Fuzzy Q-Learning in a Nondeterministic Environment: Developing an Intelligent Ms. Pac-Man Agent

Lori L. DeLooze and Wesley R. Viner

Abstract— **This paper reports the results from training an intelligent agent to play the Ms. Pac-Man video game using variations of a fuzzy Q-learning algorithm. This approach allows us to address the nondeterministic aspects of the game as well as finding a successful self-learning or adaptive playing strategy. The strategy presented is a table based learning strategy, in which the intelligent agent analyzes the current situation of the game, stores various membership values for each of the several contributors to the situation (distance to closest pill, distance to closest power pill, and distance to closest ghost), and makes decisions based on these values.**

I. Introduction

Ms. Pac-Man was originally produced by Midway and debuted in 1981. It was later adopted by Namco, the creator of Pac-man, and at that point became the official sequel to the original Pac-man video game [1]. Ms. Pac-Man is a point based game in which the player guides the Ms Pac-man agent around a maze filled with edible pills, power pills, and ghosts. The player must avoid running into ghosts, which will eat and kill Ms Pac-man upon contact. In order to beat a level, the player must devour, by traversing over, every pill and power pill in the map. Pills are worth 10 points and power pills are worth 50 points. For a short time period after eating a power pill, Ms Pac-man has the ability to eat ghosts. The first edible ghost consumed during this period is worth 200 points, the second is worth 400, the third is worth 600 and the forth is worth 800 points. Therefore, a strategy of eating multiple edible ghosts with each power pill can be very effective in increasing your score. There are also additional treats that will randomly appear during each level but these will be ignored for this study. Every pill, power pill, or edible ghost eaten adds points to the total score. When all the pills and power pills on a given level are gone, the player advances to the next level. The player can die three times before the game is over.

A Ms Pac-man software controller competition has recently been included in several international conferences in Computational Intelligence, such as CIG2008, WCCI2008, CEC2008 and CEC2009 [2]. While artificial intelligence agents with hardcoded rules have been allowed in the competition, and coincidentally hold some of the highest scores to date, the primary focus is on creating an intelligent Ms. Pac-Man agent that learns on its own and creates its own strategy [3].

The competition allows the use of either the web version of the Ms Pac-Man game [4] or Microsoft's Revenge of the Arcade. The following experiments were run using the latter and the screen capture software provided on the competition web page. The intelligent agent interfaces with the game using the screen capture of the game and analyzing the individual pixels in the picture. By looking for pixel groups of matching RGB color values and groups or clusters of particular sizes, it is possible to determine the location of the most important elements of the game (ie. Ms. Pac-Man, pills, power pills and ghosts). The data gathered during the screen capture becomes the basis for the intelligent agent's decisions. Figures 1 and 2 are the screen capture of the Revenge of the Arcade Ms. Pac-Man game and the corresponding interpretation of the game elements. The screen capture is repeated constantly and quickly captures new images of the screen, allowing the agent to determine evolving action throughout the game and to respond appropriately.

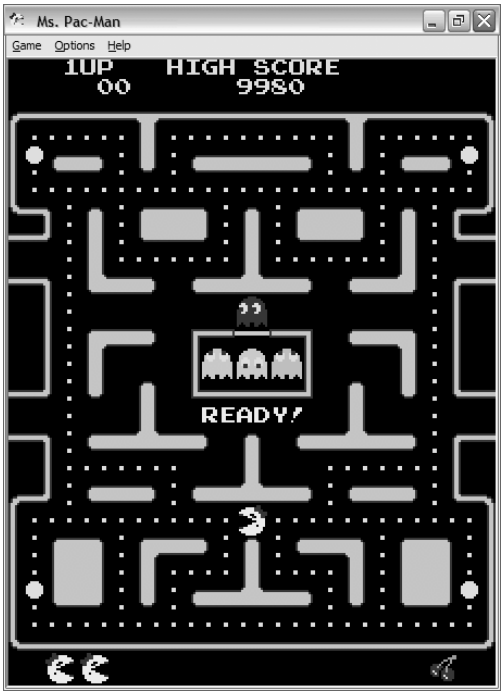

Figure 1. Screen Capture of Ms. Pac-Man Game

Lori L. DeLooze is an Assistant Professor at the United States Naval Academy, Annapolis MD 21402 USA (phone: 410-293-6820; fax: 410-293-2686; email: delooze@usna.edu).
Wesley R. Viner is a Second Lieutenant in the United States Marine Corp and a graduate of the United States Naval Academy, Annapolis, MD 21402 USA

