Enhancements for Monte-Carlo Tree Search in Ms Pac-Man

Tom Pepels

June 19, 2012

Abstract

In this paper enhancements for the Monte-Carlo Tree Search (MCTS) framework are investigated to play Ms Pac-Man. MCTS is used to find an optimal path for the agent at each turn, determining the move to make based on randomized simulations. Ms Pac-Man is a real-time arcade game, in which the protagonist has several independent goals but no conclusive terminal state. Unlike games such as Chess or Go there is no state in which the player wins the game. Furthermore, the Pac-Man agent has to compete with a range of different ghost agents, hence limited assumptions can be made about the opponent's behaviour. In order to expand the capabilities of existing MCTS agents, five enhancements are discussed: 1) a variable depth tree, 2) playout strategies for the ghost-team and Pac-Man, 3) including long-term goals in scoring, 4) endgame tactics, and 5) a Last-Good-Reply policy for memorizing rewarding moves during playouts. An average performance gain of 40,962 points, compared to the average score of the top scoring Pac-Man agent during the CIG'11, is achieved by employing these methods.

1 Introduction

Ms Pac-Man is a real-time arcade game based on the popular Pac-Man game. The player controls the main character named Ms Pac-Man (henceforth named *Pac-Man*) through a maze, eating pills and avoiding the ghosts chasing her. The maze contains four so-called power pills that allow the player to eat the ghosts to obtain a higher score. The game has no natural ending. When all pills in a maze are eaten, the game progresses to the next level. Ms Pac-Man inherited its game-mechanics from the original Pac-Man. Moreover, it introduced four different mazes, and more important, unpredictable ghost behaviour. This last feature makes Ms Pac-Man an interesting subject for AI research. The game rules are straightforward, however complex planning and foresight are required for a player to achieve high scores.

Currently two competitions are held for autonomous Ms Pac-Man agents. In the first, *Ms Pac-Man Competition (screen-capture version)* [13], the original version of the game is played using an emulator. Agents interpret a capture of the screen to determine the game's state. Each turn moves are passed to the emulator running the game. The second, *Ms Pac-Man vs Ghost Competition* [16] offers a complete implementation of the game, therefore the screen does not need to be captured by the agents, and the game state is fully available. Furthermore, Pac-Man agents compete with a variety of ghost-team agents also entering the competition.

Although most Pac-Man agents entering the competitions are rule-based, research has been performed on using techniques such as genetic programming [1], neural networks [12] and search trees [15]. Owing to the successful application of Monte-Carlo Tree Search (MCTS) in other games [5], interest in developing MCTS agents for Ms Pac-Man has grown. Samothrakis *et al.* [17] developed an MCTS agent using a Max-n tree with scoring for both Pac-Man and the ghosts. Furthermore, a target location is set as a long-term goal for Pac-Man, MCTS computes the optimal route to the target in order to determine the next move. Other MCTS-based agents were researched for achieving specific goals in Ms Pac-Man, such as ghost avoidance [22] and endgame situations [23] demonstrating the possibilities of MCTS for Pac-Man agents. In 2011 the first MCTS agent won the *Ms Pac-Man screen-capture competition* [13]. Until then rule-based agents lead the competitions. The victorious MCTS agent, Nozomu [10], was designed to avoid so-called 'pincer moves', in which every escape path for Pac-Man is blocked. The approach was successful in beating the leading rule-based agent ICE Pambush [21] with a high score of 36,280.

The research question discussed in this paper is whether strong play is possible when using an MCTS Pac-Man agent to compete in the *Ms Pac-Man vs Ghost Team* competition, and no assumptions can be made on the ghost-team's behaviour. Furthermore, the influence of several enhancements to the MCTS framework are researched: 1) a variable depth tree, 2) playout strategies for the ghost-team and Pac-Man, 3) including long-term goals in scoring, 4) endgame tactics, and 5) a Last-Good-Reply policy [9] for memorizing rewarding moves during playouts.

The paper is structured as follows. First, the Ms Pac-Man framework is introduced, MCTS and the UCT selection policy is explained. Enhancements to the MCTS framework are discussed in detail. Finally, experimental results will be given and a conclusion drawn.

2 Ms Pac-Man

The basic rules of Ms Pac-Man are based on the classic arcade game. Pac-Man initially has three lives, which she loses when coming into contact with a non-edible ghost. In this case, the location of the ghosts and Pac-Man are reset to their initial configuration. The game environment consists of four different mazes, each of which is played once per four levels. The game progresses each *time unit*, allowing Pac-Man to make a move. Ghosts are only allowed to make a move at a junction. On a path between junctions ghosts can only travel forward. When a power-pill is eaten by Pac-Man the ghosts turn blue and become edible, their movement speed decreases and they are instantly forced to reverse their direction. When Pac-Man reaches a score of 10,000 by eating pills and ghosts, she gains a life. Both the ghosts and Pac-Man have one in-game *time unit* of 40 ms. to compute a move at each turn. If no move is returned, a randomly selected move is executed. Each maze is played for 3,000 *time units*, after which the game progresses to the next level. Remaining pills in the maze are then added to Pac-Man's score as a reward for surviving the maze. There are no changes in difficulty when going to the next level. However, the time that ghosts remain edible is decreased when the game advances to the next level. The game ends either if the 16^{th} level is cleared or if Pac-Man has no lives remaining.

