Clever Pac-man

Nicole Alicia ROSSINI, Christian QUADRI and N. Alberto BORGHESE
Applied Intelligent Systems Laboratory (AIS-Lab)
Department of Computer Science
University of Milano
{nicole.alicia.rossini, christian.quadri}@studenti.unimi.it, alberto.borghese@unimi.it

Abstract. In this paper we show how combining fuzzy sets and reinforcement learning a winning agent can be created for the popular Pac-man game. Key elements are the classification of the state into a few fuzzy classes that makes the problem manageable. Pac-man policy is defined in terms of fuzzy actions that are defuzzified to produce the actual Pac-man move. A few heuristics allow making the Pac-man strategy very similar to the Human one. Ghosts agents, on their side, are endowed also with fuzzy behavior inspired by original design strategy. Performance of this Pac-man is shown to be superior to those of other AI-based Pac-man described in the literature.

Keywords. Fuzzy Q-learning, game engine, minimum path, artificial intelligence, agent.

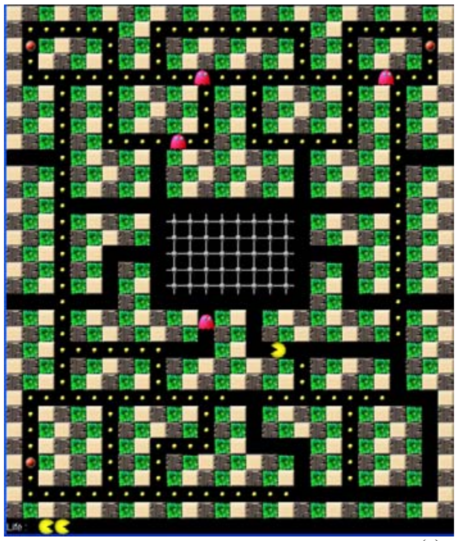
Introduction

Pac-man is one of the most successful computer games. It was originally produced by Namco and distributed by Midway, starting in May 1980 and has become since then a classic of the field [1].

Pac-man is an agent that moves in a maze constituted of corridors paved with pills. Aim of the agent is to eat all these pills (each pill eaten is worth 10 points); when this happens the game proceeds to the next game level. Also some enemies, that have the shape of ghosts, are present inside the maze; these can eat the Pac-man. Every time a Pac-man is eaten a new Pac-man is generated up to a maximum of three times. Four special pills, called power pills, are also present in the maze. These special pills enable the Pac-man to eat the ghosts, but their effect lasts for a limited amount of time; therefore, during this period, ghosts tend to run away from the Pac-man. As each eaten ghost is worth 200 points (the first one), 400 points (the second one), 600 points (the third) and 800 points (the fourth), the strategy of going for ghosts is an efficient strategy to get many points quickly.

As many other computer games, Pac-man has been widely used in the literature to study techniques for optimal decision and learning in game theory and machine learning. The challenge is due to the fact that the environment seen by Pac-man is non deterministic as no knowledge on the ghosts behavior is provided to Pac-man. The same is true for the ghosts that do not know in advance which can be the best policy to haunt the Pac-man. A further issue is the problem complexity as the number of cells can be very large: the maze can be constituted by some hundreds of cells that can be occupied by pills, power pills, ghosts and the Pac-man (cf. Fig. 1). This calls for a smart definition of the states that cannot be associated to single maze positions.

In this paper we explore the possibility of recently proposed Fuzzy Q-learning [2] to learn the optimal behavior of the Pac-man. A fuzzy approach has been adopted for the states definition to make the problem manageable. Ghosts behavior has also been completely modeled with fuzzy controllers. Results show that on average the Pac-man achieves over 6000 points that are more than the 4300 points gained in the Pac-man described in [3], using the same game map of the original Ms. Pac-man game.



(a)

Fuzzy aggre gated state	Closest ghost	Closest pill	Closest power pill
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

(b)

Figure 1. In panel (a) the game layout map used, equal to the original one. In panel (b) the states on which the Pac-man defines its policy.

1. Method

The gameplay requires the definition of the behavior of the ghosts, while the artificial engine will shape the behavior of the Pac-man.