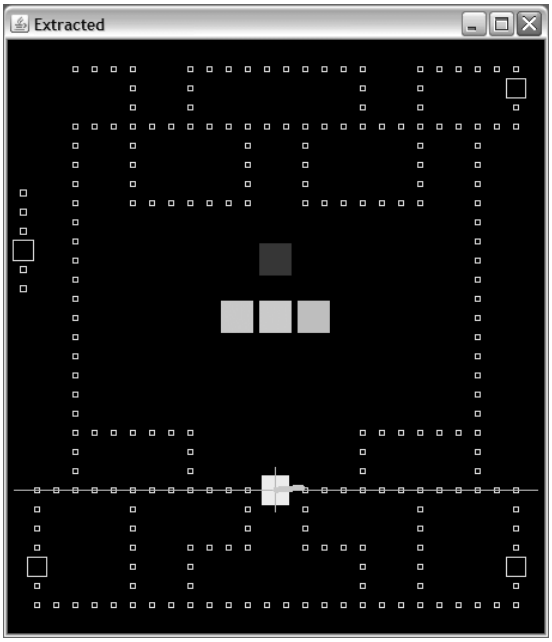

Figure 2. Extracted Game Elements for Ms. Pac-Man

Ms. Pac-Man provides an excellent framework and environment for studying decision-making and self-learning computational techniques in a nondeterministic environment. The game environment is relatively simple to understand yet requires the development of successful strategies for planning, game play and priority management. Successful research in the predatory-prey aspects of Pac-man style games may prove useful in real-world applications. History has proven that this is often the case, and games have frequently been used to study and perfect various artificial intelligence techniques. The primary focus of this study is on a self-learning computational technique called Fuzzy Q-Learning (FQL) that uses fuzzy state aggregation and Q-learning and its application to the development of an autonomous agent.

II. FUZZY Q-LEARNING

A. *Fuzzy State Aggregation*

Fuzzy state aggregation builds on the cornerstone concept of the fuzzy set [5]. The fuzzy set divides a value range into separate, yet overlapping, labeled groups called sets. External values are described by their degree of membership within each of the sets. As an example, consider a fuzzy set used later in the paper for the distance from the agent to the nearest ghost. As shown in Figure 3, the range of possible distances is broken down into several partially overlapping groups: Low, Medium, and High. A distance of 20 or less has a membership value of 100% Low and 0% Medium and 0% High : {100.0,0.0,0.0}. Likewise, the distance value 50 is 20% Low, 80% Medium and 0% High: {20.0,80.0,0.0}. Thus, the distance value 20 is purely Low while distance value 50 is mostly Medium but also a bit Low.

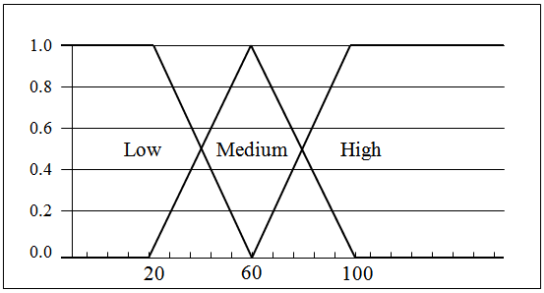

Figure 3. The Fuzzy Set for the Distance to Nearest Ghost

Fuzzy state aggregation (FSA) is a variation on Singh's soft state aggregation [6]. Given a number of fuzzy sets, FSA builds aggregate states based on the subranges of each fuzzy set. The aggregate states represent the status of an environment (problem) as described by the fuzzy sets and thus minimizes the number of states with which the learning algorithm must deal. Under FSA, a state is represented by its degree of simultaneous membership in each of the aggregate states.

For example, consider the Distance to Ghost fuzzy set again and a new fuzzy set for Distance to Power Pill that is also separated into sets of Low, Medium, and High. Combining the subsets of each fuzzy set produces the nine aggregated fuzzy states shown in Table 1. Assume the membership in the Distance to Ghost and Distance to Power Pill are Fuzzy sets are {0.2, 0.8, 0.0} and {0.0, 0.3, 0.7}, respectively.