3 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) is a best-first search method based on random sampling by Monte-Carlo simulations of the state space for a certain domain [8, 11]. In gameplay this means that decisions are made based on the results of randomly simulated playouts. MCTS has shown promising results when applied to various turn-based games such as Go [14] and Hex [2]. MCTS can be applied to other problems for which the state space can be represented as a tree. A particular challenge for agents playing real-time games is that they are usually characterized by uncertainty, a large state space and open-endedness. However, MCTS copes well when limited time is available between moves and is possible to encapsulate uncertainty in its randomized playouts [5]. The basic version of MCTS consists of four steps, which are performed iteratively until a computational threshold is reached. This may be a set number of iterations, an upper limit on memory usage or a time constraint. The four steps (Figure 1) at each iteration are [6]:

- **Selection**. Starting at the root node, children are selected recursively according to a selection policy. When a leaf node is reached that does not represent a terminal state it is selected for expansion.
- **Expansion**. All children are added to the selected leaf node given available moves.
- **Playout**. A simulated playout is ran, starting from the state of the added node. Moves are performed randomly or according to a heuristic strategy until a terminal state is reached.
- **Backpropagation**. The result of the simulated playout is propagated immediately from the selected node back up to the root node. Statistics are updated along the tree for each node selected during the selection phase and visit counts are increased.

Because results are immediately backpropagated, MCTS can be terminated anytime to determine the decision to be made. The most important characteristic is that MCTS is an evaluation function in itself. No static heuristic evaluation is required when simulations are played randomly until a terminal state. However, it is often beneficial to add domain knowledge for choosing moves made during the playout.

4 Monte-Carlo Tree Search for Ms Pac-Man

This section discusses the enhancements to the MCTS framework for the Pac-Man agent. The agent builds upon the methods proposed in [10] as well as [17]. The structure of the search tree is defined and the following subsections cover the enhancements to the MCTS algorithm.

4.1 Search Tree and Variable Depth

The game's environment is represented by four different mazes. These mazes can directly be represented as a graph where the *junctions* are *nodes*, and *paths* between junctions are *edges*. Pac-Man has the option to make a decision at any location in the graph. At a node she has a choice between more than two available directions. On an edge she can choose to maintain her course or reverse. An example of such a graph is depicted in Figure 2. The associated search tree is shown in Figure 3.

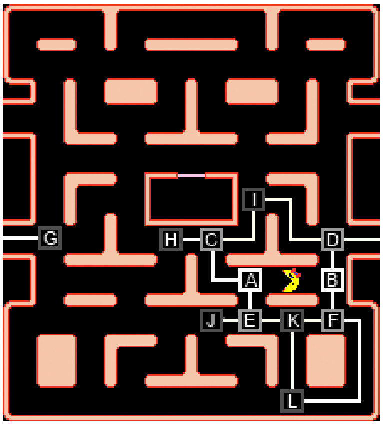

Figure 2: Graph representation of a game state.

Decisions in the tree are the moves made at nodes, i.e. junctions in the maze. Traversing the tree means that Pac-Man moves along an edge until a node is reached. At this

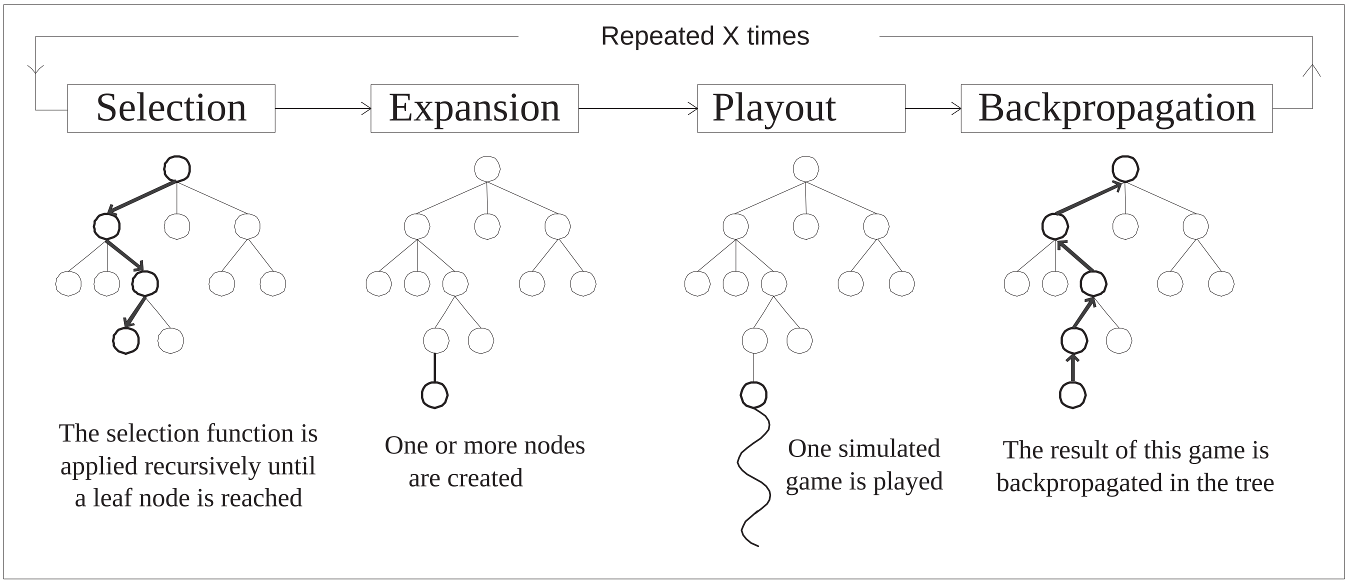
Repeated X times
Selection
Expansion
Playout
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes are created
One simulated game is played
The result of this game is backpropagated in the tree

