A Hybrid Method of Dijkstra Algorithm and Evolutionary Neural Network for Optimal Ms. Pac-Man Agent

Keunhyun Oh
Department of Computer Science
Yonsei University
Seoul, Republic of Korea
ocworld@sclab.yonsei.ac.kr

Sung-Bae Cho
Department of Computer Science
Yonsei University
Seoul, Republic of Korea
sbcho@cs.yonsei.ac.kr

Abstract—Many researchers have interest on an auto-play game agent for Ms. Pac-Man, a classical real-time arcade game, using artificial intelligence. In order to control Ms. Pac-Man two ways are used. One is human-designed rules and the other is using evolutionary computation. Though well-defined rules, that use commonly search algorithms, guarantee stable high score, unpredicted situations can be happened because it is hard to consider every case. Evolutionary computation helps making a controller that covers uncertain circumstances that human do not think. These two methods can support each other. This paper proposes a hybrid method to design a controller to automatically play Ms. Pac-Man based on hand-coded rules and evolutionary computation. Rules are based on Dijkstra algorithms. In order to cover rules, evolutionary artificial neural networks are used. We have confirmed that the controller using the method makes higher performance than using each method separately by comparing with points of each other after playing game.

Keywords—hybrid approach; game agent; Ms. Pac-Man; Dijkstra algorithm; evolutionary neural networks

I. INTRODUCTION

Recently, with the development of video games, the interest on game AI is rapidly increased. Games are ideal test environments for artificial intelligence. In order to achieve goals, a player or a component should sequentially make decision and consider its long-term effects on complex environments where there is various and much information, random events happen unexpectedly, and the decision space is often huge. So, it is a challenging task to find a good strategy [1,2].

Many researchers investigate a game agent for Ms. Pac-Man, the real-time arcade game and the one of the most popular video games played in all over the world. The game is centered on navigating Ms. Pac-Man a player controls around a given map, accumulating points, and avoiding attacking ghosts. The game agent researched plays a role of controlling Ms. Pac-Man instead of a player. While it is relatively easy to understand how to play the game, it has complex aspects for the agent to get high score. Since it is the real-time game, a game agent reacts considering only current situation. In addition, Ghosts are non-deterministic. They make different decisions in the same situation. For these reasons, people have interest in developing better intelligent strategy and many competitions have been held [3,4].

Game agents for controlling Ms. Pac-Man are divided into two groups. One is based on human-defined rules and the other uses evolutionary computation. Each method has pros and cons. If a designer understands the game well, human-defined rules can make Ms. Pac-Man move to the best direction. Basically, Ms. Pac-Man tries to eat pills effectively until ghosts come closely to Ms. Pac-Man. In order to design rules, search algorithms are often used. Well-defined rules that reflect a lot of contexts can guarantee stable high points [5]. However, it is difficult to cover every situation because ghosts’ action in the game is unpredictable to people. Therefore, evolutionary computation is used. Evolutionary computation can provide solutions a person do not expect. Evolutionary artificial neural networks and evolved fuzzy systems are often proposed for problem solving in the game [6]. Though their decision covers uncertain environment, in order to get a high performed controller, it is very time-consuming.

This paper proposes a hybrid method for controlling Ms. Pac-Man using rules based on Dijkstra algorithm and evolutionary computation. Basically, well-defined rules decide to next direction that Ms. Pac-Man goes to. When they cannot cover exceptional circumstances, evolutionary artificial neural networks help problem solving. We prove that a controller using the method makes Ms. Pac-Man longer live and get higher score than using each method separately.

II. THE MS. PAC-MAN GAME

Ms. Pac-Man is a classic arcade video game released in North America in 1981 and reached immense success. It is one of the updated versions of Pac-Man, a predator-prey style game. The Human player maneuvers an agent to eat pills and to avoid three ghosts in the maze. The Ms. Pac-Man game is shown in Figure 1. Initially, he has three lives, and gets an extra life after reaching 10,000 points. If ghosts catch Ms. Pac-Man, she loses a life. Because their behavior patterns are non-deterministic unlike original Pac-Man game, the game is more difficult and more interesting. There are 220 pills. There are four power pills in the corners of the maze. After Pac-Man eats a power pill, Ghosts’ color changes blue and ghosts are edible for 15 seconds. Killed edible ghosts are reborn at center in the maze. If every pill and power pill is eaten, one level is ended and next level is start. Table I shows the worthy of each component.
Figure 1. A snapshot of the Ms. Pac-Man game

