Monte-Carlo Tree Search in Ms. Pac-Man

Nozomu Ikehata and Takeshi Ito

Abstract—**This paper proposes a method for solving the problem of avoiding pincer moves of the ghosts in the game of Ms. Pac-Man to enhance the chance of survival for Ms. Pac-Man. To achieve this, the approach of using the UCT (UCB applied to trees) algorithm, a Monte-Carlo tree search algorithm that proved to be successful with a computer game of Go, was employed. The UCT algorithm is one of the tree search algorithms based on simulations and performs many searches on potential moves weighing the mean reward based on random simulation and the number of searches for a node. Although it is difficult to build a complete game tree that considers all situations in a real-time game like Ms. Pac-Man, because of the time limitation and the extent of implicit information, a method was developed to predict the probability of being trapped from pincer moves and avoid such situations at a feasible cost by repeating simulations on similar discretized situations. As a result of a performance comparison between the proposed system and existing programs, significant improvement in the performance of proposed system over existing programs was observed in terms of its ability to survive, implying the effectiveness of proposed method.**

I. Introduction

Ms. Pac-Man has become a popular subject of study worldwide because of being selected as a subject of AI Competitions. The rules for the game are relatively simple for a digital game, while thorough, complex, and intelligent strategies are required in order to perform superb gameplay. Ms. Pac-Man is often compared with traditional Pac-Man. Although the rules of Ms. Pac-man are almost the same as these of traditional Pac-man, the moving rules of ghosts have a decisive difference. Although there is regularity in the moving rules of ghosts in Pac-Man, the moving of ghosts in Ms. Pac-Man is seemed at random, and the rules of the moving are not revealed. In addition, there is much information which is not released. For example, exact values, such as the degree of acceleration or slowdown are not revealed when the characters turn at the corner and at the time of dot-eating.

This characteristic makes it an excellent test bed for the advancement of AI technology. The ultimate aim of the AI Competition featuring Ms. Pac-Man is to beat the highest score that an expert achieved by running an automatic operation, making programs compete against one another to obtain the highest score [1]. In order to equalize the playing environment for programs and humans, programs were prohibited to utilize internal information in the game, and instead had to make decisions and create contextual awareness through the images on the screen. This constitutes a significant constraint for computer programs. This is because a computer game like Ms. Pac-Man tends to contain a lot of information (such as unique gravitational acceleration and slipperiness of the floor) that is not displayed on the screen, but a human player can use his/her senses and obtain those information from playing the game. It is extremely difficult for a program to obtain exact values for such properties real-time. Having many unrevealed rules like these is one of the characteristics of computer games and this is one of the reasons why it is difficult to create a simple model for computer games using AI technologies.

Looking back in the history of AI Competition of Ms. Pac-Man, one would notice that a program employing a rule-based algorithm has won every competition so far, although a variety of AI techniques have been applied to the autonomous control system of Ms. Pac-Man. It is considered that this result shows that the rule-based systems, which have the advantages of having high stability and speedy decision-making, match well with the characteristics of the game elements of Ms. Pac-Man. On the other hand, the score of rule-based autonomous control systems for Ms. Pac-Man has peaked out during the past several years, showing signs that approach relying on rules are reaching their limits. One of the issues for rule-based systems is their poor capability to adapt to unknown situations. In other words, rule-based systems are excellent in situations that can be described clearly using conditional statements but are powerless in situations that are difficult or impossible to describe using conditional statements.

One such example in the game of Ms. Pac-Man is the problem of avoiding pincer moves of the ghosts, of which the conditions are very difficult to describe. Here, pincer moves refer to a situation where multiple numbers of ghosts block all possible paths for Ms. Pac-Man, eradicating all escape paths for her. This is a situation equivalent to a checkmate in Chess, and a contact with a ghost is inevitable no matter what kind of command is given. In order to ensure a high rate of survival for Ms. Pac-Man, contacts with ghosts should be avoided by all means. To predict and avoid pincer moves, future developments need to be read and evaluated perfectly. However, it is difficult to apply a game tree search to Ms. Pack-Man like the ones utilized in Chess and Go − games with perfect information− because the players have to rely on a lot of unrevealed information in the game.

Some approaches were performed by using tree search. One of the approaches was the work by Robles et al.[2]..Robles et al. tried to evaluate the danger of all the courses which can move from a Ms. Pac-Man's present coordinates by game tree search adopted widely by static thinking games, such as Shogi and Chess. They made the node what divided the maze by the grid of certain size. And they built the game tree of the depth 40 which uses a Ms. Pac-Man's move course as a branch by making a Ms. Pac-Man's current position into a route. Then, if it investigated whether a ghost would be included in the built game tree and the ghost was included in the game tree, it calculated whether a Ms. Pac-Man could reach cross point

The authors are with the Department of Computer Science, The University of Electro-Communications, Tokyo, Japan (e-mail: ito@cs.uec.ac.jp).

