Applying Monte-Carlo Tree Search to Collaboratively Controlling of a Ghost Team in Ms Pac-Man

Kien Quang NGUYEN and Ruck THAWONMAS
Intelligent Computer Entertainment Laboratory, Ritsumeikan University

***Abstract*—We present an application of Monte-Carlo Tree Search (MCTS) to controlling ghosts in the game of Ms Pac-Man. We approach the problem by performing MCTS on each ghost's tree that represents the game state from the ghost's perspective. Our goal is to create a strong ghost team that is adaptable to a variety of Ms Pac-Man's play styles. This ghost team (ICE gUCT) won the CEC 2011 Ms Pac-Man vs Ghost Team Competition for the ghost side.**

I. Introduction

Ms Pac-Man is a digital video game whose rules are very simple, but the game requires quite complex strategies for a successful game play. Because of that, it becomes a good testbed for AI research. Although there were a number of existing work in automatic control of Ms Pac-Man, there has been very limited work done in controlling of the four ghosts in the game [1]–[3]. A reason for this is that the original game does not allow the player to control the ghosts. Therefore, researchers in ghost controlling adopted a variety of game simulators. Because such simulators are different, both functionally and cosmetically, in how they resemble the original game, it was hard to compare their controllers' performances.

The main problem in controlling the ghosts is how to make them behave collaboratively. Although normally moving with the same speed as Ms Pac-Man, each ghost cannot reverse its direction and in edible state can only move with half of its normal speed. Without an effective collaborative control, it becomes nearly impossible for them to effectively trap Ms Pac-Man.

In this paper, in order to effectively control the ghosts (in other words, lower the score gained by Ms Pac-Man as many as possible), we adopt the the UCT method [3], [4], a.k.a UCB method applied for Monte Carlo Tree (MCT). In order to ease performance comparison with other ghost controllers, we use the same simulator as in the Ms Pac-Man vs. Ghost Team Competition CEC'11 [5]. The UCT method is one of the tree-search techniques based on simulation of possible future moves. The method performs random simulation more often for certain promising moves whose nodes have higher average rewards and are less often visited, attempting to keep the balance between exploitation and exploration in the search.

II. Outline of the proposed method

A. C-Road

For every maze of the game, we define a C-road as a corridor that connects two cross-points. The C-road is important because when a ghost enters a C-Road it cannot turn back according to the game rule. Thus, we can only control which direction a ghost should go at a cross-point.

B. Proposed method

To make a ghost team adaptable to any type of Ms Pac-Man controllers, we need to pay attention to the following two issues:

- having a rough image of how Ms Pac-Man moves, in other words, the ability to predict the movement of Ms Pac-Man, and
- controlling the ghosts to trap or catch Ms Pac-Man effectively.

To cope with these two issues, we propose a system that is a combination of MCTS and rules. In this system, three ghosts – Pinky, Sue, Inky – are controlled by the MCTS algorithm while Blinky moves according to a very simple rule that always moves it towards the cross-point that Ms Pac-Man is heading to by the shortest distance. Although, in general, the MCTS algorithm uses random simulation (during simulation, all ghosts and Ms Pac-Man will move randomly), having a rough knowledge about Ms Pac-Man's movement patterns will considerably reduce the search space and hence reduce the time needed for the MCTS ghosts to come up with an actually good strategy. As a result, we use the k-nearest-neighbor (KNN) algorithm for predicting Ms Pac-Man's movements as described in Sec. IV-C. Moreover, Ms Pac-Man is a real-time game, requiring each ghost to return its next direction within a limited time. By having a ghost, Blinky in our case, move according to some rules, we can reduce the work load to the CPU and to some extent can guide searching on the search space to find an optimal move much faster.

III. Controlling of ghosts

A. Game Tree - Monte Carlo Tree Search

For each MCTS ghost, we construct a tree (Monte-Carlo Tree) that represents the game state from the ghost's perspective. We consider each cross-point a node in the tree; therefore, a tree branch represents a C-road. The root node is the cross-point that a ghost of interest is going to visit next in the maze, and its child nodes are all adjacent cross-points that the ghost can reach while coming from the parent node. Because the ghost cannot turn back to the previous node it just visited, obviously, a node cannot have the direct parent node as one of its direct child nodes. In addition, when a global reverse occurs (all ghosts are forced to reverse their directions by the game system), we start constructing of a new tree with a new root node.