TABLE 1
AN EXAMPLE OF FSA MEMBERSHIP

FSA STATE	DISTANCE TO GHOST	DISTANCE TO POWER PILL	FSA MEMBERSHIP
1	LOW	LOW	0.10
2	LOW	MED	0.25
3	LOW	HIGH	0.45
4	MED	LOW	0.40
5	MED	MED	0.55
6	MED	HIGH	0.75
7	HIGH	LOW	0.00
8	HIGH	MED	0.15
9	HIGH	HIGH	0.35

Equation 1 shows how to compute the FSA membership for the example.

$$(s1, s2) = \frac{m(Dist_G, s1) + m(Dist_{pp}, s2)}{2} \quad (1)$$

The function $m(f,s)$ takes a fuzzy set, f, and a subset ϕ of f and returns a fractional fuzzy state membership (e.g. $m(Dist_G, Low) = 0.20$ while $m(Dist_{pp}, Low) = 0.0$.

Q-learning [7,8] is one of the simplest and most commonly used reinforcement learning techniques. Q-learning assigns values to state-action pairs $Q(s,a)$ which implicitly represents a policy. The algorithm works by selecting one action from many possible given the current

state. In its simplest form, which is used for this paper, the action with the highest Q-value is selected. After selecting the action, *a*, the Q-value representing the state-action pair is updated based on a reward *r* received as a result of the action and an expected reward given by examining the Q-values of the next state, s, according to the function:

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a), \right] \quad (2)$$

where $\alpha \in [0,1]$ is the learning rate and $\gamma \in [0,1]$ is the discount factor of the next state's expected value.

B. FSA Q-learning

Fuzzy logic enables the representation of continuous state spaces as discrete, thereby making it possible to implement Q-learning for continuous states spaces. The combination of a fuzzy logic controller (FLC) with Q-learning, known as Fuzzy Q-Learning (FQL), has been used for many single robot applications [9,10]. Notable examples include soccer ball chasing behavior for a Sony Aibo robot [11,12] and intercepting a passed ball [13]. In [14] a neural network was combined with an FLC to allow a soccer player agent to adapt to environmental changes.

In a domain with a large state-space it is inefficient to learn Q-values for each state-action pair. Thus it is natural to employ Q-learning with some form of state aggregation. Using the term *Q(s,a,r)* to approximate *Q(s,a)* where *r* is a vector of learned parameters, the parameter update rule for a timestep *t* is:

$$r_t = r_t + \alpha\delta\Delta r_t Q(s,a,r_t), \quad (3)$$

where α is the learning rate and δ_t is the Bellman error. The Bellman error is given by:

$$\delta = g(t) + \gamma \max_{a} Q(s_{t+1},a) - Q(s_t,a), \quad (4)$$

where g(t) is the net cost of the action taken and γ is the discount rate.

Previously, [15] used fuzzy state aggregation as the function approximation architecture. In doing so, a Q-value, *q(k),* was maintained for each aggregate fuzzy state *k* in the set of all aggregate fuzzy states *K*. To determine the Q-value of the state-action pair, *Q(s,a),* the following equation is used:

$$Q(s,a) = \sum_{k \in K} q(k)\ _k(s,a). \quad (5)$$

The function $_k(s,a)$ is the degree of membership of state *s* to aggregate fuzzy state *k* with respect action *a*. Expressing aggregate fuzzy states *k* as an n-tuple combination of subsets, φ, from each of the fuzzy states $f \in F$ where $|F| = n$, $k = \{\varphi_1, \varphi_2, \ldots, \varphi_f, \ldots, \varphi_n\}$, the aggregate membership function is given by

$$_k(s,a) = \frac{1}{n} \sum_{\varphi_f \in k} m(f, \varphi_f), \quad (6)$$

Where $m(f, \varphi_f, s, a)$ is a look-up function that returns the fractional membership of state *s* in subset φ_f of fuzzy set *f* for some considered action *a*.

Replacing $\Delta r_t Q(s_t,a,r_t)$ from Equation 3 with $_k(s,a)$ the rule for updating Q-values for the aggregate fuzzy states becomes:

$$q(t) = q(t) + \alpha\delta_t\ _k(s,a), \forall_k \in K \quad (7)$$

III. Experimental Setup

During the game, the Ms. Pac-Man agent has three different choices to make in any given situation. It can go toward the nearest pill, go towards the nearest power pill, or run away from the closest ghost. In a situation where there are no power pills remaining, the power pill choice will no longer be an option. The agent will make its choice based on the current situation and what it has learned about this situation in the past. The agent's knowledge base is stored in three tables: a pill table, a power pill table, and a ghost table. These tables contain a set of Mu values for each of the 27 situations. The situations are defined in terms of Low, Medium and High subsets for distance to nearest pill, distance to nearest power pill and distance to nearest ghost.