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

ahead of a ghost and it was evaluated whether the route would be really safe. However, since the depth of search was fixed, there were some insecure elements. For example, the movements of ghosts which is not included in a game tree are not taken into consideration at all and the precision of an algorithm is dependent on the reliability of calculation of cross point attainment.

This study aims to solve the problem of avoiding pincer moves in the game of Ms. Pac-Man and enhance the chance of survival for Ms. Pac-Man. The UCT (Upper Confidence bound applied to Trees) algorithm, a Monte-Carlo tree search algorithm proved to be successful in a computer game of Go, was employed in approaching the problem. The UCT algorithm is a tree search method based on simulations and performs many searches on potential moves weighing the mean reward based on random simulation and the number of searches for a node. Although it is difficult to build a complete game tree accounting for all situations in a real-time game like Ms. Pac-Man due to the extent of implicit information, it is considered that performing a number of simulations on similar and simplified situations would compensate for it.

While the problem has not been solved in rule-based systems, avoiding pincer moves of the ghosts is an important problem to be resolved in order to earn high scores in Ms. Pac-Man. As mentioned above, being trapped in a pincer move means that all possible paths for Ms. Pac-Man are blocked, leaving Ms. Pac-Man with no escape route. Avoiding such situations would enhance the chance of survival for Ms. Pac-Man dramatically.

With Ms. Pac-Man, which presents a lot of unrevealed information, it is practically impossible to build a complete game tree and, making it very difficult to avoid pincer moves that require predicting future situations. However, even if it is difficult to avoid pincer moves completely, the probability of getting trapped may be reduced by avoiding situations where the probability of getting trapped is high by calculating the risk of getting trapped. Hence, this study attempts to increase the chance of survival by avoiding situations where there is a high probability of getting trapped, rather than aiming at avoiding pincer moves completely. In order to attain this purpose, we tried to apply the Monte Carlo approach which succeeded in guessing a difficult evaluation value in the game of Go to Ms. Pac-Man.

II. UCT

By the mid 2000s, the performance of shogi and chess computer programs had reached a level comparable to that of human players. On the other hand, the performance of computer Go was not even as good as an amateur player. Behind this were the problems that, unlike shogi and chess, Go lacked the concept of "pieces" and each "stone" has exactly the same value as other pieces, making it difficult to define the feature elements for the assessment of situations.

Another problem was the fact that the search space was vast, in comparison with shogi and chess, making it difficult to develop Go programs utilizing static evaluation functions and game tree search methods. Nevertheless, in around 2005, reports on the success of a computer Go program adopting the Monte-Carlo method started to emerge. The term Monte-Carlo method refers collectively to methods that perform simulations using random numbers, which makes it possible to obtain solutions by approximation for problems that cannot be solved analytically by repeating a sufficient number of simulations. The basic flow of the primitive Monte-Carlo method applied to the game of Go is as follows.

1. Place a stone by picking a move from all available moves using a random number.
2. Repeat the step 1 for White and Black players alternatively. When there are no places to move the stone and a player passes his turn twice in a row, the game ends.
3. Assign points at the end of the game (typically, giving "1" point for winning and "0" points for losing).
4. The average score is determined by repeating the steps 1 through 3 several times.

This is a method that selects a move that scored the highest point after repeating a large number of simulations from the position at the time until the end of the game. Since the game of Go has the characteristic that legal moves are decreased to the end of the game, it is possible to complete playout against a certain number of legal moves in most cases. Because of this characteristic, Go is extremely compatible with simulation -based methods.

Furthermore, although one of the bottlenecks for the performance of computer Go programs is the difficulty to create evaluation functions, the Monte-Carlo method does not require evaluation functions to assess possible moves. However, the primitive Monte-Carlo method has a flaw in that there is no guarantee to return the best move for trees with the depth of two levels or higher. For example, take a situation where it is advantageous if the opponent makes a bad move, but disadvantageous if the opponent responds by making the correct move. If the number of "right" moves for the opponent was small, the probability that a "right" move is chosen during the simulation would consequently be small. In a real game, however, there is a great possibility that an opponent will choose the "right" depending on the ability of the player. Since the primitive Monte-Carlo method cannot account for the ability of the opposing player, there was a risk of the program making an aggressive move expecting the opponent to make a mistake.

In order to solve this problem, Remi *et al.* proposed the Monte-Carlo tree search method, in which the Monte-Carlo method was expanded to a tree search method [3]. There were two changes made to the primitive Monte-Carlo method. One was to assign a greater number of simulations to an advantageous move and another was to make the tree grow when the number of simulations exceeded a threshold. Although this method seemed to solve the problems of the primitive Monte-Carlo method, there was an obstacle of how to allocate a number of simulations to each available move. To begin with, determining an advantageous move itself was a demanding task in computer Go programs, for which the development of static evaluation functions was difficult. A revolutionary solution to this problem came in the form of UCT (UCB applied to Trees), formulated by Kocis *et al.* in the same year [4].