The initial MCT of each MCTS ghost only contains the root node. From this node, we construct its tree as follows:

1) Select a satisfiable road (a road containing nodes selected according to formula (1)) from the root node.
2) Expand the leaf node of the selected road and randomly select one of its child nodes.
3) Start the Monte-Carlo simulation where each of the other two MCTS ghosts moves according to the selected road in its tree while Blinky and Ms Pac-Man move according to rules described in Secs. III-B1 and III-B2, respectively.
4) Reward each node on the selected road based on the result from the Monte-Carlo simulation (III-C).
5) Repeat from step 1 to step 4 as many times as possible. When the MCTS ghost has actually reached its next cross-point (the root node in its tree), stop its tree construction and select its next direction to move as the one that leads to the child node with the highest amount of rewards as the first criterion, and with the highest number of revisits as the second criterion.

During tree construction, the UCT is applied. While going down the tree of a given MCTS ghost, we choose the next child node as the one that holds the largest UCB1 value. The UCB1 value of node i is denoted as:

$$UCB1(i) = \frac{X_i}{T_i} + C\sqrt{\frac{\ln T}{T_i + \epsilon}} \tag{1}$$

In this formula, C is a pre-defined constant, T_i and T denote the number of times node i and its parent node have been visited and ϵ is a small enough number ($\epsilon \approx 10^{-6}$). Note that because the parent node is always visited before its child nodes, the value of T is always greater than zero. Also X_i here denotes the accumulated reward of node i.

B. Simulation

First, we calculate the game state, i.e., the state of all objects in game, when an MCTS ghost of interest will have arrived at the root node. Then, we start the simulation from the root node of the MCTS ghost's MCT. We move this MCTS ghost according to its tree. When we arrive at a leaf node, the MCTS ghost will randomly determine its next direction at a cross-point until the simulation stops. This simulation is using the same mechanism as that of the game simulator used in Ms Pac-Man vs Ghost Team Competition 2011. However, in our simulation, Ms Pac-Man and the other ghosts move according to rules described in the next section. And the simulation will stop if one of the following conditions is met.

- Ms Pac-Man is eaten.
- All pellets and power pills are eaten.
- A certain amount of game cycles has passed.

1) Ghosts' moving rules: In the simulation, all of the other MCTS ghosts move according to their trees, whose construction has been stopped, currently in use in the game. And because we limit the maximum number of times the root node should be revisited (say, 1000 times), the depth of each tree is also limited. After having reached leaf nodes, these MCTS ghosts will determine their next directions randomly. The MCTS ghost of interest, whose tree is under construction, also determine it next direction in a similar fashion, first moving according to its tree and then performing random movements. Because of this mechanism, each MCTS ghost has partial knowledge about how their friends will move in the future to some extent, which helps the MCTS ghosts collaboratively work with each other. Although Blinky always moves to the cross-point in front of Ms Pac-Man, in the simulation, we replace its movement style with a randomly moving one. This is heuristically done so as to make the other MCTS ghosts behave more aggressively.

2) Ms Pac-Man's moving rules: Ms Pac-Man in the simulation moves according to the following rules:

- Determine Ms Pac-Man's next direction at a corner or a cross-point.
- As an exception rule, immediately reverse Ms Pac-Man' direction if there is an inedible ghost ahead in the same C-road within a certain amount of steps (i.e., pixels in the maze).
- Fix Ms Pac-Man' direction if there is a power pill ahead within a certain amount of steps.

At first, we use the Ms Pac-Man's movement predictor in Sec. IV for determining Ms Pac-Man's next directions, but we also move it randomly under a certain probability. After a while, Ms Pac-Man's movements will become fully random. The reason behind our policy is because we cannot completely trust the predictor. Thus, some random movements compensate such prediction errors.