TABLE 2
SITUATIONS FOR MS PACMAN AGENT USING FQL

Situation	Pill	Power Pill	Ghost
1	Low	Low	Low
2	Low	Low	Medium
3	Low	Low	High
4	Low	Medium	Low
5	Low	Medium	Medium
6	Low	Medium	High
7	Low	High	Low
8	Low	High	Medium
9	Low	High	High
10	Medium	Low	Low
11	Medium	Low	Medium
12	Medium	Low	High
13	Medium	Medium	Low
14	Medium	Medium	Medium
15	Medium	Medium	High
16	Medium	High	Low
17	Medium	High	Medium
18	Medium	High	High
19	High	Low	Low
20	High	Low	Medium
21	High	Low	High
22	High	Medium	Low
23	High	Medium	Medium
24	High	Medium	High
25	High	High	Low
26	High	High	Medium
27	High	High	High

A. Hardcoded Map

Levels 1 and 2 of the Ms. Pac-Man game use the same pink map. At each intersection on the map, the Ms. Pac-

Man agent can decide to go up, down, left or right. Knowledge of this map is very helpful when deciding the agent's movement options. The map is represented by a file of ones and zeros, where zeros represent empty, navigable space and ones represent walls. This file was created by taking a bitmap screen capture of the game, filtering out the appropriate RGB values for Ms. Pac-Man, ghost, pills and power pills.

The Ms. Pac-Man agent can reference the map array to determine where it can legally move. When the agent decides to run away from the nearest ghost, it will examine the map array to determine which directions it can actually choose. Initially, a similar strategy was used for pursuing pills and power pills, but that was quickly proved ineffective and was discarded. For pills and power pills, a vertex map is used instead.

B. Vertex Map

As in [16], we decided to create a vertex map by examining a screen capture of the game and recording each the location of each pill and power pill. Each of these locations became a vertex of the game graph. Edges were created between pills or power pills that were next to each other on the screen capture. Then vertices were hardcoded into the intersections of the graph where there were no pills or power pills. These special intersections are in the middle of the game board surrounding the ghost's nest. Edges were then hardcoded to connect these artificial vertices to the other vertices already in the graph. Finally, edges were hardcoded in between the two vertices on each end of the warps, or shortcuts, on the sides of the game board.

The Floyd-Warshall algorithm [17] was then used to find a two-dimensional table that can be used to find the shortest path between any two vertices. When the Ms. Pac-Man agent decides to go after the nearest pill or power pill, it first determines the vertex that corresponds to its current position and the vertex that corresponds to the pill's position. Finally, it references the table created by the Floyd-Warshall algorithm to find the first (of perhaps many) direction choices. It continues to make moves and look up the table until it reaches its goal. This allows it to take the shortest path to the goal.

C. Learning Mode

The Ms. Pac-Man agent has two modes: learning and testing. In the learning mode, the agent will play many games, randomly making one of three possible decisions every time it receives a new screen capture. It will record the move it makes (a move consists of the situation-action pair) in a rolling window that tracks the most recent n moves. The value n varied throughout the various experimental rounds and will be discussed later.

When the agent dies, the Mu value for the moves in the rolling window will change. For example, if the agent dies and one of the moves in the rolling window (and, therefore, one of the moves that contributed to her death) consisted of situation 4, {pill-low, power pill-medium, ghost-low}, and the action taken was to move towards the nearest power pill, the agent would decrease the Mu value (negative reward) corresponding to situation 4 in the power pill table. The Ms. Pac-Man agent will run in a game loop, continually decrementing the Mu values when it dies with the intent that poor decision-making in a given situation will be reflected in the Mu values after many rounds of training. We used a learning rate of 0.9 and a discount factor of 0.3 for these experiments.

D. Testing Mode

In the testing mode, the Ms. Pac-Man agent will read the Mu tables for pills, power pills and ghosts that have been generated during the learning mode. After every screen capture, the agent will analyze the situation and check the current Mu values for that situation in each of the three Mu tables. If the agent has learned properly, the table that has the highest Mu value should, theoretically, be the best choice to make. Thus, the agent will make this choice and perform the associated action. As with the Ms. Pac-Man competitions, we will take the average of three runs to record a test score.