The UCT method expanded the UCB1 method, which is effective in solving the multi-armed bandit problem, to a game tree search format. At each node of a game tree, a child node

with the maximum UCB (Upper Confidence Bound) value, as obtained by equation (1), is calculated.

$$X_i + C\sqrt{\frac{\ln T}{T_i}} \quad (1)$$

Here, *Xi* is the mean reward of a child node i, *Ti* is the number of searches performed for the same child node *i*, *T* is the number of searches performed for the parent node, and *C* is the balance parameter. Hence, more simulations will be allocated for candidates with potential and candidates in which a search has not been performed. Since it is possible to use the result of the game as an evaluation value in the UCT method, no evaluation function is required. For this reason, designing a strategic model or feature elements is not necessary and the method works effectively for a new game or a game that has not been researched for a long time, for which little knowledge has been accumulated.

The emergence of UCT greatly enhanced the performance of computer Go programs that were struggling to make advances in improving their performance. The UCT method was further adopted in many other static strategy games as a result of its success in computer Go. The method proved to be more effective than previous methods in most of the games, exhibiting its versatility in different games. On the other hand, the application of UCT in real-time games has remained at a low level with only a few case studies being reported for the application in RTS (real-time strategy) games [5][6]. So far, there has been no study carried out on its application in real-time action games like Ms. Pac-Man.

III. Proposed Method

A. Subject of Assessment

We defined "C path" as a connection between two cross-points (T intersections and/or intersections) in a maze. A C path is a straight path in a maze that does not have any branching points in the middle. In Ms. Pac-Man, C paths hold extremely significant implications. The only operation allowed for a player playing Ms. Pac-Man is to operate Ms. Pac-Man and there are many instances in which the player ends up by making the wrong move. Depending on the C path Ms. Pac-Man chooses next at a T intersection or an intersection, Ms. Pac-Man may be trapped by multiple ghosts. Therefore, the player must be very careful in choosing the C path at any cross-point. Furthermore, the path that connects the current location of Ms. Pac-Man to a C path connected to a "cross-point 1 that is the closest to Ms. Pac-Man (if the current position of Ms. Pac-Man is at a cross-point, this position is not included)" and a "cross-point 2 which has a direct link to cross-point 1" is defined as a "path connected to Ms. Pac-Man." This path represents a short-term moving path which has a possibility of Ms. Pac-man passing through in the immediate future.

B. Game Tree

Fig. 1 depicts a game tree of Ms. Pac-Man used in this study.

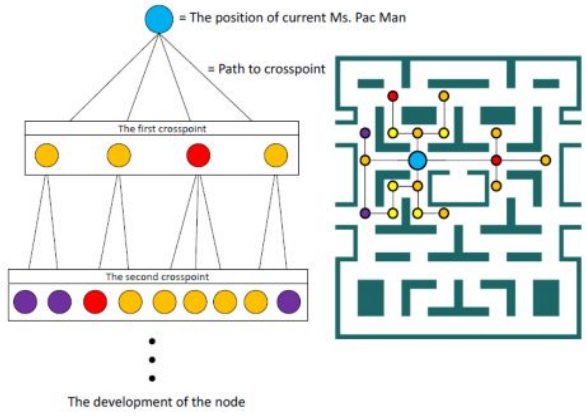

Fig. 1. A game tree for Ms. Pac-Man

Each node represents a cross-point while a branching point represents a C path. The root node is the current position of Ms. Pac-Man, a child node is the first cross-point which Ms. Pac-Man will reach, and a grand-child node is the second cross-point Ms. Pac-Man will reach. Descending a path from the root node to a terminal node means that Ms. Pac-Man selected one of the paths that connect to Ms. Pac-Man. When descending the tree, the movement path of Ms. Pac-Man is determined following a particular intermediate node, while the movements of ghosts are not restricted by the game tree and are chosen in a non-deterministic manner. The reason why the game tree does not account for ghosts' movement is because doing so will immensely increase the number of situations if the ghosts' movement path was included in the game tree.

Considering the cost of computation, the search will be limited to a shallow level. In other words, when movement is made from the root node to a terminal node passing through the same branches and nodes, all while the movement path of Ms. Pac-Man remains the same, the movement paths of the ghosts may be different. When descending down the game tree, the character moves in a discretized space, as described in the following section, rather than in an actual game space. When the visit count of a terminal node exceeds a predefined threshold, that terminal node will be expanded. The newly generated child nodes represent all cross-points reachable from the terminal node, excluding the parent node.