### TABLE I. THE WORTHY OF EACH COMPONENT

<table>
<thead>
<tr>
<th>Number</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pill</td>
<td>220</td>
</tr>
<tr>
<td>Power pill</td>
<td>4</td>
</tr>
<tr>
<td>Edible ghosts</td>
<td>4 Consecutively, 200, 400, 800, and 1600</td>
</tr>
</tbody>
</table>

III. PRVIOUS STUDIES

A. Hand-coded rule based approaches

Lucas proposes a tree search strategy for path finding to play Ms. Pac-Man. The approach is to expand a route-tree based on possible moves that the Ms. Pac-Man agent can take to depth 40 [5]. RAMP is a rule-based agent. It recorded one of the high scoring agents in WCCI 2008. RAMP architecture is implemented with layers for both conditions and actions. When conditions are sufficient, following actions are done [7]. Ice Pambush 2 is based on path costs. They used two variants of the A* algorithm to find the best path, the one with the lowest cost, between Ms. Pac-Man and the target location. The Manhattan distance is used in the search algorithm. At each iteration, one of the defined rules is fired to control her [8]. Wirth applies influence maps to the task of creating an artificially intelligent agent for playing Ms. Pac-Man. The model is relatively simple and intuitive and has relatively few user parameters that require tuning [9]. Though these hand-coded rule-based systems can produce the high scoring controllers, It is difficult to made rules considering every situations.

B. Evolutionary computation based approaches

Generic algorithms help a designer get Ms. Pac-Man controllers dealing with novel circumstances. Szita and Lorincz proposed a simple rule-based policy. The purpose of rules is organized into action modules and a decision about which direction to move is made based on priorities assigned to the modules in the agent. They are reinforcement learning method using evolutionary computation to enhance rules’ performance [1]. Lucas shows a method using evolved neural networks. The single layer perceptron he used then consists of just one such layer [3]. Gallagher and Ryan proposed a method using a simple finite state machine and rule sets, with parameters that specify the state transition and the probabilities of movement according to each rule. These parameters are learned by evolutionary computation [10].

IV. THE PROPOSED METHOD

A. The game agent

Common game agents are composed of sensing, thinking, and acting. Figure 2 shows the proposed Ms. Pac-Man game agent. The sensing module catches the information on the game such as the locations of ghosts and Ms. Pac-Man. The situation of the game is input to the agent by screen capture of game’s user interface. The pixel extractor finds the color of each pixel. The feature extractor enables the agent to understand coordinates of each component such as power pills and ghosts based on pixel color. Directions of movement of ghosts, Ms. Pac-Man and relative directions of each other, and other information on the game are the information extractor. By the thinking module, the agent determines the way to go. After thinking, the agent checks that a selected direction is available through action validation. If invalid, the agent senses a new situation and considers the way, again. Finally, the agent controls the game using keyboard hooking and makes Ms. Pac-Man move to the selected direction.

This paper focuses on thinking. First of all, simple rules are used to escape danger situation that has very high probability to be caught by ghosts. Secondly, rules based on Dijkstra algorithms help the agent detect safe direction to go. If these rules cannot cover the circumstance, the way is selected by evolved neural networks.

B. The hybrid method

Well-designed rules by human expert guarantee stable and high scores. However, it is impossible to consider every circumstance because Ms. Pac-Man game is complex and non-deterministic. Though a controller that is an evolved
neural network enables her to respond to all situations, it is very time-consuming and a difficult problem to get a high performed controller due to characteristics of evolutionary computation.

This paper proposes a hybrid approach to determine direction of Ms. Pac-Man by using human-designed rules and evolved neural networks. Figure 3 shows the flow chart of the way. In Dijkstra based way, threshold of edge’s weight is defined to survive. If every path’s cost is over the threshold, direction is not selected by rules and the controller thinks through an evolved neural network. It is based on idea in the open software kit (http://mspacmanai.codeplex.com/). Her movement is important at a moment time because it has an effect on overall game play. The neural network makes her more safety.

