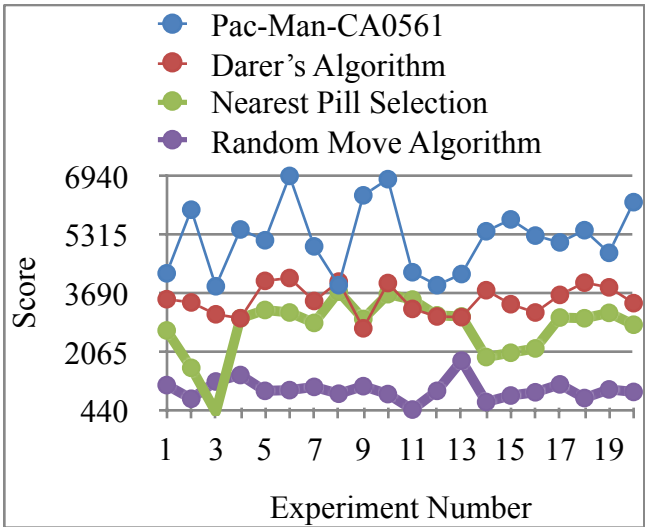


Figure-4: Scores obtained in each experiment

Algorithm	Pac-Man-CA0561	Darer's Algorithm	Nearest Pill Selection	Random Selection
Avarage Score	5204.0	3481.5	2869	988
Maximum Score	6940	4100	3720	1800
Minimum Score	3870	2700	1600	440
Standard Deviation	958.866	415.353	562.053	283.611

Table-4: Neighbour decision rules

4.3 Discussion

In this section we interpret the results we have represented in section 4.2. According to *table-4,* Our algorithm Pac-Man-CA0561 was able to obtain the highest maximum score with 6940 points and the highest minimum score with 3870 points among the others. However, the more significant result we can obtain from the table is that, Pac-Man-CA0561 also gets the highest average score. It is more important because, the behaviour of the ghosts are non-deterministic and therefore getting the maximum score may be result of a chance. Getting maximum average score,on the other hand, shows us that our implementation provides the best performance on the average case. In this way, we are able to talk about performance of the algorithm without thinking about the performance of ghosts.

We cannot obtain any valid results from the figure-4 because, there is not any meaningful pattern in the figure. The only valid conclusion we can get from the figure is that the performance of the agent is independent from the previous runs.

As a result of visual testing, we can say that performance of the cellular automaton based agent is highly dependent the content of the maze. When the sparseness of the maze increases i.e. when the number of pills and power pills decreased, cellular automaton performance start to decrease significantly since empty cells are not able to dominate content of their neighbours and thus agent cannot deduce strong inferences from the cellular automata.

5 Conclusion

Using Cellular Automaton technique with dynamic update parameter rather than static update parameter improves performance of the agent considerably. With this version of the agent, it is capable of getting higher scores in terms of both average case and boundary cases. However, the agent starts to locked in loops while the pills in the maze consumed. Therefore, through the end of the game, nearest pill algorithm is used to find the most promising direction and this modification makes the agent more stable. As a result, agent based on cellular automata with dynamic update parameter for Ms. Pac-Man video game behaves better than original approach in terms of both average and best cases and it worths to improve further.

6 Future Work

In the project, when the agent selects her next move by using nearest pill selection algorithm, the algorithm does

not keep track of the position of ghosts. Therefore, it may cause agent to loose her lives. As a future work, controlling ghosts before selecting the appropriate action may be added.

Bibliography

[1] Mario AI. http://www.marioai.org/, Julian Togelius.

[2] StarCraft AIIDE Competition. http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/,Michael Buro, David Churchill.

[3] TORCS - The Open Racing Car Simulator. http://torcs.sourceforge.net/index.php, Bernhard Wymann.

[4] Ms. Pac-Man vs Ghosts League. http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html,Game Intelligence Group, University of Essex.

[5] Darer, A., & Lewis, P. (2013). A Cellular Automaton Based Controller for a Ms. Pac-Man Agent. *arXiv preprint arXiv: 1312.5097*.

[6] Oh, K., & Cho, S. B. (2010, December). A hybrid method of Dijkstra algorithm and evolutionary neural network for optimal Ms. Pac-Man agent. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on* (pp. 239-243). IEEE.

[7] Alhejali, A. M., & Lucas, S. M. (2010, September). Evolving diverse Ms. Pac-Man playing agents using genetic programming. In *Computational Intelligence (UKCI), 2010 UK Workshop on* (pp. 1-6). IEEE.

[8] Ms. Pac-Man. http://en.wikipedia.org/wiki/Ms._Pac-Man,

[9] Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, *10*(1), 1-35.

[10] Ms. Pac-Man Simulator. https://github.com/kefik/MsPac-Man-vs-Ghosts-AI