Once a game tree is built, the UCT algorithm is executed. The first process of UCT is to descend the game tree until encountering a terminal node. In the process of descending the game tree, a child node with the highest UCB value will be chosen at each node. The UCB value will be calculated using the following equation for node *i*.

$$UCB(i) = \begin{cases} n & (T_i = 0) \\ X_i + C\sqrt{\frac{\ln T}{T_i}} & (T_i > 0) \end{cases}$$

If T_i, is the visit count of a node is "0", or if a node is visited for the first time, a number *n*, which is sufficiently large, will be assigned to the UCB value. Otherwise, the UCB value will be determined by the UCB1 formula. The average number of survivals of Ms. Pac-Man, or the survival rate, shall be assigned to the mean reward X_i used in UCB1 formula

C. Simulation

When Ms. Pac-Man reached a terminal node without encountering a ghost in a tree search, a simulation starts. The objective of a simulation is to obtain the reward for a trial. During a simulation, Ms. Pac-Man and the ghosts will take turns in making non-deterministic moves in a discretized space.

1) Rules in the Discretized Space: During a simulation, Ms. Pac-Man and the ghosts will make their respective moves alternately under a unique set of rules that are different from actual game rules of Ms. Pac-Man. These rules are set for the purpose of simplifying the game play so that the computational cost for one update will be sufficiently small in executing the heuristic algorithm, while giving considerations to the movements of characters in the original Ms. Pac-Man game to an extent that does not undermine the advantages of discretizing the space. The rules applied in the discretized space are as follows. (P: Ms. Pac-Man, G: ghost(s))

(1) P and G will move straight ahead in a C path (do not travel back).
(2) P and G will determine the next course of movement only at cross-points.
(3) P will skip the update of next cycle every time it eats a certain number of pellets.
(4) P will update its state successively every time it turns a certain number of corners.
(5) When P eats a power pellet, G will turn blue and reverse its course.
(6) G in the blue state will skip its update of next cycle after every two moves.
(7) Once G has turned blue, its state will not be reset.
(8) G will not come back to life once it is eaten.
(9) Additional G will not come out of the ghost hut.
(10) There will be no bonus item.

Among these rules, the acts of "skipping the update of next cycle" and "updating its state successively" are meant for replicating the differences in the moving speed of characters in the discretized space. For example, if the condition for (4) is satisfied, the normal update cycle of "a move of Ms. Pac-Man → a move of a ghost" will be modified to "a move of Ms. Pac-Man → a move of Ms. Pac-Man → a move of a ghost", causing the state of Ms. Pac-Man to be updated one more than the ghost, causing Ms. Pac-Man to move one grid position more.

2) The Movements of Ms. Pac-Man: During a simulation, each character will move in a non-deterministic manner based on a certain set of behavioral policies. The behavioral policy of Ms. Pac-Man is to eliminate an obviously dangerous C path when choosing a path at each cross-point to avoid an encounter with a ghost to prevent coming in contact with it, unless she is trapped from a pincer move of two or more ghosts. The following four rules are established to attain this goal.

(1) Ms. Pac-Man will not go back to the C path she was on when choosing a path at a cross-point.
(2) When choosing a path at a cross-point, if there is an active (not in the blue state) ghost on a C path Ms. Pac-Man can move to and if it is heading towards her, she eliminates that C path from the list of available paths.
(3) If a unique C path is not determined by applying rules (1) and (2), she will choose a C path from the remaining C paths at random.
(4) If there is an active ghost within a certain number of grids ahead of Ms. Pac-Man on a C path, Ms. Pac-Man will reverse her course at the spot.

3) The Movements of Ghosts: Aggressive characteristic is given to the ghosts so that they will attempt to get nearer to Ms. Pac-Man as the number of pellets in the maze decreases. In order to replicate this characteristic, a method has been contrived to move ghosts probabilistically closer to Ms. Pac-Man when choosing a path for a ghost to move in the discretized space. The following equation defines the probability P of a ghost approaching Ms. Pac-Man.

$$P(G_{type}) = A(G_{type})\frac{AP - RP}{AP}$$

In the above equation, AP represents the number of all pellets arranged in the maze and RP represents the number of pellets remaining in the maze. $A(G_{type})$ is the aggressiveness of the ghosts and takes on one of the values shown in TABLE I depending on the color of the ghost. At a cross-point, a ghost will choose a C path with the probability P so that it will get close to Ms. Pac-Man. There are two types of target points for ghosts according to the manner in which they move towards Ms. Pac-Man. One is "the nearest cross-point or the corner in the direction Ms. Pac-Man is heading." By setting this target point, a ghost will attempt to get near Ms. Pac-Man by appearing in front of Ms. Pac-Man. Another target point is "the nearest cross-point or the corner in the reverse direction from Ms. Pac-Man." If this target point is selected, a ghost will attempt to get close to Ms. Pac-Man by chasing her from behind. The manner of approaching is set differently for different ghosts to increase the chance of pincer move by two or more ghosts occurring in the discretized space.

TABLE I
AGGRESSIVENESS OF THE GHOSTS AND THEIR TARGET POINTS

Ghost Color	Aggressiveness	Target Points
Red	0.9	Ahead of Ms. Pac-Man
Orange	0.8	Behind Ms. Pac-Man
Pink	0.7	Ahead of Ms. Pac-Man
Light blue	0.6	Behind Ms. Pac-Man

4) Conditions for Terminating the Simulation: The simulation is terminated if one of the following conditions is satisfied.