Figure 3. Flow chart of the proposed hybrid method

C. Danger escape rules

If Ms. Pac-Man is on a dangerous environment, danger escape rules play a role to survive Ms. Pac-Man as possible as fast. Danger is defined as the probability of ghost’s catching Ms. Pac-Man. If ghosts are near her within 4 nodes and its direction is the opposite of direction of her or makes them meet, Ms. Pac-Man needs to turn her direction. However, if a power pill is closer than a half of the number of nodes of ghosts, she goes to the direction of the power pill. The number of nodes is defined by an agent designer.

D. Dijkstra algorithm based rules

Dijkstra’s algorithm is conceived by Edsger Dijkstra. It is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs(weight), producing a shortest path tree. This algorithm is often used in routing. The algorithm finds the path of minimum total length between two given nodes P and Q. The fact is used in routing. The algorithm finds the path of minimum total length between two given nodes P and Q. The solution presented, the minimal paths from P to the other nodes are constructed in order of increasing length until Q is reached [11]. The pseudo code of the algorithm is shown Figure 4. An upper bound of the running time of the algorithm is defined as equation (1) where \(dk_Q\) and \(em_Q\) are times needed to perform decrease key and extract minimum operations and \(|E|\) and \(|V|\) represent edges E and nodes V in set Q, respectively.

\[
\text{TimeCost}(Q) = O(|E| \cdot dk_Q + |V| \cdot em_Q)
\]

First of all, a graph is constructed from the Ms. Pac-Man game environment. The game agent makes the map divided into 28*31 nodes. These nodes map each node to the graph. Adjacent nodes are connected by edges. Secondly, weight is calculated by how much dangerous. The Source node is the node has Ms. Pac-Man. Basically, Costs of movement to a node is calculated by equation (2). It means Euclidian distance between the node and ghosts. Additionally, direction of ghosts, power pills, and ghosts, whether ghosts are edible or not, and how much remaining edible ghosts’ flee time of each node are considered. It is similar to Danger escape rules. The distance of edible ghosts and power pills influences reducing weights. Finally, Dijkstra algorithm determines direction of her. The node on that furthest pill is from her becomes the destination node.

\[
\text{Cost}_{\text{src},\text{target}} = \sum_{i} \sqrt{(X_{\text{target}} - X_i)^2 + (Y_{\text{target}} - Y_i)^2}
\]

E. Evolutionary neural networks

Evolutionary computation searches through the space of behaviors for neural networks that performs well at a given task. This approach can solve complex control problems and it is effective in problems with continuous and high-dimensional state space instead of statistical techniques that attempt to estimate the utility of particular actions [12].

Especially, in this paper, the NEAT method proposed by Kenneth O. Stanley is used to evolve networks to control Ms. Pac-Man. The method enables not only connections’ weight but also topologies of neural networks [13]. We define 20 input nodes and 4 output nodes. 20 input nodes are shown in Table II. ‘Distance’ means relative distance between Ms. Pac-Man and each game component. Also, relative directions that are ‘Up’, ‘Down’, ‘Right’, and ‘Left’ are included. If the nearest ghost is located in Ms. Pac-Man’s left side, ‘Left’ is
but others are 0. Output nodes are the ways for Ms. Pac-Man to go that are directions. The highest scored node is selected to move. Figure 5. shows the agent for evolving neural networks. The sensing module catches information on the game. In the thinking module, one of the neural networks made by NEAT decides for Ms. Pac-Man to go. The Acting module controls her. When games of that the number is defined are ended, each network is estimated. Fitness is measured by average score of games using it. After the end of one generation, populations are evolved by generic operators that are selection, crossover, and mutation. The procedure of evolution is shown in Figure 6.

### TABLE II. DEFINITION OF INPUT NODES FOR NEURAL NETWORKS

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>The nearest ghost,</td>
<td>Distance</td>
<td>Float</td>
</tr>
<tr>
<td>The nearest edible ghost,</td>
<td>Up</td>
<td>Bool</td>
</tr>
<tr>
<td>The nearest pill,</td>
<td>Down</td>
<td>Bool</td>
</tr>
<tr>
<td>The nearest power pill</td>
<td>Right</td>
<td>Bool</td>
</tr>
<tr>
<td>Ms. Pac-Man</td>
<td>Left</td>
<td>Bool</td>
</tr>
</tbody>
</table>

V. EXPERIMENT

We evaluated the proposed method comparing with human-designed rules and evolutionary computation. We measured the average scores of Ms. Pac-Man games by controlling each method 10 times for reliable evaluation. The agent recodes scores of each game. In addition, we observed how many defined rules and an evolved neural network has effects on the game in the proposed method.