Figure 1: Strategic steps of Monte-Carlo Tree Search [6].

point either the tree ends and playout starts, or a new edge is chosen based on a child of the current node.

Within the tree, *reverse moves*, i.e. moves that lead back to a parent, are not considered. When a node n_p, representing junction j_p is expanded, each child n_i represents a move that leads to a different junction j_i in the maze, excluding junction j_p.

Nodes store three reward values both averaged and maximized over all their children's values:

1. The maximum and average ghost score S_{ghost}.
2. The maximum and average pill-score S_{pill}.
3. The maximum and average survival rate $S_{survival}$.

The values are used when determining v_i during selection and backpropagation. Furthermore, the final decision is based on these values depending on the currently active tactic (Subsection 4.2).

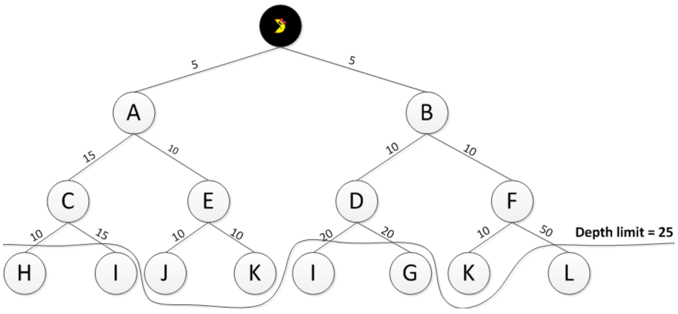

Figure 3: Example tree with variable tree-depth of 25. Based on the game state in Figure 2.

Ikehata and Ito [10] used a search tree restricted in depth to a fixed number of edges, without regard for the length of these edges. Although the search tree in this paper is constructed similarly, the search path is variably determined by a threshold path-length T_{path}. A leaf is only expanded if the length of the path to the root node does not exceed T_{path} (Figure 3). The variable depth search prevents the agent from choosing 'quick fixes' when in danger, i.e. it may be safer to traverse a short path in the game when Pac-Man is in danger, than a long path which could be the case when tree-depth is limited by a fixed number of edges. Furthermore, the scoring potential over all possible paths in the tree is normalized due to the uniform length of paths in the tree.

4.2 Tactics

According to the current game state a tactic [10] for determining the behaviour of Pac-Man is selected. Tactics are based on the three subgoals of Pac-Man. At any time one of the following is active:

- The **Ghost score** tactic is selected if edible ghosts are in the range of Pac-Man, and the maximum survival rate is above the threshold $T_{survival}$.
- The **Pill score** tactic is applied when Pac-Man is safe and there are no edible ghosts in range, and the maximum survival rate is above the threshold $T_{survival}$.
- The **Survival** tactic is used when the maximum survival rate of the previous search was below the threshold, $T_{survival}$.

The v_i value used for selection and backpropagation is based on the current tactic. It is either the maximum survival rate, $v_i = S_{survival}$, when the survival tactic is active, or the current score multiplied by the survival rate,

$v_i = S_{ghost} \times S_{survival}$ or $v_i = S_{pill} \times S_{survival}$, for the ghost and pill tactics, respectively. The survival rate $S_{survival}$ is interpreted as a predictive indicator that the node's reward will be achieved.

The final move to be played is determined by selecting a child from the root node with the highest *maximum* v_i score over all its children, based on the current tactic. If the current tactic provides no feasible reward i.e. all scores are 0, it is replaced according to the order in the above list. This occurs when, for instance, the nearest pill or edible ghost is out of the search tree's range. If this is the case for several consecutive moves, the endgame tactic is applied (Subsection 4.8).

4.3 Selection and Expansion

During the selection step, a balance is required between selecting nodes that maximize the expected reward (exploitation) and exploring the tree (exploration). Therefore a tree policy is required to explore the tree for rewarding decisions and finally converge to the most rewarding one. Upper Confidence bound applied to Trees (UCT) [11] is derived from the UCB1 function [3] for maximizing the rewards of a multi-armed bandit. UCT balances the exploitation of rewarding nodes whilst allowing exploration of lesser visited nodes. The policy that determines which child to select given the current node is the one that maximizes the following equation:

$$v_i + C\sqrt{\frac{\ln n_p}{n_i}}$$

v_i is the score of the current child based on the active tactic, defined in Subsection 4.2. In the second term, n_p is the visit count of the node and n_i the visit count of the current child. C is the exploration constant to be determined by experimentation.