- Ms. Pac-Man has gotten in contact with a ghost that is not in the blue state.
- All pellets in the stage are eaten.
- A certain number of cycles have been performed after starting the simulation.

In the UCT method adopted in *Go*, a simulation is performed until the end of the game (no more moves are available for both players). Since the number of moves it takes to play a game of *Go* is around 500 at most, a simulation would not take long even if it is performed to the end of the game. On the other hand, it is realistically impossible to continue the simulation for the game of Ms. Pac-Man until the end of the game because the

time required to end a game is immense. Therefore, it will be necessary to terminate the simulation after a certain number of cycles instead of performing it to the end.

5) Reward: The following rewards will be obtained based on the result of simulation.

- Survival (a binary number "0" or "1" to indicate if Ms. Pac-Man has survived)
- The number of pellets eaten (0 to 1)
- The number of ghosts eaten (0 to 1)

After a simulation, the number of pellets eaten will be divided by the number of pellets remaining at the start of simulation and the number of ghosts eaten will be divided by the number of ghosts active at the start of simulation. This operation normalizes all values of rewards to take on a value between 0 and 1.

D. Updating the Game Tree

If the simulation is terminated by satisfying the above condition, the information held by each node in the game tree will be updated based on the rewards obtained. Updates of nodes will propagate from terminal nodes to the root node in the same manner as those in a normal game tree search. Although it is common to update a parent node by assigning the average value of information held by its child nodes in the UCT method adopted in *Go*, the proposed method passes the information held by a child node that has the highest survival rate to the parent node. In other words, assuming that the survival rates were 30% and 40% respectively for child nodes A and B, the parent node will be given the information held by the child node B. The reason why such a communication method was adopted is because in a situation where there is only one safe path among three available C paths, averaging the survival rate will decimate the presence of a safe path, ultimately eliminate the chance of the path being selected. In the case of computer *Go*, changing the placement of one stone would not have that much influence on the chance of winning. However, the survival rate of Ms. Pac-Man will change drastically depending on the path chosen.

E. Evaluation of Paths

If the number of simulations exceeds a threshold, the UCT algorithm is terminated. At this moment, the terminal nodes (grand child nodes for the root node) on the path connected to Ms. Pac-Man in the game tree will be compared with one another to determine the best path to take. Note that the discretized space in this study is designed to limit the cause of death for Ms. Pac-Man mostly to pincer moves of the ghosts because of the rules and behavioral policies described in Section 4.3. For this reason, there is a relationship where "the survival rate in the simulation" is roughly to the same as "the probability of pincer moves not occurring in the simulation." In other words, if the survival rate of a path is 0.620, then the chance of pincer move occurring is considered to be 0.380. Hence the path obtained by weighing the survival rate is considered to be safer than others because the chance of being trapped is lower.

F. Tactics

In the case with the UCT method employed in computer *Go* programs, it is considered that maximizing the chance of winning is more effective than maximizing the size of territory, which is the condition for winning. Attempting to gain more territory when a player's territory is already bigger than the opponent's is considered to be a dangerous move, having the possibility of presenting an opportunity for the opponent to take an advantage of. For this reason, the index of "winning," which is more stable than the size of territory, is used. Although it is difficult to determine a player's superiority in intermediate phases in the game of *Go*, it is relatively easy to determine a win or loss at the end of a game.

In case of Ms. Pac-Man, one cannot simply consider a single index, like "win" in the game of *Go*. This is because Ms. Pac-Man is not a game in which the ultimate goal is to beat the opponent, and therefore lacks the absolute index equivalent to a "win" in *Go*. Although it seems reasonable to maximize the average value of the achieved scores, a higher score does not necessarily mean having the capability to make smart moves. Scoring a lot of points momentarily in the game does not mean anything if Ms. Pac-Man is caught by a ghost a moment later. Although the main goal of Ms. Pac-Man remains achieving a high score, it is necessary to consider the three sub-goals of "avoiding contacts with the ghosts", "eating all the pellets", and "eating as many ghosts as possible".

The approach used in this study introduces a concept of "tactics" that define what Ms. Pac-Man should consider and how she should behave depending on the current situation. Tactics are made to achieve the sub-goals and Ms. Pac-Man is made to behave accordingly to the tactics. During the "survival" mode, avoiding ghosts is the primary goal, and during the "feeding" mode, eating as many pellets as possible is the priority. In addition, there is the "pursuit" mode in which eating more ghosts is the priority.

Tactics change depending on the UCT evaluation value and the state of the ghosts. For example, if the current tactics is "survival" and the evaluation results of paths using UCT shows that all paths have survival rates above a certain threshold, Ms. Pac-Man is considered to be in a relatively safe situation. In this instance, the tactics switched to "feeding" mode to put a priority on eating as many pellets as possible. When Ms. Pac-Man eats a power pellet, making the ghosts turn blue, the tactics of "pursuit" can be adopted to eat as many ghosts as possible. As illustrated in these examples, different tactics are adopted depending on the situation of Ms. Pac-Man. The following are the conditions for switching the tactics.