1.1 The ghosts

The classical implementation of Pac-man assumes that the four ghosts have different personalities [4] and therefore different behaviors. As stated by Toru Iwatani: “INTERVIEWER: *What was the most difficult part of designing the game?* - IWATANI: *The algorithm for the four ghosts who are the enemies of the Pac Man* *getting all the movements lined up correctly. It was tricky because the monster movements are quite complex. This is the heart of the game. I wanted each ghostly enemy to have a specific character and its own particular movements, so they weren't all just chasing after Pac Man in single file, which would have been tiresome and flat. One of them, the red one called Blinky, did chase directly after Pac Man. The second ghost is positioned at a point a few dots in front of Pac Man's mouth. That is his position. If Pac Man is in the center then Monster A and Monster B are equidistant from him, but each moves independently, almost sandwiching him. The other ghosts move more at random. That way they get closer to Pac Man in a natural way*".

This leads to different implementations. We have adopted the implementation proposed in [5]: all the ghosts can assume all the three possible behaviors and they decide through a fuzzy controller which of the three behaviors is most adequate to the present situation. Therefore the decision on the actual behavior of a given ghost does not depend on the ghost but on the game situation.

The ghost behavior is different depending if it is attacking the Pac-man or defending from it. At start all the ghosts start at the maze center and, as they have to protect the pills from being eaten by the Pac-man, they tend to distribute along the maze corridors. If a ghost becomes close to the Pac-man, it aims to it, unless the power pill is active (has been just eaten by the Pac-man) in which case the ghost runs away from the Pac-man. The more the game progresses the more the ghosts tend to stop the Pac-man. These qualitative statements are translated into fuzzy controller rules as follows.

We have the following ghosts' behaviors: *shy, random, hunting* and *defense*. In the original implementation the four ghosts have the following behaviors: two move randomly, one actively goes after the Pac-man and the fourth waits for the Pac-man close to the pills.

We have preferred here to optimize the ghosts' behavior according to the game situation. In the *shy behavior*, the ghost gets away from the closest agent. In the default condition the ghosts tend to move as far as possible one from the other, while when the power pill is active they tend to move as far as possible from the Pac-man. To achieve this, the ghost evaluates the new distance from each agent moving in the four orthogonal directions (north, east, south, west), that are allowed (it is not allowed to go through a wall). The direction that produces the maximum increment of the distance is chosen. To avoid stereotyped behaviors, when ties are present, the four directions are explored in random order and the agent always chooses the first best direction.

The *random behavior* is the same of the original game: at each time step, the ghost moves towards an adjacent admissible cell, at random.

In the *hunting behavior*, the ghost goes for the Pac-man, to eat it. To this aim, the ghost chooses always the minimum path to the Pac-man, that is pre-computed at game loading using the Floyd-Warshall algorithm [7] creating a look-up table. With this behavior all the ghosts get aligned and follow the Pac-man from behind. To break this, one of the ghosts has to move in a suboptimal (random) direction in one step, and eventually surround the Pac-man. We remark here that the evaluation of the best direction has to be repeated at each step, as the Pac-man is moving in a way that is not known to the ghosts.

The actual action in the defence behavior, also present in the original version, depends on the pills density left. In fact to complete the game the Pac-man has to eat all the pills and when few pills are left, the ghosts know that the Pac-man has to go for them. Therefore in the *defense behavior* the ghost goes in the area with the maximum pills density and waits that the Pac-man moves towards this area, instead of actively chasing the Pa-man. In the implementation the maze is subdivided into nine partially overlapped areas and the Pac-man moves towards the central pill of the chosen area. The areas have limits on both axes respectively at {0, ½; ¼ ¾ and ½ 1}. When the ghost is in the central pill he always moves to a random adjacent cell.

The ghosts choose their actual behavior as a function of their actual state according to a fuzzy policy. The variables input to the fuzzy controller associated to each ghost are: the distance between the ghost and the Pac-man, the distance with the nearest other ghost, the frequency with which the Pac-man eats the pills and the lifetime of the Pac-man that represents the Pac-man ability. For each of these variables three fuzzy classes are defined. An additional Boolean variable, not fuzzy, is defined that states if the ghost can be eaten by the Pac-man and it is asserted for the time span in which the effect of the power pill lasts.