UCT is applied when the visit count of a child node is above a threshold T. When a node's visit count is below this threshold, a child is selected randomly. In the case of Ms Pac-Man, the threshold used is 3, which ensures a higher confidence on the safety of the path of playouts through the node. An otherwise safe and/or rewarding node may have resulted in death the first time it is expanded, due to the non-deterministic nature of the game. Using the threshold ensures that this node is explored again, increasing the confidence on the safety of the decision.

4.4 Playout

During the playout, Pac-Man and the ghost team make moves in a fully functional game state. Playout consists of two phases: 1) the *tree phase*, in which moves by Pac-Man are made according to the nodes selected during the selection phase, and 2) the *playout phase*, in which moves by Pac-Man are performed according to a randomized playout strategy described in Subsection 4.6.

During the *tree phase*, the path represented by the nodes selected during the selection phase is traversed. Moves corresponding to each selected node during the selection phase are performed by Pac-Man. Meanwhile, the ghosts move according to the playout strategy (Subsection 4.6). This provides the basis for determining the achievable score of the selected path while allowing for Pac-Man to be captured by the simulated ghost team. If Pac-Man does not lose a life during the tree phase, and the junction represented by the leaf node is reached, the *playout phase* starts. Both Pac-Man and the ghosts move according to the playout strategy.

In two cases, the *tree phase* can be interrupted due to a change in the game state which cannot be predicted by the search tree:

1. If Pac-Man loses a life during the *tree phase*, the *playout phase* is started from the last-visited junction. Losing a life during the *tree phase* is basically a suicide-move, as Pac-Man may only travel forward. Therefore the playout can still be run to determine whether Pac-Man could have avoided the loss of life.
2. Pac-Man eats a ghost or a power pill, in this case the *playout phase* is started immediately.

A game of Ms Pac-Man ends when either Pac-Man loses all lives, or the 16^{th} level is cleared. It is neither useful nor computationally achievable within the strict time limit of 40 ms., to run a playout until one of these conditions holds.

The goal of the playouts is to determine the short- and long-term safety and reward of a selected path. Therefore different *stopping conditions* for playouts should be used. Two natural stopping conditions can be considered, either Pac-Man loses a life (dies), or the game progresses to the next maze. However, to prevent lengthy playouts, additional stopping conditions have to be introduced. Therefore during the *playout phase*, moves are made until one of the following four conditions applies:

1. A pre-set number of time units T_{time} have passed.
2. Pac-Man is considered dead, i.e.:
   - Came into contact with a non-edible ghost.
   - Is trapped at the end of the playout, every available path is blocked by ghosts.
3. The next maze is reached.
4. Pac-Man eats a power pill while edible ghosts active. As penalty, this is considered the same as not surviving the playout.

When the playout ends for any of the aforementioned reasons, Pac-Man's score is determined based on the three subgoals of the game. Results of a playout consist of three values:

- $$R_{survival} = \begin{cases} 0 & \text{if Pac-Man died} \\ 1 & \text{if Pac-Man survived} \end{cases}$$
- $R_{pill} \in [0, 1]$ the number of pills eaten, normalized by the number of pills at the root.

- $R_{ghost} \in [0, 1]$ the number of ghosts eaten, normalized by the number of ghosts in the ghost team.

The goal for Pac-Man during the playout is acquiring the highest score possible whilst avoiding a loss of life. The ghosts have three goals: 1) ensure that Pac-Man loses a life by trapping her, i.e. every possible path leads to a non-edible ghost, 2) ensure the lowest ghost-reward R_{ghost}, which increases when Pac-Man eats edible ghosts, and 3) decrease as much as possible the number of pills Pac-Man can eat.

4.5 Long-Term Goals

Time is an important aspect of the game. Ghosts need to be eaten as fast as possible such that they do not return to their normal state when edible, remaining in a maze longer than necessary increases the risk of being trapped. Furthermore, after 10,000 points, Pac-Man gains a life. These are examples of long-term goals in the game. Any MCTS implementation looks at short-term rewards when running playouts. However, Pac-Man has several long-term goals to consider. To estimate the rewards of long-term goals, results are altered for both pill and ghost rewards.

To encode the long-term goal in the playouts' ghost reward R_{ghost}, for every eaten ghost its reward is multiplied with $t_{edible}(g)$, the edible time remaining before the ghost was eaten. This ensures that ghosts eaten early are preferred over ghosts eaten later. Furthermore, when Pac-Man eats a power-pill (while no edible ghosts are active) during playout, she must achieve a ghost score higher than 0.5 at the end of the playout. If this is not the case, i.e. the ghosts were too far away to be eaten in time, the pill-reward R_{pill} is set to 0. This enables Pac-Man to wait for the ghosts to be close enough before eating a power-pill. If the minimum ghost score of 0.5 is achieved after eating a power-pill, the pill-reward is increased: $R_{pill} = R_{pill} + R_{ghost}$. This high reward ensures that Pac-Man eats a power-pill when the opportunity to eat the ghosts easily arises.

The longer Pac-Man remains in a maze, the higher the probability that she will be eaten. There are only four power pills available per maze to provide a guaranteed safe escape. Eating all pills in a maze before the game naturally progresses to the next maze is therefore a beneficial long-term goal. Points for eating pills are only added to R_{pill} during playout, when the current edge has been cleared i.e. the last pill on the edge is eaten. It ensures that Pac-Man prefers to eat all pills on the edges visited, rather than leaving isolated pills which may become hard to reach as the game progresses.