- If the survival rate of the previous frame was above a threshold, then the tactics of "feeding" are adopted.
- If the survival rate of the previous frame was below a threshold, then the tactics of "survival" are adopted.
- If the ghosts are in the blue state and the closest ghost is within a certain distance, then the tactics of "pursuit" are adopted.
- If the ghosts are in the blue state and the closest ghost is beyond a certain distance, then the tactics of "feeding" are adopted.
- If all ghosts are active, then the tactics of "survival" are adopted.

In order to enhance the effectiveness of UCT evaluation, changing the behavior of Ms. Pac-Man when she comes into contact with a power pellet or a ghost during the simulation is considered, depending on the tactics. For example, it is

inefficient to eat another power pellet during the simulation while the power pellet eaten previously is still effective. Therefore, if Ms. Pac-Man encounters a power pellet in the "pursuit" mode, it is considered a failure similar to an encounter with a ghost. By using such a setting, the survival rate of a path that may lead to an excessive consumption of power pellets will drop and, ultimately, it will be eliminated from the selection.

TABLE II
SELECTION OF A MOVEMNET PATH

Current Tactics	Selection Method
Survival	Selects a path in which the survival rate is the highest among the paths connected to Ms. Pac-Man.
Feeding	Selects a path which contains more pellets that can be eaten from the ones connected to Ms. Pac-Man and have a survival rate at or higher than the threshold
Pursuit	Selects a path on which more ghosts can be eaten among the paths connected to Ms. Pac-Man with the threshold survival rate or higher

<Survival> Encounters with a power pellet and a ghost in the blue state are allowed
<Feeding> An encounter with a power pellet is allowed, an encounter with a ghost in the blue state is not allowed
<Pursuit> An encounter with a power pellet is not allowed, an encounter with a ghost in the blue state is allowed

G. *Selection of a Movement Path*

The UCT method ultimately selects a path connected to Ms. Pac-Man based on the tactics. With the primary goal of avoiding contact with the ghosts, it chooses a path that enables the execution of tactics among the safe paths available. The selection will be made as summarized in TABLE II.

IV. PERFORMANCE EVALUATION TEST

A. *Method*

In the experiment, the proposed system is set to autonomously control Ms. Pac-Man to verify its performance. The program is executed for a total of twenty times according to the rules of the Ms. Pac-Man Competition. In each execution, (1) the final score at the end of the game and (2) the number of stages reached at the end of the game were recorded. The experiment was carried out on a platform with a Core2Duo 3GHz CPU and 2GB of memory. For the implementation of the proposed system, the C++ language was used. The number of simulations was set to 300 and the maximum number of cycles of simulation was set to 30. In addition, the game of Ms. Pac-Man included in the package “Revenge of Arcade” from Microsoft Games was used.

B. *ICE pambush3*

In order to compare the performance of proposed system with existing programs, ICE Pambush3, the winning program at the Ms. Pac-Man Competition CIG2009, was selected as a competitor. ICE Pambush3 was developed by a team from the Intelligent Entertainment Laboratory overseen by Professor Ruck in Ritsumeikan University and holds the world record in autonomously controlled Ms. Pac-Man [7]. It is the highest performing program available today. ICE Pambush3 adopts a rule-based algorithm and controls Ms. Pac-Man using nine conditional statements that take the costs of paths into account.

C. *Comparison of Final Scores at the End of the Game*

TABLE III summarizes the final scores at the end of the game.

TABLE III
SCORES ACHIEVED

p	min	max	mean	s.d.
Proposed system	6840	37630	24926.50	9058.71
ICE Pambush3	8620	30660	20574.50	6213.21

TABLE III shows that the proposed system outperforms ICE Pambush3 in terms of the maximum score and mean score and indicates its general performance in scoring is better than that of ICE Pambush3. Especially, the highest score of 37630 is better than the score of any program that participated in the Pac-Man Competition in the past, meaning that it is in effect a new world record. On the other hand, from the fact that the minimum score and the standard deviation of the proposed system are inferior to those of ICE Pambush3, the proposed system is behind ICE Pambush3 in terms of stability. After a comprehensive comparison including other results, ICE Pambush3 is still ahead of the proposed system in terms of stability.

D. *Comparison of Number of Stages Reached at the End of the Game*

TABLE IV shows that the proposed system outperforms ICE Pambush3 in terms of the maximum and average number of stages reached and indicates the superiority of its ability in clearing stages. On the other hand, the standard deviation is larger for the proposed system, meaning it lacks stability compared to ICE Pambush3. In addition, the minimum number of stages reached is "1" for both programs. This means that there were cases where the program failed to clear the first stage for both ICE Pambush3 and the proposed system.