IV. RESULTS

Throughout the course of the experiment, many factors were altered. The values that determined the division between Low, Medium, and High for the distance between the Ms. Pac-Man agent and the pills, power pills or ghost was frequently changed in search for the best parameters. In addition, the size of the rolling window that stores the most recent moves was also changed. After considerable testing, we determined that the agent was not learning well because there was no persistence of action. In other words, because she looked up the table after each screen capture, she frequently vacillated between two different choices (a different choice with each screen capture). Therefore, we experimented with various ways to include persistence of action: the agent could make one choice (pill, power pill or ghost) for a given amount of time, forcing the agent to persist with the chosen action until the situation changed, or persisting with the chosen action until the ghosts' situation changed. In other experimental runs, rewards and punishment were both included to change the Mu values, thereby increasing values in the Mu tables when points were earned as well as decreasing values for situation-action pairs that resulted in the agent's death.

In all cases, if there is a tie between Mu values between grazing pills, pursuing power pills or avoiding ghosts, avoiding ghosts will always be the first option because it is the most conservative move. If there is a tie between grazing pills and pursuing power pills, grazing pills is

preferable because it is more benign and will not impact the game play very much. Likewise, when there is a tie between grazing pills, pursuing power pills or eating edible ghosts, eating ghosts will always be chosen because it will have the greatest advantage to the overall score.

A. *No Persistence, No Rewards*

In the first set of experiments, after training for several days with no persistence and no rewards, the Ms. Pac-Man agent averaged between 2000 and 2400 points based on the parameters on the machine. The first machine used pill, power pill and ghost ranges with very low distance values, measured in pixels, for the divisions between low, medium and high. On the second machine, we used more moderate values for distance divisions. Both of these machines used a rolling window size of 10. We used very high values on the third machine for the distances to all objects, while the fourth machine used very low values for the pills and very high values for the ghost. This led to the ghosts being more frequently classified as low or medium, while the pills were typically classified as medium or high. Machines 3 and 4 used a rolling window size of 15. Ultimately all four machines performed relatively equally, so no definitive advantages or disadvantages could be determined for either the distance divisions or the rolling window size.

Machine		Low	Med	High	Window	Score
1		20	60	100	10	2140
2		50	100	150	10	2310
3		100	150	200	15	2150
4	pill	20	60	100	15	2410
	ghost	100	150	200		

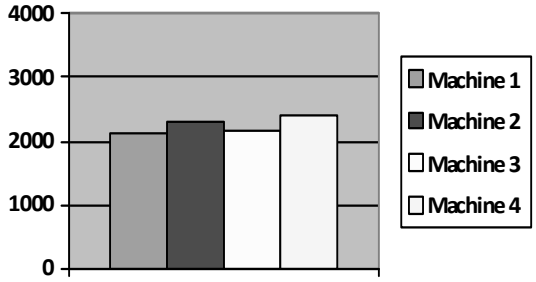

Figure 3. Results for No Persistence, No Rewards

B. *Persistence, No Rewards*

Because we noticed the frequent redirection of the Ms. Pac-Man agent during the testing because she changed her mind during each screen capture, we decided to add persistence of action to both the agent's learning and testing modes. All four test machines were set for the conditions of the best run in the previous set, i.e. we used very low values for the pills and very high values for the ghost and a window size of 15. The first machine always persisted with a minimum of 10 consecutive moves. The second machine persisted for a minimum of 15 consecutive moves. The third machine persisted with a chosen move until the total situation changed, while the fourth machine only persisted until the ghost's situation changed. The results make it evident that it is more effective to persist with a chosen move for a certain amount of time or number of moves than to simply persist with a chosen move only until the situation changed.

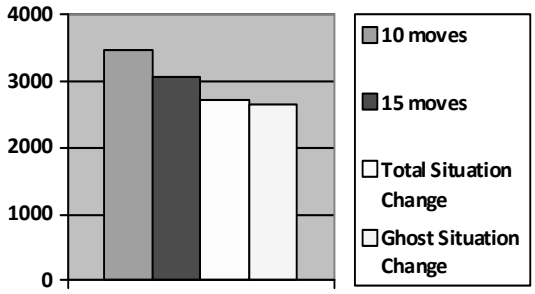

Figure 4. Results for Persistence, No Rewards

The following Mu table is the result of the best trial of the sequence, which reflects a window size of 15. Note that all values have been rounded to the nearest integer value for publication.