C. Reward

All nodes that are visited along an MCTS ghost's path will be rewarded according to the result from the simulation. We use two kinds of criteria to decide the reward. The first one is the reverse of scores earned by Ms Pac-Man, and the second one is the reverse of time taken to finish that simulation. We also take into account penalties to ensure that all ghosts stay dispersedly with a safe distance from Ms Pac-Man.

IV. Ms Pac-Man's movement prediction

A. Considered aspects

To predict the movement of Ms Pac-Man, there are some aspects we need to consider. When a human player controls Ms Pac-Man, he or she would pay attention to the distance from Ms Pac-Man to the nearest ghost, the distance to the nearest power pill and so on. Accordingly, to estimate Ms Pac-Man's movement, we focus on the following features:

- The status of the nearest and the second nearest ghosts.

- The distances from Ms Pac-Man to the nearest ghost and the second nearest ghost
- The distance to the nearest power pill
- The distance to the nearest pellet
- The distance to the nearest cross-road
- The status of the nearest power pill's nearest ghost
- The distance from Ms Pac-Man to the nearest power pill's nearest ghost
- The distance from Ms Pac-Man's nearest power pill to the nearest power pill's nearest ghost

To some degree, we can see these measurements as a representation of the current game state. With these measurements, to estimate and predict Ms Pac-Man movements, we use the k-Nearest-Neighbor algorithm for finding the previous game states that mostly resemble the current game state and from them try to find the direction which Ms Pac-Man would go to.

B. Space formation

A set of the above features forms a vector in multidimensional feature space that we call input space. Note that the input vector space only expresses the current state of the game, it cannot express the movement of Ms. Pac-Man. To express this, we introduce another space – the move space that shows the meaning of these moves (e.g., a move to be away from or move to be close to an object). From this, we perform mapping of a representative game state in the input space to a move Ms. Pac-Man should do, given that game state, in the move space. To some extent, this mapping tells us how Ms Pac-Man would move in near future (Fig. 1).

C. Ms. Pac-Man's move prediction process

We predict Ms. Pac-Man next moves or directions according to the following steps (Fig. 1):

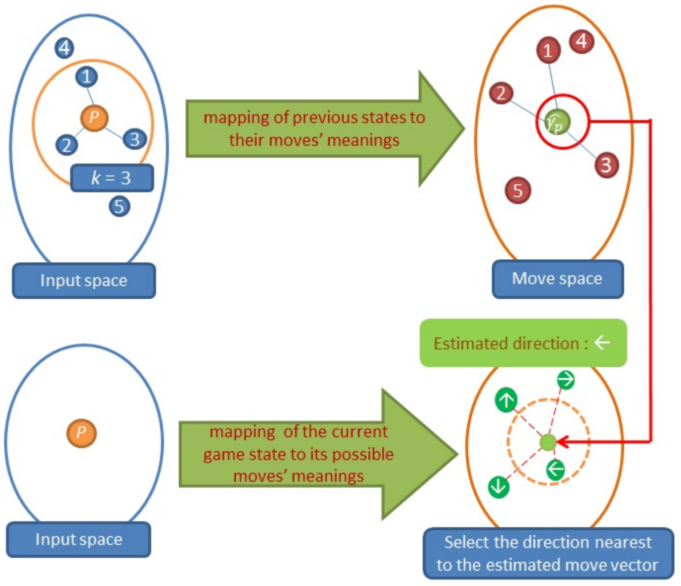
- Collect Ms. Pac-Man's moving data (a series of input vectors, each corresponding to a game state).
- From the collected data, choose the k nearest input vectors to the current input vector p (we set k to 3).
- From these k input vectors, estimate the move vector $\widehat{\gamma_p}$ in the move space as follows:

$$\widehat{\gamma_p} = \frac{\sum_{i=1}^{k} \frac{\gamma_i}{d(p,i)}}{\sum_{i=1}^{k} \frac{1}{d(p,i)}},$$

  where γ_i is vector i in the move space, $d(p,i)$ is the normalized distance between p and its i^{th} nearest neighbor in the input space.
- Determine the set of possible moves Γ_p, given the current input vector, in the move space as follows:

$$\Gamma_p = \{\gamma_{p,dir} | dir \in \{\uparrow, \rightarrow, \downarrow, \leftarrow\}\}$$

  where $\gamma_{p,dir}$ is the move vector corresponding to vector p with the attempt to move to direction dir.