TABLE IV
SUMMARIZES THE NUMBER OF STAGES REACHED

p	min	max	mean	s.d.
Proposed system	1	7	4.4	1.497
ICE Pambush3	1	5	3.4	0.970

E. *Observations*

The fact that the proposed system exhibited superior ability to survive compared to ICE Pambush3, which is a rule-based program, indicates that it was effective in placing a priority on the survival rate, as well as on the rate of defeating ghosts, by avoiding pincer moves of the ghosts using the UCT method and changing tactics. It implies that the formation of strategies was possible even with a minimal amount of preliminary knowledge. This was a type of achievement that was seen in other games that adopted the UCT method in a successful manner, which implies that the adoption of UCT method in Ms. Pac-Man was effective as well. On the other hand, the proposed system turned out to be inferior to ICE Pambush3 in terms of stability in its performance. One possible reason for this is that

the proposed system did not attempt to eat pellets placed in dangerous locations under certain situations as a result of making survival a primary goal. This phenomenon was often observed in a later phase of a stage, where all four power pellets had been consumed and pellets remained in the corners and down long stretches, where it is easy to be trapped by multiple ghosts.

As the number of pellets in the maze decreases, the ghosts become more active and the chance of being caught up by a ghost increases. Because of this, the survival rate of Ms. Pac-Man drops drastically. Hence, Ms. Pac-Man's main focus will be to survive, rather than attempting to eat pellets remaining in dangerous locations. Since the nature of proposed system does not guarantee avoiding pincer moves completely, Ms. Pac-Man will eventually be trapped in by ghosts, resulting in her death, as she spends time aimlessly without being able to eat pellets. Since similar situations are often observed in the earlier stages of the game, if such a situation develops when not that much points have been scored, the game will be over without being able to achieve any score.

V. Improving the Efficiency of Eating Pellets

The reason why the problem described in the above observation had developed is because the proposed system does not really consider the order of eating pellets. The cause is believed to be the failure to consume pellets in dangerous areas in an early phase of a stage, when the ghosts are still not active. Therefore, an attempt was made to improve the efficiency of the order of eating pellets by considering the level of risks in different areas of the maze. At the end of a stage when the ghosts become more active, it will be difficult to consume pellets in the corners and down long stretches, where the survival rate is low. The problem could be solved by prioritizing and consuming pellets located in such dangerous areas early in the stage. First, a risk level map was developed to indicate the survival rate of Ms. Pac-Man in different areas of the maze.

(1) Placement of Ms. Pac-Man

Ms. Pac-Man is placed on one of the grids where a character is allowed to assume a position (other than walls and OB). Here, the orientation of Ms. Pac-Man is determined at random from the directions of movement available at the position.

(2) Placement of Ghosts

Four ghosts are placed in the maze. The reason behind placing all four ghosts is to calculate the survival rate at the latter phase of the stage under a dangerous situation. For the same reason, all four power pellets are eliminated from the maze. Ghosts are placed on grids located in an area of 20 grids in radius in Euclidean distance. Here, the orientations of ghosts are determined at random from the directions of movement available at each position.

(3) Execution of the UCT Algorithm

For the situation determined by (1) and (2), a UCT algorithm that performs 2,000 simulations is executed to calculate the survival rate of Ms. Pac-Man on the best path.

(4) Perform steps (1) to (3) for all maze and all grids

The danger level map has been generated as shown in Fig.2.

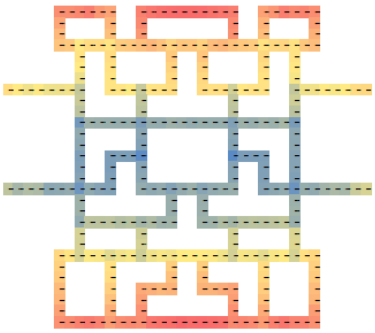

Fig. 2. Danger Level Map

In Fig. 2, a deeper shade of red indicates that the area is more dangerous. Replace the reward "the number of pellets consumed" with a new reward X that accounts for the level of danger in each grid a pellet is placed. In equation (1), $W_{i,j}$ represents the level of danger (1 - the survival rate of Ms. Pac-Man) at the grid (*i, j*) where the *k*-th pellet eaten was placed. Note that the reward is set to a sufficiently small value *n* for the paths where power pellets can be obtained during the simulation, because there is almost no danger at all and there is no need to take a risk to consume pellets.

$$X = \begin{cases} \sum_{k=1}^{n} W_{i,j} & (n_{ppill} = 0) \\ n & (n_{ppill} > 0) \end{cases}$$

VI. Performance Evaluation Test 2

The objective of this experiment is to verify the performance gain of the proposed system due to the optimization of the order of eating pellets. Therefore, the same experiment conditions as those of previous experiment were used.

A. Score at the End of the Game

TableV summarizes the final scores at the end of the game.