4.6 Playout Strategy

During playout, Pac-Man and the ghost team's moves are simulated simultaneously. Both the ghosts and Pac-Man have the possibility to store moves as a Last-Good-Reply (LGR) [9]. Any time the ghost team traps Pac-Man during playout, the ghosts' moves based on Pac-Man's last visited junction are remembered. Similarly, Pac-Man's moves at junctions are remembered each time she survives a playout. Otherwise moves are forgotten [4] and no longer part of the LGR move-policy. In this subsection the playout strategies for the ghosts and Pac-Man are detailed. The strategies were designed to ensure that any time Pac-Man does not survive the playout (Subsection 4.4), it is due to all possible paths being blocked by ghosts. Therefore, the $S_{survival}$ score stored at nodes in the tree can be considered as an indicator of the probability of a *pincer move* occurring along the path [10].

GHOST PLAYOUT STRATEGY. The goal of the ghosts is to trap Pac-Man in such a way that every possible move leads to a path blocked by a ghost, i.e. a pincer move. The ghosts therefore are assigned a random target-location vector $\vec{target}$ that determines whether an individual ghost is to approach the front or rear of Pac-Man.

Ghosts move based on an ϵ-greedy strategy [19, 18]. With a probability $\epsilon = 0.05$ at each turn, a ghost makes a random move. With probability $1 - \epsilon$ the ghosts move according to strategic rules, derived from the rules proposed in [10]. For the ghosts there are two exclusive cases to consider, i.e. not-edible or edible, when selecting a move during playout. Moreover, there is a third case which overrules a selected move in any case.

Case 1, if ghost g_i is *not* edible. A move is selected according to the following rules:

1. If the ghost can make a move that can immediately trap Pac-Man it is performed.
2. If a move, that is a Last-Good-Reply, is available based on Pac-Man's last junction, it is performed.
3. If the ghost is in the immediate vicinity of Pac-Man, i.e. within 10 distance units, the ghost moves along the next direction of the shortest path to Pac-Man.
4. If the ghost is on a junction directly connected to the edge that Pac-Man is located on, the ghost chooses the move that leads to this edge.
5. Otherwise, the ghost moves closer to the assigned target location. Based on the value of $\vec{target}_i$ this is either the nearest junction in front or behind Pac-Man.

Case 2, if ghost g_i is edible, a move is chosen that maximizes the distance between him and Pac-Man.

Case 3, if a ghost is to move on an edge currently occupied by another ghost moving in the same direction, the move is eliminated from the ghost's selection and a different move is selected randomly. This policy ensures that ghosts are spread out through the maze, increasing their possible trapping or catching locations. It also prevents multiple ghosts from chasing Pac-Man at the same (or similar) distance and location shown in Figure 4.

PAC-MAN PLAYOUT STRATEGY. Moves made by Pac-Man are prioritized based on safety and possible reward. When more than one move has the highest priority, a random tie-breaking rule is applied. Before discussing the strategy in

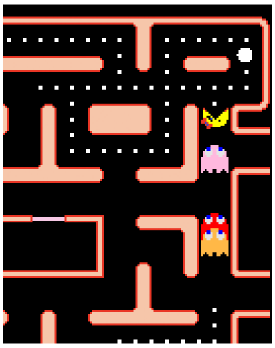

Figure 4: Ghosts chasing Pac-Man from similar distance and location.

detail, the concept of a *safe move* has to be defined first. A safe move is any move that leads to an edge which:

- Has no non-edible ghost on it moving in Pac-Man's direction.
- Next junction is safe, i.e. in any case Pac-Man will reach the next junction before a non-edible ghost.

During the playout Pac-Man moves according to the following set of rules. If Pac-Man is at a junction the following rules apply, sorted by priority:

1. If a safe move that is a Last-Good-Reply is available, it is performed.
2. If a safe move leads to an edge that is not cleared, i.e. contains any number of pills, it is performed.
3. If all safe edges are cleared, select a move leading to a safe edge.
4. If no safe moves are available, a random move is selected.

If Pac-Man is on an edge, she can either choose to maintain her current path or reverse course. The following rules consider the cases when Pac-Man is allowed to reverse:

- There is a non-edible ghost heading for Pac-Man on the current path.
- A power-pill was eaten, in this case the move which leads to the closest edible ghost is selected.
- A ghost in the edible state was eaten, the move which leads to the closest next edible ghost is selected.

In any other case Pac-Man continues forward along the current edge. Note that, if Pac-Man previously chose to reverse on the current edge, she may not reverse again until she reaches a junction.

4.7 Backpropagation

Results are back-propagated from the expanded leaf node to the root based on maximum backpropagation [8]. Scores stored at each node represent the maximum scores of its children based on v_i according to the current tactic (Subsection 4.2). Whereas most games use average backpropagation, maximization is chosen since each move at a junction can have altogether different results [10]. For example, at a junction Pac-Man has two options to move. A decision to go left may lead to a loss of life for Pac-Man in all playouts, whereas a choice to go down is a determined to be safe in every playout. When using averaged values, the resulting survival rate is 0.5, whereas maximum backpropagation would result in the true survival rate of 1.