A. Experimental settings

In this paper, we use a framework to control Ms. Pac-Man developed by Jonas Flensbank and Georgios yannakakis (http://mspacmanai.codeplex.com/). As already mentioned, rules are used in this paper based on controller of the software kit because they are one of the best performed rules to control Ms. Pac-Man. C in Equation (1) is defined as 40 and weight threshold is 2. Table III shows the parameters for evolving neural networks by the NEAT method. In one generation, 10 games per a population were played. Fitness of a population is the average of game scores. In order to reduce time, we conduct evolutionary computation on a simulator that can be modified game speed. We got the best performed gene after evolution and test a controller using the network. The hybrid method uses these rules and evolutionary neural networks.

### TABLE III. PARAMETERS FOR EVOLVING NEURAL NETWORKS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>100</td>
</tr>
<tr>
<td>Generation</td>
<td>1000</td>
</tr>
<tr>
<td>The mutation rate for connection’s weight</td>
<td>0.96</td>
</tr>
<tr>
<td>The mutation rate to add and delete nodes</td>
<td>0.2</td>
</tr>
<tr>
<td>The mutation rate to add and delete connections</td>
<td>0.2</td>
</tr>
<tr>
<td>Elitism proportion</td>
<td>0.1</td>
</tr>
<tr>
<td>Selection proportion</td>
<td>0.8</td>
</tr>
</tbody>
</table>

B. Evaluation

The average fitness of every step is shown in Figure 7. The evolved neural network that is best performed has 120 edges among nodes and 12 nodes on a hidden layer. Fitness of the best gene recorded 3053. It indicates that the evolved neural networks help Ms. Pac-Man go to relatively safe direction. Experimental results are shown in Figure 8. Comparing with an evolved neural network, human-designed rules get more points. Surely, it is possible to make more smart neural networks through designing networks and setting parameters well. However, these things are difficult and need to additional effort. In order to evolve neural networks we designed, one or more times are spent. Though
the structure and parameters were changed sometimes based on other research, they did not give us more scores. It implies that if human understands how to solve problem, human-designed rules are adapt to find a way.

The proposed hybrid method is much higher performed than other ways. Worst scores are lower than minimum of rules but the gap is tiny. However, best points outstand. It indicates that the hybrid approach can solve some problem that a designer does not predicate and designed rules cannot deal with. We verify how much the neural network and rules influence to determine Ms. Pac-Man’s direction. Table IV shows the average of the number of decisions that each method in the hybrid method and percent of selections. Though the evolved neural network seldom determines her direction, it makes Ms. Pac-Man longer lives. The fact means that one decision influences the overall game. We verify that the hybrid approach is a better controller than each method alone.

<table>
<thead>
<tr>
<th>Method</th>
<th># of decisions</th>
<th>% of decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Dijkstra-based rules</td>
<td>71504.6</td>
<td>0.99860343</td>
</tr>
<tr>
<td>The evolved neural network</td>
<td>100.5</td>
<td>0.00139657</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed the hybrid method to control Ms. Pac-Man using human-designed rules and the evolved neural network. Hand-coded rules can guarantee best choice in some situations but cannot cover every circumstance. The evolutionary computation makes her go to the safety location in an overall game. However, it is difficult to make a best decision. In the hybrid approach, firstly, the game agent considers her way through designed rules based on Dijkstra algorithm. If Dijkstra algorithm does not find a safe course, the evolved neural network based on NEAT is used to solve problem. We conducted experiments to verify the proposed method. The Ms. Pac-Man game was played by methods that are Rules, the evolved neural network, and the hybrid approach and scores of each game is compared with other methods. The hybrid approach got most scores in these things.

For future works, we are planning to cover two issues. One is improving a Dijkstra-based search algorithm and the other is combination to other methods. As a human expert understands Ms. Pac-Man game well, it is possible to consider more situations. Each machine learning algorithm has unique characteristics. In addition to evolutionary computation, they may help her get more scores in specific environment that they can solve well.

Acknowledgement. This work was supported by Mid-career Researcher Program through NRF grant funded by the MEST. (No. 2009-0083838)

REFERENCES