Evolving Opponents for Interesting Interactive Computer Games

Georgios N. Yannakakis*
*Centre for Intelligent Systems
and their Applications
The University of Edinburgh
AT, Crichton Street, EH8 9LE
g.yannakakis@sms.ed.ac.uk

John Hallam**
** Mærsk Mc-Kinney Møller Institute
for Production Technology
University of Southern Denmark
Campusvej 55, DK-5230
john@mip.sdu.dk

Abstract

In this paper we introduce experiments on neuro-evolution mechanisms applied to predator/prey multi-character computer games. Our test-bed is a modified version of the well-known Pac-Man game. By viewing the game from the predators' (i.e. opponents') perspective, we attempt off-line to evolve neural-controlled opponents capable of playing effectively against computer-guided fixed strategy players. However, emergent near-optimal behaviors make the game less interesting to play. We therefore discuss the criteria that make a game interesting and, furthermore, we introduce a generic measure of predator/prey computer games' interest. Given this measure, we present an evolutionary mechanism for opponents that keep learning from a player while playing against it (i.e. on-line) and we demonstrate its efficiency and robustness in increasing and maintaining the game's interest. Computer game opponents following this on-line learning approach show high adaptability to changing player strategies which provides evidence for the approach's effectiveness against human players.

1. Introduction

Machine learning in computer games is nowadays still in its very early stages, where computer games continue to use simple rule-based finite and fuzzy-state machines for nearly all their AI needs (Woodcock, 2001). Therefore, we still meet new released games with the same 20-year old concept in brand new graphics engines. Unfortunately, instead of designing intelligent opponents to play against, game developers mainly concentrate on the graphical presentation of the game.

Predator/prey games is a very popular category of computer games and among its best representatives is the classical Pac-Man released by Namco (Japan) in 1980. Even though Pac-Man's basic concept and graphics are very simple, the game still keeps players interested after so many years, and its basic ideas are still found in many new released games. In Pac-Man, the player's (*PacMan's*) goal is to eat all the pellets appearing in a maze-shaped stage while avoiding being killed by four opponent characters named '*Ghosts*'. On the other hand, *Ghosts* are aiming to kill (by 'touching') *PacMan* as soon as possible. The game is over when either all pellets in the stage are eaten by *PacMan* or *Ghosts* manage to kill *PacMan*. In that case, the game restarts from the same initial positions for all five characters. Since there are four *Ghosts* on the game field, they are designed to be slower than *PacMan* so that the game is fairer to play.

There are some examples, in the Pac-Man domain literature, of researchers attempting to teach a controller to drive *PacMan* in order to acquire as many pellets as possible and to avoid being eaten by *Ghosts*. Koza (1992) considers the problem of controlling an agent in a dynamic non-deterministic environment and, therefore, sees Pac-Man as an interesting multi-agent environment for applying off-line learning techniques based on genetic programming. The same Pac-Man application domain has been used for analyzing size and generality issues in genetic programming (Rosca, 1996).

On the other hand, there are many researchers who use predator/prey domains in order to obtain efficient emergent teamwork behavior of either homogeneous or heterogeneous predators. For example, Luke and Spector (1996), among others, have designed an environment similar to the Pac-Man game (i.e. Serengeti world) in order to examine different breeding strategies and coordination mechanisms for the predators. Finally, there are examples of work (see (Haynes and Sen, 1995) and (Miller and Cliff, 1994)) in which both the predators' and the prey's strategies are co-evolved in continuous or grid-based environments.

Similar to Luke and Spector (1996), we view Pac-Man from the *Ghosts'* perspective. Our first aim is to emerge effective teamwork hunting behaviors by the use

of an off-line training approach, based on evolutionary computation techniques, applied to homogeneous neural controlled (Yao, 1999) *Ghosts*. However, playing a computer game like Pac-Man against optimal hunters cannot be interesting because of the fact that you get easily killed. To this end, the primary objective of this work is to introduce an efficient generic measure of interest of any predator/prey game. We believe that the interest of any computer game is directly related to the interest generated by the opponents' behavior rather than to the graphics or even the player's behavior. Thus, when 'interesting game' is mentioned we mainly refer to interesting opponents to play against.

We present a robust on-line neuro-evolution learning mechanism capable of increasing the game's interest (starting from near-optimal off-line trained behaviors) as well as keeping that interest at high levels as long as the game is being played. The arcade version of Pac-Man uses a handful of very simple rules and scripted sequences of actions combined with some random decision-making to make the *Ghosts'* behavior less predictable. The game's interest decreases at the point where *Ghosts* are too fast to beat (Rabin, 2002). In our Pac-Man version we require *Ghosts* to keep learning and constantly adapting to the player's strategy instead of being uninteresting opponents with fixed strategies. In addition, we explore learning procedures that achieve good real-time performance (i.e. low computational effort while playing).

2. The Pac-Man World

The computer game test-bed studied is a modified version of the original Pac-Man computer game released by Namco. The original game includes a number of special pellets called 'power-pills' that allow *PacMan* to 'eat' *Ghosts* for a short period of time. Power-pill is the only feature not included in this modified version of the game. This omission contributes to the simplicity of the game without affecting the game's playing interest.

As previously mentioned, the Pac-Man game is investigated from the viewpoint of *Ghosts* and more specifically how *Ghosts'* emergent adaptive behaviors can contribute to the interest of the game. Pac-Man — as a computer game domain for emerging adaptive behaviors — is a two-dimensional, multi-agent, grid-motion, predator/prey game. The game field (i.e. stage) consists of corridors and walls. Both the game's dimensions and its maze structure are predefined. For the experiments presented in this paper we use the 19×29 grid maze-stage presented in Figure 1.