Near Pill	Near PPill	Near Ghost	GoTo Pill	GoTo Ppill	Avoid Ghost
Low	Low	Low	13	40	38
Low	Low	Med	50	50	50
Low	Low	High	50	50	49
Low	Med	Low	-13	17	31
Low	Med	Med	50	50	50
Low	Med	High	50	50	50
Low	High	Low	50	38	49
Low	High	Med	50	50	50
Low	High	High	50	50	50
Med	Low	Low	-176	-221	-65
Med	Low	Med	50	49	50
Med	Low	High	50	50	49
Med	Med	Low	-97	-161	-13
Med	Med	Med	50	50	50
Med	Med	High	49	50	50
Med	High	Low	22	-8	23
Med	High	Med	-66	-98	-5
Med	High	High	50	50	50
High	Low	Low	-19	-44	8
High	Low	Med	49	49	49
High	Low	High	47	47	47
High	Med	Low	33	15	36
High	Med	Med	48	48	49
High	Med	High	50	47	50
High	High	Low	22	49	42
High	High	Med	48	48	48
High	High	High	50	50	50

Let's make an intuitive interpretation of the Mu tables for several specific situations: pill close, power pill close and ghost far away; ghost close in any situation; and power pill close and ghost close. First, when the pill is close, the power pill is close and the ghost is far away (L,L,H), there is a tie between the pill and the power pill. Because we imposed the rule of selecting a pill in this situation, the agent will ignore the power pill for now and graze on the nearest pill. Of course, when the pills that are close by are

consumed, we have a situation where pills are further away, the power pill is close, and the ghost is far away (M,L,H or H,L,H). Depending on the ties, the agent will choose to either move toward the pill or avoid the ghost and save the power pill for later - ideally, for when the ghost is close by, thereby getting a higher payoff for eating the edible ghost.

The second set of situations includes any time the ghost is close. There are nine cases where the ghost can be close. Of these, two situation-action pairs (L,L,L and H,H,L) favor going towards a power pill, while six others favor avoiding the ghost. Only a single situation-action pair indicates that the agent should graze on the nearest pill. This situation (L,H.L) has the default value for the pill, which is higher than either of the other two. Perhaps it never came up during training followed by the death of the agent. If it had, it would probably be decremented leading to results that would favor avoiding the ghost.

Finally, if we look closer at the three situations where the power pill is close and the ghost is close. A good strategy for acquiring high points is to wait until the ghost is close before consuming the power pill, which will render the ghost edible. The first situation (L,L,L) favors eating a power pill while the other two (M,L,L and H,L,L) still favor avoiding the ghost. Therefore, we decided to arrange the learning environment's rewards and punishments so the agent will be able to develop this potential strategy.

C. *Persistence and Rewards*

During the final set of experiments, we added positive rewards to the training mode. Until now, only three game objects were considered in the decision-making process: pills, power pills and ghosts. Decisions were made to go after pills or power pills or to avoid ghosts based on Mu values determined by punishment alone. Punishment was imposed when bad decisions led to the agent's death. We now include edible ghosts in an attempt to reward decisions that can lead to higher scores. Note that the same 27 situations are used, but the agent will need to differentiate between pursing an edible ghost (moving toward the nearest ghost) or avoiding the nearest ghost depending on whether ghosts are currently edible.

All four machines were set as before and used the window size of 15 and the 10-move persistence algorithm that proved most effective in the second round of experiments. We now varied the values for *r* for the condition when a pill, power pill or edible ghost was consumed and points are added to the score. For the first machine, the value for r was set at 0.005 times the value of the object consumed. For example, for example, each pill is worth 10 points so the value of r would be set to 0.05 when calculating the new Mu value for that situation-action pair during training. Likewise, because edible ghosts are worth 200 or more points, the value for r would be set to 10 points when updating the Mu values after successfully eating a ghost. This should make eating edible ghosts much more attractive than without a reward. The second and third machines used r values of 0.01 and 0.02 times the value of the object consumed, while the forth machine did not use any rewards at all and was essentially the same as the first machine in the previous round.

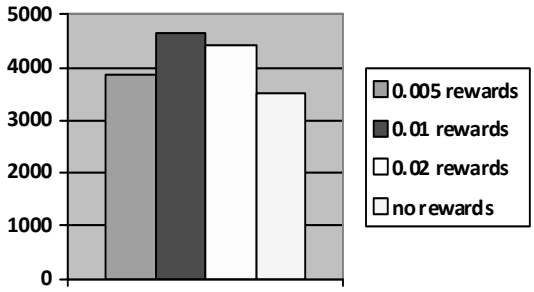

Figure 4. Results for Persistence and Rewards

There seems to be a good balance between the rewards and punishment with the rewards set at 0.01 of the object's value. Having the reward set lower, at 0.005 of the object's value, seems to have the agent favor avoidance behavior over pursuing pills and power pills, while having a the reward set higher, at 0.02 of the object's value, seems to have the agent take more chances when pursuing an edible ghost, even when it is not very close.