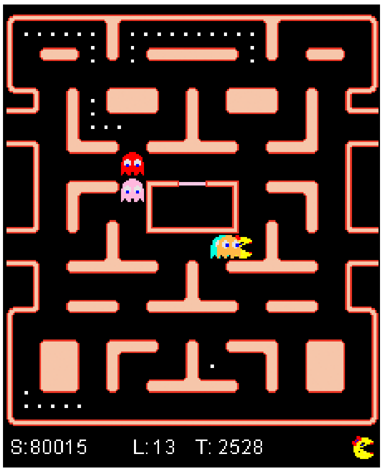

Figure 5: Example of an endgame situation, $maze\_time > 2000$ and the nearest pill is outside the search tree's range.

4.8 End Game Tactics

During the final moments in a maze, or when Pac-Man is located in an isolated area of the maze, the nearest possible reward, i.e. an edible ghost or a pill, may be out of the search tree's range (Figure 5). These cases are considered as the *endgame*, Pac-Man will no longer be motivated to choose one move over another due to the lack of rewards. This leads to a continuous fallback to the survival tactic as defined in Subsection 4.2. This is problematic because the survival tactic only provides motivation when Pac-Man is in danger of being eaten by the ghosts. In this case, the endgame tactic is applied when one of the following criteria holds:

1. No move could be selected based on the active tactic for 5 consecutive moves, i.e. $S_{pill} = 0$ or $S_{ghost} = 0$ based on the active tactic.
2. The $maze\_time > 2000$, i.e. the time Ms. Pac-Man was in the current maze is higher than 2000 time units.

When the endgame tactic is active, similar to [17] and [23] a target location is selected based on a heuristic evaluation of the game state. The pseudo-code for the algorithm used to select the current target t is listed in Algorithm 1. A new target is selected each time Pac-Man is at a junction in the game.

When a target is set, at the end of the playout phase R_{pill} (Subsection 4.4) is replaced by:

$$R_{target} = \begin{cases} 0 & \Delta Dist \leq 0 \\ \Delta Dist & \Delta Dist > 0 \\ 1 & \text{Target location was reached} \end{cases}$$

where $\Delta Dist$ is the normalized difference in distance to t at the start and end of the playout.

If the endgame tactic was only applied for the first reason, thus $maze\_time < 2000$, it is possible to terminate the endgame tactic, returning to one of the default tactics discussed in Subsection 4.2. This occurs when a move is selected with a score, S_{pill} or S_{ghost} based on the active tactic, of at least 0.5, implying that there is again sufficient motivation to select a move based on one of the default tactics.

Algorithm 1 Select endgame target t

```
if $edible\_ghost\_in\_range$ then
    $t \leftarrow nearest\_edible\_ghost$
else if $power\_pill\_available$ or $maze\_time > 2000$ then
    $t \leftarrow nearest\_power\_pill$
else
    $t \leftarrow nearest\_pill$
end if
```

5 Experiments

In the following subsections the experimental setup will be detailed, and experimental results discussed.

5.1 Experimental setup

The MCTS PAC-MAN agent was implemented using the framework provided by the *Ms Pac-Man vs Ghost competition* [16]. Furthermore, an MCTS GHOST TEAM using enhancements within the MCTS framework discussed in this paper was developed. The MCTS GHOST TEAM uses the strategic playout and tactics discussed in this paper. However, due to the difference in goals between Pac-Man and the ghost team, the MCTS GHOST TEAM uses a constant depth tree, no endgame tactics, and no long-term goals.

The version of the Ms Pac-Man game framework used is WCCI12_1.1. It includes pre-computed distances for the four mazes, providing a fast method for determining shortest paths and distances. Both the framework and the agent were developed in Java.

Results are comprised of the average, maximum and minimum score, the average number of lives remaining and average maze reached at the terminal state of the game. Average scores are rounded to the nearest integer. Each time 100 runs are performed, allowing the agents the official 40ms. to run playouts and compute a move.

The following values, determined by trial-and-error, were used for the parameters discussed in the paper: the minimum survival rate $T_{survival} = 0.7$, the maximum variable tree depth $T_{path} = 55$, the maximum time units per playout phase $T_{time} = 80$, and a UCT constant $C = 1.5$ was used.

To determine the influence of the proposed enhancements, results are compared to agents with a single enhancement disabled. Additional experiments are ran using agents with, 1) a fixed node depth-limit, 2) no endgame tactic, 3) a randomized playout strategy, and 4) the Last-Good-Reply policy disabled.

5.2 Results

Experiments were ran to evaluate the performance of the MCTS PAC-MAN agent against the benchmarking ghost team LEGACY2THERECKINING (LEGACY2 T.R.) provided by the competition framework. Furthermore, the agent's performance is tested against the MCTS GHOST TEAM.