mapping of previous states to their moves' meanings
k = 3
Input space
Move space
Estimated direction : ←
mapping of the current game state to its possible moves' meanings
Input space
Select the direction nearest to the estimated move vector

Fig. 1. Prediction of Ms Pac-Man movement by mapping the location of Ms Pac-Man p in the input space, using its k-nearest neighbors, to the location $\widehat{\gamma_p}$ in the move space and then finding the nearest direction – from up, down, left, and right – to $\widehat{\gamma_p}$.

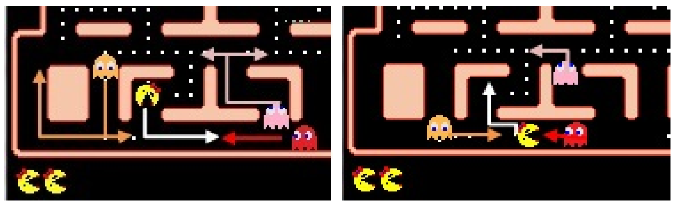


Fig. 2. Example of how to use the predicted direction of Ms. Pac-Man for controlling ghost to make a pincer attack. The arrows show the path Ms. Pac-Man and ghosts should take according to the predictor and ghost MCTs.

- From the resulting $\widehat{\gamma_p}$, determine the nearest vector in Γ_p – the most resembles the estimated move's meaning.
- Consider the direction associated with that vector as the next Ms. Pac-Man's direction.

If the data used for predicting Ms Pac-Man's movements are too large, this prediction process will take too much time and do not fit for this kind of real-time application. On the other hand, if the training data are too small, there is a high chance that this prediction will be wrong. However, with a suitable number of training data, we may be able to correctly estimate Ms Pac-Man's local movements (Fig. 2), which in turn may contribute to good guessing of the movement of Ms Pac-Man whose strategy changes from time to time.

V. Results and Conclusion

Our proposed ghost team ICE gUCT recently won the competition in the aforementioned competition in CEC 2011 [5]. Table I shows a part of the result of ICE gUCT. Here, we only reported the scores of the best three ghost teams versus best three Ms Pac-Man teams [5]. Note that the stronger a ghost team is, the less score it will lose to an opponent Ms Pac-Man team. Comparing ICE gUCT with

TABLE I
THE CEC 2011 MS PAC-MAN VS GHOSTS SCORES.

	ICE gUCT	BruteForce	emgallar
James	16436	24158	21805
emgallar	16208	22599	17938
MsAriadne	19282	19031	20076
Average	17308.7	21939.3	19939.7

the rule-based ghost team (2nd place: BruteForce) and the Ant-Colony ghost team (3rd place: emgallar), one can readily see that using MCTS with the ability to look ahead and predict Ms Pac-Man's movement is an effective way to create a strong ghost team adaptable to a variety of Ms Pac-Man play styles. Also, we believe that our approach has great potential in developing generic AI agents.

REFERENCES

[1] P. Rohlfshagen and S. Lucas, "Ms Pac-Man Versus Ghost Team CEC 2011 Competition," *Proc. IEEE Congress on Evolutionary Computation (2011)*, pp. 70-77, Jun. 2011.

[2] B.K.B. Tong and C.W. Sung, "A Monte-Carlo approach for ghost avoidance in the Ms. Pac-Man game," *Proc. IEEE GIC*, Hong Kong, pp. 1-8, Dec. 2010.

[3] N. Ikehata and T. Ito, "Monte Carlo Tree Search in Ms Pac-Man," *in The 15th Game Programming Workshop, IPSJ Symposium Series*, Vol. 2010/12, 2010. (in Japanese)

[4] S. Samothrakis, D. Robles, and S. Lucas. "Fast approximate max-n Monte-Carlo Tree Search for Ms Pac-Man," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 3, No. 2, pp. 142-154, Jun. 2011.

[5] The Ms. Pac-Man vs. Ghost Team Competition CEC11 Result Page: www.pacman-vs-ghosts.net/cec11results/