A set of fuzzy rules are defined inspired to [4], like for instance:

- *If pacman_near AND skill_good, Then hunting_behavior*
- *If pacman_near AND skill_med AND pill_med, Then hunting_behavior*
- *If pacman_near AND skill_med AND pill_far, Then hunting_behavior*
- *If pacman_med AND skill_good AND pill_far, Then hunting_behavior*
- *If pacman_med AND skill_med AND pill_far, Then hunting_behavior*
- *If pacman_far AND skill_good AND pill_far, Then hunting_behavior*

The rules activated are then defuzzyfied, choosing the behavior that is associated to the rule with maximum fitness [8]. This, in turns, induces the ghost to move in one of the possible directions. The definition of the classes boundary was carried out such that the ghosts have as preferred action "hunting", that is chose four times more frequently than the other actions.

1.2 The Pac-man

In this work the Fuzzy Q-learning algorithm developed for deterministic frameworks [2] has been adapted to the non-deterministic setting considered here. The need of using a fuzzy description of the state is mandatory here to avoid the explosion in the number of states that would make the computation infeasible. At the basis of Fuzzy Q-learning is the aggregation of different states into the same fuzzy class, the generation of a policy and its evaluation as a function of the value assumed by the fuzzy classes. To this aim, the analysis is not carried out on the single maze positions but on three fuzzy variables that assume the values low, medium high and represent the minimum distance from the closest pill, the minimum distance from the closest power pill and the minimum distance from the closest ghost. With this discretization we obtain 27 fuzzy classes that represent the different states of the Pac-man and are shown in Fig. 1c.

1.3 Q-learning

In reinforcement learning (RL) [6], an agent has to interact with an environment that can be described through a state variable, *s*. To this aim, the agent has to learn a policy (1a). The agent has to learn the policy that allows the best reward in the long term. The reward is represented by a discounted sum of the reward obtained at each step of interaction with the environment, $r(s_t, a_t, s_{t+1})$ and it is represented by a value function, Q(s,a).

$a_t = \pi(s_t)$	Policy	(1a)
$r_t = r(s_t, a_t, s_{t+1})$	Reward	(1b)
$Q_t = Q(s_t, a_t)$	Value function	(1c)

$$Q(s_t,a_t)' = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a_{'}) - Q(s_t,a_t)] \quad (2)$$

RL can be well cast to the Pac-man game where the agent has to learn how to earn the maximum score. Among the most successful algorithms of this family is *Q-learning* [9]. In this case, the agent learns $Q(s_t,a_t)$ from the actual action, a_t, chosen in the current state, s_t with the current policy, and at the same time improves its policy to obtain a larger long-term reward. $Q(s_{t+1},a_t)$ is maximized with respect to the current action, a'. r_{t+1} represents the reward obtained in the transition from state s_t to state s_{t+1}, α and γ represent respectively the learning rate and the discount rate of the reward.

In the present state, we distinguish between the actual state of the Pac-man, s, represented by the distance of the closest ghost, of the closest pill and of the closest power pill that can assume continuous values and the fuzzy aggregated states, q, that can assume one of the 27th defined situations presented in Fig. 1b.

Each of the three input variables has a membership function associated to it, and the degree of membership to one of the classes is computed here according to the average method. Given $s^* = \{c_1, c_2, c_3\}$ the actual input state and $m(c_i)$ the degree of membership of c_i to one of the fuzzy input classes, and f_i the current fuzzy aggregated state analized, the degree of membership of s* to the fuzzy aggregated class, f_i will be:

$$\mu(f_t) = \frac{\sum_{i=1}^{3} m(c_i)}{3} \quad (3)$$

Given the fuzzy nature of the input classes, more than one fuzzy aggregated state can be active at the same time. Therefore, the actual Q value of the current state, s*, can be computed summing over all the active fuzzy aggregated states as:

$$Q(s^*,a) = \frac{1}{n}\sum_{i=1}^{n} \mu(f_i)\, q(f_i,a) \quad (4)$$

where the $\mu(f_i)$ add to one. The Q function for each aggregated state active, q(.) is update according to Q-learning [6] as:

$$q(f,a) = q(f,a) + \alpha_{f,a}\left[r + \gamma \max_{a'} Q(s',a') - q(f,a)\right] \quad (5)$$

Eq. (5) is applied to all aggregated states f that are active in one transition step. The value of $\alpha_{f,a}$ is chosen according to:

$$\alpha_{f,a} = \frac{1}{\sum_{\tau=0}^{\tau=t-1} \mu\left(f_{jt}\left(s_t, a_t\right)\right)} \tag{6}$$

that is a natural extension to fuzzy states of the computation of the running average [6].