Table 1: Achieved scores, 100 games

Pac-Man agent: MCTS PAC-MAN				
Ghost Team agent	Avg. score	Max. score	Min. score	95% conf. int.
LEGACY2 T.R.	$107,561$	$127,945$	$40,495$	$2,791$
MCTS GHOST TEAM	$36,477$	$65,195$	$2,830$	$2,498$
Pac-Man agent: STARTER PAC-MAN				
Ghost Team agent	Avg. score	Max. score	Min. score	95% conf. int.
LEGACY2 T.R.	$4,260$	$9,050$	$1,460$	280
MCTS GHOST TEAM	$2,799$	$5,980$	$1,040$	211

Table 1 shows the resulting scores of our MCTS PAC-MAN agent versus the benchmarking team LEGACY2 T.R. and the MCTS GHOST TEAM. 100 games were played to determine the scores. For comparison the same ghost teams played 100 games against the STARTER PAC-MAN agent which uses a limited rule set. From this we can conclude that the MCTS GHOST TEAM outperforms LEGACY2 T.R. when playing against both the MCTS PAC-MAN and STARTER PAC-MAN agents. Furthermore, it is clear that the MCTS PAC-MAN agent outperforms the STARTER PAC-MAN by far.

Currently, no official competition results in which the WCCI12_1.1 version of the Ms Pac-Man framework was used exist. Past competitions used a similar framework, which also provided the benchmarking ghost team LEGACY2 T.R.. However, it is not the case that the underlying data structures provided in the current version of the software provide an unfair advantage to either the ghost team or Pac-Man. Moreover, since the LEGACY2 T.R. ghost team's rule-base has remained the same, a rough comparison may be drawn. In Table 2, the top-3 scoring Pac-Man controllers during the CIG'11 [7] are presented with their achieved scores versus the LEGACY2

Table 2: CIG'11 rankings, 10 games

Ghost Team: LEGACY2 T.R.					
	Pac-Man agent	Avg. score	Max. score	Min. score	95% conf. int.
1	SPOOKS	66,599	76,080	35,270	7,378
2	PHANTOMMENACE	56,313	88,090	30,350	13,311
3	ICEPAMBUSH_CIG11	20,619	29,320	9,160	4,384
-	MCTS PAC-MAN	107,561	127,945	40,495	2,791

T.R. ghost team. Because the results of these agents differ substantially from our MCTS PAC-MAN agent it is safe to conclude that the performance has increased. An average performance gain of 40,962 points, based on the top scoring Pac-Man agent during the CIG'11, is achieved by our MCTS agent.

To test each of the proposed enhancements for the MCTS agent, 100 games per enhancement were played against the LEGACY2 T.R. ghost team. The enhancements were individually disabled or defaulted by the following:

- The playout strategy was replaced by a simple random strategy for both ghosts and Pac-Man, in which the Pac-Man cannot reverse and chooses each move randomly. The ghosts always choose the path that leads closer to Pac-Man.
- A constant tree-depth of 4 ply, determined by trial-and-error, replaces the variable depth tree enhancement.
- The Last-Good-Reply (LGR) policy was disabled.
- The endgame tactic was disabled altogether. Only the default strategies can be used when selecting a move.

Results of these games are shown in Table 3.

The random playout has the largest impact on overall scoring, since MCTS is dependent on the results of its playouts to determine the best move. It is followed by the constant tree-depth, which causes discrepancies when determining the scoring potential and survival rate over each path in the tree.

Both the LGR policy and endgame tactic have a low impact on the mean scores, lives remaining, and maze reached. However, it is likely that against more advanced ghost teams these enhancements play a more important role. Because the LEGACY2 T.R. ghost team was not designed to trap Pac-Man, the increase in performance may be lower than were the agent to perform against more intelligent ghost teams.

For both the random playouts and constant tree depth, according to Table 4, the number of lives Pac-Man has at the end of each run, and the average maze reached is lower. It implies increased survivability of the agent due to these enhancements.

5.3 Results WCCI 2012

Table 3: Disabled enhancements, scores, 100 games

Ghost Team: LEGACY2 T.R., Pac-Man agent: MCTS PAC-MAN				
Enhancement disabled	Avg. score	Max. score	Min. score	95% conf. int.
Strategic playout	44,758	65,270	11,900	2,310
Var. depth tree	101,836	124,925	43,595	3,326
Last-Good-Reply	105,723	125,885	45,830	2,964
Endgame tactic	108,020	125,440	40,945	2,551
MCTS PAC-MAN	107,561	127,945	40,495	2,791

Table 4: Disabled enhancements, statistics, 100 games

Ghost Team: LEGACY2 T.R., Pac-Man agent: MCTS PAC-MAN				
Enhancement disabled	Avg. lives remaining	95% conf. int.	Avg. level reached	95% conf. int.
Strategic playout	0.55	0.19	12.76	0.70
Var. depth tree	1.21	0.24	14.32	0.53
Last-Good-Reply	1.78	0.25	15.01	0.45
Endgame tactic	1.70	0.24	15.21	0.39
MCTS PAC-MAN	1.76	0.25	15.19	0.41

The MCTS PAC-MAN and MCTS GHOST TEAM were entered in the *Ms Pac-Man vs Ghost team competition* [16] held for the WCCI 2012 under the nickname MAASTRICHT. At the time of writing preliminary results of the games played rank the MCTS PAC-MAN agent in second place of 63, and the MCTS GHOST TEAM 12^{th} place of 55. Table 5 shows the top ranked Pac-Man agents and their scores, whereas table 6 shows the top ranked Ghost teams and their scores.