The following Mu table is the result of the best trial of the sequence, which reflects a window size of 15 and a reward of 0.01 times the object's value. Note that all values have been rounded to the nearest integer value for publication.

Near Pill	Near PPill	Near Ghost	GoTo Pill	GoTo Ppill	Avoid Ghost	GoTo Ghost
Low	Low	Low	18	41	23	77
Low	Low	Med	53	50	50	50
Low	Low	High	50	50	50	50
Low	Med	Low	19	26	12	62
Low	Med	Med	54	50	50	50
Low	Med	High	58	50	50	50
Low	High	Low	21	31	46	50
Low	High	Med	50	50	50	50
Low	High	High	73	50	50	50
Med	Low	Low	-133	-36	-72	52
Med	Low	Med	50	49	50	50
Med	Low	High	50	50	49	50
Med	Med	Low	-102	-143	-9	50
Med	Med	Med	50	50	50	50
Med	Med	High	46	50	50	50
Med	High	Low	29	-12	34	50
Med	High	Med	-52	-111	7	50
Med	High	High	50	50	50	50
High	Low	Low	-34	-62	-3	50
High	Low	Med	49	49	49	54
High	Low	High	47	47	47	50
High	Med	Low	33	15	36	50
High	Med	Med	41	47	49	50
High	Med	High	50	45	50	50
High	High	Low	37	49	40	50
High	High	Med	48	48	48	50
High	High	High	50	50	50	50

Once again, we can look at the three situations that are of interest when the ghosts become edible: pill close, power pill close and ghost far away; ghost close in any situation; and power pill close and ghost close. When the pill and power pill are close and the ghost is far away, the agent should learn that there is not much benefit to using the power pill just yet. Unlike this situation in the earlier round, this one was never encountered during training, so the defaults will be used. When this situation arises, the agent will avoid the ghost (even though it is far away). Obviously, this won't harm the agent, but it also won't waste the power pill, so it seems like an acceptable result of the learning. We expect that if a different set of training situations had occurred, there would be at least one pill successfully consumed and this value would be above 50. It is also important to recall that pills must be much closer than the ghosts to be considered a low distance.

The second set of situations we would like to examine is any case where the ghost is close. This situation came up only a couple of times during training, with mixed results. If an edible ghost was consumed during training, the Mu value for edible ghosts would increase. Likewise, if an edible ghost is consumed within 10 moves of the agent's death, all situation-action pairs leading up to that death will be modified. It looks like (L,L,L), (L,M,L), (M,L,L) and (H,H,L) are all situations where the agent learned to go towards a power pill when the distance to the nearest ghost is low, allowing the agent to quickly consume the edible ghost and increasing the score. It is also possible that this strategy led to the death of the agent several times because the Mu values for the edible ghost are much lower than expected in these situations. In contrast, the situations of (L,H,L), (M,M,L), (M,H,L), (H,L,L), (H,M,L), where the distance to the nearest ghost is also low, had Mu values that caused the agent to avoid the ghost, which is also acceptable behavior but does not result in very high scores.

Finally, if we take a closer look at the specific situation where the power pill is close and the ghost is close, the Mu value seems to indicate that this led much more successful consumption of edible ghosts. Although it was not tracked, it is possible that even more than one ghost was consumed during the same period. This is a common strategy to significantly increase the overall score and was successfully encouraged during the learning process.

V. CONCLUSIONS AND FUTURE WORK

Fuzzy Q-learning has proven to be an effective method for enabling a Ms. Pac-Man agent to learn how to play the game. There are, however, some significant shortcomings in our approach that can be addressed in future research.

First, our learning approach only allowed the agent to learn for a relatively short time. While this time would be more than adequate for a human player, our software agent may benefit from many more games. Some situations never came up and therefore have the default Mu value of 50, rather than changing the situation-action pair appropriately. In most cases, this had little or no impact, either positive or negative, on the outcome of the game. We could also start with trained agent, but allow for random choices every once in a while during regular game play to simulate a human player experimenting with new strategies.