The action available to Pac-man are: *Go to Pill, Go to Power Pill, Avoid Ghost and Go to Ghost*, of which the last two are mutually exclusive: avoid ghost is available when Power Pill is not active while Go to Ghost when it is active. As in any time step there can be both edible and hunting Ghosts (those that have been eaten and has born again), the distance is measured only from the closest ghost, independently on its attitude.

The actual success of the Pac-man depends on the implementation of these four actions, as different implementation can produce very different behaviors. Hereafter is a description of the implementation chosen here. When the Pac-man decides to *Go to pill*, he always goes to the closest pill, independently on the position of the ghosts. To achieve this, the look-up table containing the minimum path to goal is addressed. To achieve this, the distance from all the pills is examined and the pill towards which to move has to be decided randomly among all the pills that are closest. Otherwise, the Pac-man would move always towards the pills in the upper west region of the maze. The same strategy is used for the action *Go to Power Pill*. In the *Go to ghost* action, the Pac-man goes run always after the closest ghost.

Avoid Ghost is the most critical action. If we move the Pac-Man to increase the distance from the closest ghost, at the maze crossings, where the turning choice would produce an increased distance, easily the Pac-man can choose the direction that leads it directly to another ghost coming from that direction. A possible alternative is to choose the direction that maximizes a weighted distance from all the ghost. Although this second solution is more efficient for the life length of the Pac-man it induces the Pac-man to move inside a small area at the corners of the maze. A possible solution is to consider a weighted distance only inside a given area around the actual state. In this situation the Pac-man is forced at the corner of such an area that changes at every step. The last improvement is related to the situation in which multiple directions allow increasing the distance from the ghosts. In this situation the Pac-man will choose the direction that leads it towards the closest power-pill, if still present in the maze, otherwise the closest "normal" pill is chosen.

2. Results and Discussion

2.1 Implementation notes

We have implemented the basic version of the game: the only bonus is represented by the Power Pill. We have not considered the fruit bonus, that may show for a brief time inside the maze and can change the behavior of the ghosts, according to [3]. Moreover, a single map has been used, but any other map associated to the different levels can be easily loaded.

A few heuristics have been implemented to make the behavior pleasant. The first is *persistence* [3]. The policy learnt by the Pac-man often forced it to change action at each step, resulting into an unstable behavior and of little motion of the Pac-man. We have avoided this forcing the Pac-man to follow the same policy for *n* consecutive steps, with *n* set to five here. This choice introduces a criticality when there is a brisk change in the situation, for instance because the Power pill effect ends: as a consequence a ghost can get very close to the Pac-man. In such critical situation (when a ghost is as close as to positions) persistence is cancelled and the actual optimal action of the Pac-man is issued. Another heuristic that has been added is *taboo*, that requires the Pac-man not to issue the same move that would bring it in the same position of state t-1. This is because, in some situations, the Pac-man was starting oscillating between two closest states.

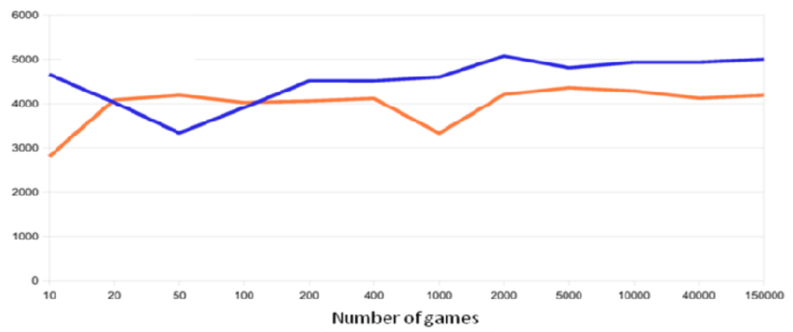


Figure 2. Total points collected with the number of games, the red line is associated to low, medium, high distance classes centered in {6, 18, 30} while the blue one in {5, 12, 25}.

2.2 Results and Discussion

To allow an effective learning of the policy, the parameters have to be carefully set. The (negative) reward for the death of a Pac-man was set experimentally to -1000. A much smaller value did not sufficiently compensate the positive rewards gained, while a more negative value would produce a depressed Pac-man that became too shy and largely preferred the action Avoid Ghost, choosing implicitly to survive rather than gaining points.