Table 5: Preliminary Pac-Man results WCCI 2012

Rank	Agent name	Avg. score	Games played
1	EIISOLVER	90,448	179
2	MAASTRICHT	88,502	203
3	ICEP-FEAT-SPOOKS	85,084	183

6 Conclusion & Future Research

The discussed enhancements for the Monte-Carlo Tree Search (MCTS) framework have resulted in a Pac-Man agent achieving a high score of 127,945 points versus the LEGACY2THERECKONING ghost team. Regarding the results of previous competitions, an average performance gain of 40,962 points, compared to the top scoring Pac-Man agent during the CIG'11, is achieved by our MCTS agent. The variable depth tree and strategic playout ensure the highest increase in scores. Although the endgame tactics and Last-Good-Reply policy did not increase the final scores significantly, they may be crucial to competing with more advanced ghost teams. However, it is possible that when the playout strategy is further improved, LGR will have less effect on overall scores. Based on the results we may conclude that the MCTS framework makes strong Pac-Man agents possible.

Table 6: Preliminary Ghost results WCCI 2012

Rank	Agent name	Avg. score	Games played
1	EIISOLVER	2,818	216
2	MEMETIX	2,842	211
3	GREANTEA	2,967	224
12	MAASTRICHT	9,274	199

The performance of the MCTS agent could be improved by using Heat-maps [10] to determine the most dangerous locations in a maze. This could improve the pill-reward. Although the proposed ghost playout strategy is designed to maximize the possibility of a pincer-move, ghosts do not always capture Pac-Man when possible in playouts. To improve the playout-phase further two improvements could be made. 1) N-Grams [20], when applied to the playout phase may improve the possibility of Pac-Man being caught whenever possible, increasing the confidence of the safety of moves in the search tree. 2) The knowledge rules of fast rule-based agents from the upcoming competitions could be used if they perform well.

References

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

[2] Arneson, B., Hayward, R. B., and Henderson, P. (2010). Monte-Carlo tree search in Hex. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 4, pp. 251–258.

[3] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, Vol. 47, No. 2-3, pp. 235–256.

[4] Baier, H. and Drake, P. D. (2010). The power of forgetting: Improving the last-good-reply policy in Monte Carlo Go. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 4, pp. 303–309.

[5] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte-Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 1, pp. 1–43.

[6] Chaslot, G. M. J-B., Winands, M. H. M., Herik, H. J. van den, Uiterwijk, J. W. H. M., and Bouzy, B. (2008). Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, Vol. 4, No. 3, pp. 343–357.

[7] Ms Pac-Man vs Ghost competition CIG11 ranknings. `http://cig11.pacman-vs-ghosts.net/rankings.php`. Accessed, May 2012.

[8] Coulom, R. (2007). Efficient selectivity and backup operators in Monte-Carlo tree search. *Computers and games: 5th international conference (CG)*, Vol. 4630, pp. 72–83, Springer.

[9] Drake, P. D (2009). The last-good-reply policy for Monte-Carlo Go. *International Computer Games Association Journal*, Vol. 32, No. 4, pp. 221–227.

[10] Ikehata, N. and Ito, T. (2011). Monte-Carlo tree search in Ms. Pac-Man. *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 39–46, IEEE.

[11] Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. *Machine Learning: ECML 2006*, Vol. 4212 of *Lecture Notes in Computer Science (LNCS)*, pp. 282–293. Springer.

[12] Lucas, S. M. (2005). Evolving a neural network location evaluator to play Ms. Pac-Man. *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pp. 203–210, IEEE.

[13] Ms Pac-Man Competition, screen-capture version. `http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html`. Accessed May, 2012.

[14] Rimmel, Arpad, Teytaud, Olivier, Lee, Chang-Shing, Yen, Shi-Jim, Wang, Mei-Hui, and Tsai, Shang-Rong (2010). Current frontiers in computer Go. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 4, pp. 229–238.

[15] Robles, D. and Lucas, S. M. (2009). A simple tree search method for playing Ms. Pac-Man. *IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 249–255, IEEE.

[16] Rohlfshagen, P. and Lucas, S. M. (2011). Ms Pac-Man Versus Ghost Team CEC 2011 competition. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 70–77.

[17] Samothrakis, S., Robles, D., and Lucas, S. M. (2011). 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.

[18] Sturtevant, N. (2008). An analysis of uct in multiplayer games. *Computers and Games*, Vol. 5131 of *Lecture Notes in Computer Science (LNCS)*, pp. 37–49. Springer.

[19] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction.* MIT Press.

[20] Tak, M. J. M., Winands, M. H. M., and Björnsson, Y. (2012). N-grams and the last-good-reply policy applied in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 2. Accepted.

[21] Thawonmas, R. and Ashida, T. (2010). Evolution strategy for optimizing parameters in Ms Pac-Man controller ICE Pambush 3. *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 235–240.

[22] Tong, B. K. B. and Sung, C. W. (2011). A Monte-Carlo approach for ghost avoidance in the Ms. Pac-Man game. *International IEEE Consumer Electronics Society's Games Innovations Conference (ICE-GIC)*, pp. 1–8, IEEE.

[23] Tong, B. K. B., Ma, C. M., and Sung, C. W. (2011). A Monte-Carlo approach for the endgame of Ms. Pac-Man. *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 9–15, IEEE.