Second, our pink game map is only good for the first two levels. Therefore, when the agent successfully makes it to the third level, which uses the light blue maze, it quickly dies because the move choices do not match the third level maze. This map is used for levels three, four and five. We plan to add this map and the resulting vertex graph for these additional levels. We plan to also include a mechanism for recognizing the additional treats that show up randomly in each level. Consuming these treats, especially in later levels, can be worth thousands of additional points.

Third, the window size of 15 means that the program holds 15 situation-action pairs that lead to either the agent's death for punishment or lead to the successful consumption of an object for a reward. We expect that it will be more meaningful and a better representation to prorate the reward or punishment, such as 1/n, so those moves that are more proximate to the triggering action are rewarded or punished more.

Finally, pill density or ghost density may be a good addition to the state vector. According to the influence map model for playing Ms. Pac-Man [18], ideal strategies for the game of Ms. Pac-Man include going towards patches of a high density of pills or to go towards the power pill when there is a high density of ghosts close by.

The goal of this work was to develop an agent that was able to learn in a nondeterministic environment. Gains in this area can obviously be applied to other computer games, but may also be applicable to solve real-world problems. For example, this type of algorithm can be applied in a simulated environment for autonomous vehicles that are required to rally at several defined areas on a map, which may have varying strategic values, while avoiding potentially lethal obstacles. To some, this is just research on how to beat a archaic game, but to others it is an attempt to solve a very important challenge.

REFERENCES

[1] K. Uston, *Mastering Ms. Pac-Man*. Macdonald and Co. 1981.

[2] S. Lucas, "*Ms. Pac-Man Competition*," Available at http://dces.essex.ac.uk/staff/sml/pacman/CEC2009Results.html (accessed 6/12/09).

[3] A. Fitzgerald, P. Kemeraitis, and C. B. Congdon, "RAMP: A Rule-Based Agent for Ms. Pac-Man," in *Proc. IEEE World Congress in Computational Intelligence (WCCI2008),* Hong Kong, 2008.

[4] "*Ms. Pac-Man Game*," Available at http://www.webpacman.com/mspacman.htm (accessed 6/13/09).

[5] L. A. Zadeh. Fuzzy sets. *Journal of Information and Control*, 8:338-353, 1965.
[6] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Reinforcement Learning with Soft State Aggregation," *Advances in Neural Information Processing*, 7:361-368, 1994.
[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[8] C. J. Watkins and P. Dayan. "Q-Learning," *Machine Learning*, 8:279-292, 1992.
[9] H. Berenji and D. Vengerov, "Advantages of Cooperation between Reinforcement Learning Agents in Difficult Stochastic Problems," In *Proc. IEEE International Conference on Fuzzy Systems*, pages 871-876, 2000.
[10] H. R. Berenji and D. Vengerov. "Cooperation and coordination between Fuzzy Reinforcement Learning Agents in Continuous State Partially Observable Markov Decision Processes," In *Proc. IEEE International Fuzzy Systems Conference*, pages 621-627, 1999.
[11] D. Gu and H. Hu, "Reinforcement Learning of Fuzzy Logic Controllers for Quadruped Walking Robots," In *Proc. 15th IFAC World Congress*, 2002.
[12] D. Gu and H. Hu, "Learning Fuzzy Logic Controller for Reactive Robot Behaviours," In *Proc. International Conference on Advanced Intelligent Mechatronics*, 2003.
[13] J. Ammerlaan and D. Wright, "Adaptive Cooperative Fuzzy Logic Controller," In Proc *ACS Conferences in Research and Practice in Information Technology (CRPIT)*, 2004.
[14] T. Nakashima, M. Udo, and H. Ishibuchi, "Acquiring the Positioning Skill in a Soccer Game using Fuzzy Q-learning," In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 1488-1491, 2003.
[15] D. C.Wardell and G. L. Peterson, "Fuzzy State Aggregation and Policy Hill Climbing for Stochastic Environments," *International Journal of Computational Intelligence and Applications*, 6(3):413-428, 2006.
[16] H. Handa, "Constitution of Ms. Pac-Man Player with Critical-Situation Learning Mechanism", In *Proc. 4th International Workshop on Computational Intelligence and Applications*, Hiroshima, Japan, 2008.
[17] *Data Structures and Algorithms in Java*, John Wiley and Sons, 2006.
[18] N. Wirth and M. Gallagher, "An Influence Map Model for Playing Ms. Pac-Man," In *Proc of IEEE Symposium on Computational Intelligence and Games (CIG'08)*, Perth, Australia, 2008.