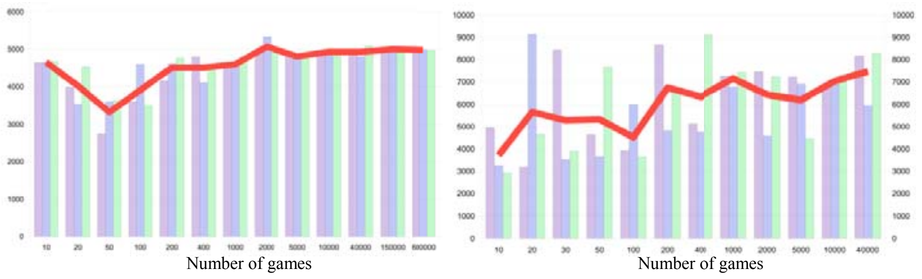


Figure 3. The score reached as games progressed for thee different trials. In read the average value. On the left a random policy is adopted to learn Q(.), on the right an -greedy policy has been implemented.

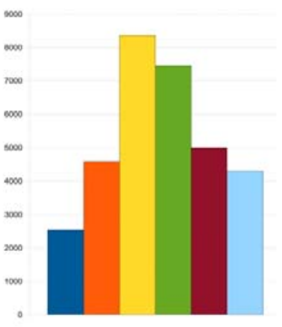
Another important element is the definition of the boundaries of the fuzzy classes. Although several algorithms have been proposed to determine the optimal class boundaries for classification purposes [10], these cannot be extended easily to control and we have adopted here an experimental approach, evaluating the Q(.) function associated to a greedy agent playing the game. We could explore few combination of classes boundaries, but these have suggested classes that allowed very good results; in particular we have chosen d = 5 as maximum membership for *low* distance, d = 12 as maximum membership for *medium* distance and d = 25 as maximum membership for *high* distance. Other combinations of distance led to worse results as shown in Fig. 2.

The reward associated to each pill has also been investigated, but it is shown that there is no large difference in the results for a wide range of results as shown by the score difference when r = 1 (the score associated to each pill) and r = 0,1 (Fig. 3a).

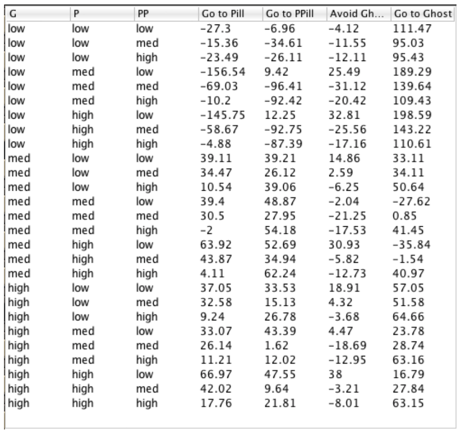
The effect of the greediness of the policy has also been investigated. If a random policy was chosen, the improvement resulted very slow but consistent as shown in Fig. 3a, where the average of three different set of trials is reported. In this situation, the function Q(.) is learnt by moving randomly inside the maze. When using an ε-greedy policy, results are variable, depending if the random initial choices were in the correct or wrong direction. In the second case, learning became slower, while in the first one a score as large as 9000 points was reached after only 20 games (Fig. 3b).

To avoid such variability obtaining at the same time a consistent improvement, as initialization of the Q(.) function according to Optimistic initial value [6] has been implemented. This has allowed to achieve results that were superior to those reported in [4]. Scores obtained in the different situations are summarized in Figure 4a. The Q function well represents the best actions in each state (Fig. 4b).

The results obtained here show that a fuzzyfication of the state allows making this game manageable and discover an effective gameplay that, given the high number of cells, would have not been computable. On average the Pac-man has obtained more than the 4300 points reported in [4].