TABLE V
SCORES ACHIEVED

p	min	max	mean	s.d.
Proposed system (before improvement)	6840	37630	24926.50	9058.71
Proposed system (after improvement)	16800	58990	31105.60	11605.57

Table V shows a significant improvement in the score achieved compared to the system before the improvement was made. In particular, the highest score achieved with the improved system was 58990, which shows a drastic gain over the previous world record of the game of Ms. Pac-Man achieved before the improvements were made. In addition, there is also a significant improvement in the minimum score achieved compared to the system before the improvement. The number of unstable movements, in which the game ends with an extremely low score due to the failure to clear the first stage, has been reduced. On the other hand, the increase in the

standard deviation indicates that the range of scores achieved is still wide.

TABLE VI
SUMMARIZES THE NUMBER OF STAGES REACHED IN GAME

p	min	max	mean	s.d.
Proposed system (before improvement)	1	7	4.4	1.497
Proposed system (after improvement)	2	7	4.8	1.361

B. *Number of Stages Reached at the End of the Game*

From TableVI, it is observed that the average number of stages reached during the game has improved, indicating an improvement in the system's ability to clear stages. In addition, since there have been no instances of game ending without clearing the first stage, which was a situation encountered with the system before the improvement, and the standard deviation was reduced, it is considered that the system has become able to consume pellets in a more stable manner.

C. *Observations*

The performance of the proposed system has improved drastically by improving the efficiency of the order of consuming pellets. In earlier stages, when it is relatively safe with less number of active ghosts and all the power pellets aren't consumed, Ms. Pac-Man had been observed to make aggressive moves to eat pellets placed in dangerous locations, such as corners and down long stretches of the maze. This was a behavior that was not exhibited by ICE Pambush3 or the proposed system before the improvement. In addition, by classifying the paths with power pellets as the paths where the rate of obtaining pellets is low, the timing of consuming power pellets has been pushed back towards the latter half of the game, where the risk level is high. This translated to an increase in the average number of ghosts defeated with a single power pellet. This also increased the chance for Ms. Pac-Man escaping dangers by eating power pellets, thus increasing the survival rate.

However, while the system exhibited a significant performance gain over existing methods, the risk of being trapped in a pincer move by the ghosts could not be avoided completely. There were situations where the path determined to be safe by the system turned out to be a dangerous one in actuality. This proves that the accuracy of simulations performed is still imperfect for the real-world space and implies the limit of setting up rules manually for the discretized space. In particular, it is possible that inaccurate recognition of differences in the moving speed of Ms. Pac-Man and the ghosts is affecting the accuracy of predicting pincer moves. Ideally, it is preferable to make the system extract the rules of Ms. Pac-Man (implicit information such as the moving speed of characters and the extent of speed reduction after eating normal pellets) from captured screen automatically and set optimum parameter values against the rules. This may be achieved by using conventional machine learning techniques, such as reinforcement learning and neural networks. Obtaining environment (the world of game) automatically using image recognition is a significant challenge for the future study of artificial intelligence in computer games.

VII. CONCLUSIONS

This study attempted to solve the problem of avoiding pincer moves of the ghosts using the UCT method, a Monte-Carlo tree search algorithm that has been successful in computer games of *Go*. In a discretized simulation space representing situations in Ms. Pac-Man, which is a continuous game, a large number of simulations were performed with situations, whose probability of pincer moves occurring was intentionally set high, in order to calculate the probability of pincer moves occurring in the real-world situations. The survival rate of Ms. Pac-Man was improved by avoiding paths with high probability of pincer moves occurring. As a result, the proposed system was capable of scoring higher than all the other autonomous programs that participated in the Ms. Pac-Man Competition, establishing a new world record, greatly exceeding the previous record.

The future issues to be considered include the improvement of algorithm. The most important issue is, especially, improving the simulation accuracy. Although various parameters were set manually in this study, it is preferable to adjust them automatically by obtaining information from the real-world through real-time learning. It is considered that this approach would improve the performance of the proposed method significantly, leading to a more accurate grasp of the playing environment.

REFERECES

[1] Ms Pac-Man Competition, http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html

[2] D. Robles and S. Lucas, "A Simple Tree Search Method for Playing Ms. Pac-Man," *IEEE Symposium on Computational Intelligence and Games*, pp. 249-255, 2009.

[3] R. Coulom, "Efficient selectivity and backup operators in Monte-Calro tree search", *Proceedings of the 5th International Conference on Computer and Games*, 2006.

[4] L.Kocsis and C.Szepesvari,"Bandit based montecarlo planning", *European Conference on Machine Learning*, pp. 282-293, 2006.

[5] Michael Chung, Michael Buro, Jonathan Schaeffer, "Monte Carlo Planning in RTS Games", *IEEE Symposium on Computational Intelligence and Games*, 2005.

[6] R. Balla and A. Fern, "UCT for tactical assault planning in realtime strategy games", *Proceedings of International Joint Conference on Artficial Intelligence*, pp. 40-45. 2009.

[7] R. Thawonmas, H. Matsumoto and T. Ashida, "Ice pambush 3", *2009 IEEE Symposium on Computational Intelligence and Games competition entry*. 2009.