(a)



G	P	PP	Go to Pill	Go to PPill	Avoid Gh...	Go to Ghost
low	low	low	-27.3	-6.96	-4.12	111.47
low	low	med	-15.36	-34.61	-11.55	95.03
low	low	high	-23.49	-26.11	-12.11	95.43
low	med	low	-156.54	9.42	25.49	189.29
low	med	med	-69.03	-96.41	-31.12	139.64
low	med	high	-10.2	-92.42	-20.42	109.43
low	high	low	-145.75	12.25	32.81	198.59
low	high	med	-58.67	-92.75	-25.56	143.22
low	high	high	-4.88	-87.39	-17.16	110.61
med	low	low	39.11	39.21	14.86	33.11
med	low	med	34.47	26.12	2.59	34.11
med	low	high	10.54	39.06	-6.25	50.64
med	med	low	39.4	48.87	-2.04	-27.62
med	med	med	30.5	27.95	-21.25	0.85
med	med	high	-2	54.18	-17.53	41.45
med	high	low	63.92	52.69	30.93	-35.84
med	high	med	43.87	34.94	-5.82	-1.54
med	high	high	4.11	62.24	-12.73	40.97
high	low	low	37.05	33.53	18.91	57.05
high	low	med	32.58	15.13	4.32	51.58
high	low	high	9.24	26.78	-3.68	64.66
high	med	low	33.07	43.39	4.47	23.78
high	med	med	26.14	1.62	-18.69	28.74
high	med	high	11.21	12.02	-12.95	63.16
high	high	low	66.97	47.55	38	16.79
high	high	med	42.02	9.64	-3.21	27.84
high	high	high	17.76	21.81	-8.01	63.15

(b)

Figure 4. In panel (a) the average score obtained in different conditions from left to right: blue – random policy, no learning; orange – shy ghost, always escaping the ghosts except when Power Pill is on, no learning; yellow – Pac-man learning with an □-greedy policy (reward factor = 0.1); Pac-man learning with an □-greedy policy (reward factor = 1); random policy, learning, fuzzy boundaries {5. 12. 25}; cyan, random policy, learning, fuzzy boundaries {6, 18, 30}. In panel (b) the Q value of the state-action pairs for the yellow case after 10,000 games.

The fuzzy approach has also allowed a simple but close to the original implementation of the ghosts behavior.

We have always relied the choices to the closest element (ghost, pill, power pill) while a human player relies more on the concept of density or of “average easy to reach”. This more comprehensive analysis of the game status can make the Pac-man even more effective, but would introduce more high level knowledge into the game.

With the same idea in mind, the escape from ghost strategy implemented was quite simple as the Pac-man tries to escape the closest ghost looking forward at one step. For this reason, a few times, the Pac-man is trapped between two ghosts and is eaten. To avoid this a multi-step forward analysis might be more performing, allowing the Pac-man elaborate escaping strategies, for instance through min-max strategies. However this solution would require again to insert knowledge into the game, and has not been further pursued.

3. Conclusion

We have presented here an application of Fuzzy Q-learning to the Pac-man game, where, with the help of a few heuristic the Pac-man can achieve very high scores. Key element is the fuzzyfication of the state space that allows reducing the number of states on which the fuzzy inference engine has to work making the game engine feasible.

Although with such a simple implementation the game played by the Pac-man instructed by Q-learning is extremely similar to that played by a Human player.

References

[1] http://pacman.com/en/

[2] H. Berenji, A reinforcement learning-based architecture for fuzzy logic control, International Journal of Approximate Reasoning 6 (2) (1992) 267–292.

[3] DeLooze, L.L.; Viner, W.R.; "Fuzzy Q-learning in a nondeterministic environment: developing an intelligent Ms. Pac-Man agent", Computational Intelligence and Games, 2009. CIG 2009. pp.162-169, 7-10 Sept. 2009.

[4] Susan Lammers: "Interview with Toru Iwatani, the designer of Pac-Man", Programmers at Work 1986

[5] Shaout,A.; King, B.; Reisner, L.; "Real-Time Game Design of Pac-Man Using Fuzzy Logic", The International Arab Journal of Information Technology, Vol.3 No.4, October 2006

[6] Sutton, R.S.; Barto, A.G. "Reinforcement Learning: An Introduction", MIT Press, 1998.

[7] J. Nocedal, S.J. Wright. *Numerical Optimization*. Seconda Edizione. Ed. Springer, 2006.

[8] B. Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, 1992.

[9] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279-292.

[10] A. Celikyilmaz, I. Burhan Turkşen, R. Aktaş, M. Mete Doganay and N. B. Ceylan, Increasing accuracy of two-class pattern recognition with enhanced fuzzy functions, Expert Systems with applications, 36, Vol. 2(1), pp. 1337-1354, 2009.