The characters visualized in the Pac-Man game (as illustrated in Figure 1) are a white circle that represents *PacMan* and 4 ghost-like characters representing the *Ghosts*. Additionally, there are black squares that represent the pellets and dark grey blocks of walls.

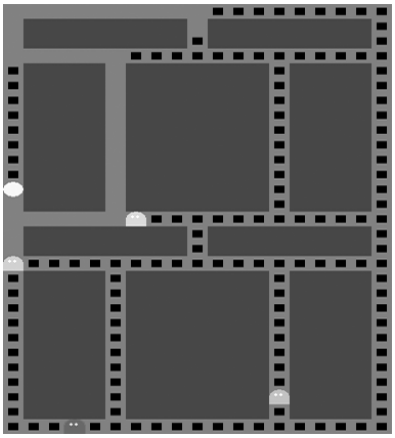


Figure 1: Snapshot of the Pac-Man game

PacMan moves at double the *Ghosts'* speed and since there are no dead ends, it is impossible for a single *Ghost* to complete the task of killing it. Since *PacMan* moves faster than a *Ghost*, the only effective way to kill *PacMan* is for a group of *Ghosts* to hunt cooperatively. It is worth mentioning that one of *Ghosts'* properties is permeability. In other words, two or more *Ghosts* can simultaneously occupy the same cell of the game grid.

The simulation procedure of the Pac-Man game is as follows. *PacMan* and *Ghosts* are placed in the game field (initial positions) so that there is a suitably large distance between them. Then, the following occur at each simulation step:

1. Both *PacMan* and *Ghosts* gather information from their environment.

2. *PacMan* and *Ghosts* take a movement decision every simulation step and every second simulation step respectively. (That is how *PacMan* achieves double the *Ghost's* speed.)

3. If the game is over (i.e. all pellets are eaten, *PacMan* is killed, or the simulation step is greater than a predetermined large number), then a new game starts from the same initial positions.

4. Statistical data such as number of pellets eaten, simulation steps to kill *PacMan* as well as the total *Ghosts'* visits to each cell of the game grid are recorded.

2.1 PacMan

Both the difficulty and, to a lesser degree, the interest of the game are directly affected by the intelligence of the *PacMan* player. We chose three fixed *Ghost*-avoidance and pellet-eating strategies for the *PacMan* player, differing in complexity and effectiveness. Each strategy is based on decision making applying a cost or probability

approximation to the player's 4 neighbor cells (i.e. up, down, left and right). Even though the initial positions are constant, the non-deterministic motion of *PacMan* provides lots of diversity within games.

- Cost-Based (CB) *PacMan*: The CB *PacMan* moves towards its neighbor cell of minimal cost. Cell costs are assigned as follows: $c_p = 0$, $c_e = 10$, $c_{ng} = 50$, $c_g = 100$, where c_p: cost of a cell with a pellet (pellet cell); c_e: cost of an empty cell; c_g: cost of a cell occupied by a *Ghost* (*Ghost* cell); c_{ng}: cost of a *Ghost's* 4 neighbor cells. Wall cells are not assigned any cost and are ignored by *PacMan*. In case of equal minimal neighbor cell costs (e.g. two neighbor cells with pellets), the CB *PacMan* makes a random decision with equal probabilities among these cells.

  In other words, the CB *PacMan* moves towards a cost minimization path that produces effective *Ghost*-avoidance and (to a lesser degree) pellet-eating behaviors but only in the local neighbor cell area.

- Rule-Based (RB) *PacMan*: The RB *PacMan* is a CB *PacMan* plus an additional rule for more effective and global pellet-eating behavior. This rule can be described as follows. If all *PacMan's* neighbor cells are empty ($c = 10$), then the probability of moving towards each one of the available directions (i.e. not towards wall cells) is inversely proportional to the distance to the closest pellet on that direction.

- Advanced (ADV) *PacMan*: The ADV *PacMan* checks in every visible direction (i.e. no wall cell) for *Ghosts*. If there is at least one *Ghost* in sight, then the probability of moving towards each one of the available directions is directly proportional to the distance to a *Ghost* in that direction. If there is no *Ghost* in sight, then the ADV *PacMan* behaves like a RB *PacMan*.

  The ADV moving strategy is expected to produce a more global *Ghost*-avoidance behavior built upon the RB *PacMan's* good pellet-eating strategy.

2.2 Neural Controlled Ghosts

Neural networks are a suitable host for emergent adaptive behaviors in complex multi-agent environments (Ackley and Littman, 1992). A feedforward neural controller is employed to manage the *Ghosts'* motion and is described in this subsection. Apart from the neural controller, three non-evolving fixed ways of controlling the *Ghosts* are presented in subsection 2.3.

Input

Using their sensors, *Ghosts* inspect the environment from their own point of view and decide their next action. Each *Ghost* receives input information from its environment expressed in the neural network's input array of dimension 4 (see Figure 2). The input array consists of the relative distances from (a) *PacMan* in x ($\Delta_{x,P} = x_g - x_p$) and y ($\Delta_{y,P} = y_g - y_p$) axis and (b) the closest *Ghost* in x ($\Delta_{x,C} = x_g - x_c$) and y ($\Delta_{y,C} = y_g - y_c$) axis; where (x_g, y_g), (x_p, y_p) and (x_c, y_c) are the cartesian coordinates of the current *Ghost's*, *PacMan's* and closest *Ghost's* current position respectively. *Ghost's* input includes information for only one neighbor *Ghost* as this constitutes the minimal information for emerging teamwork cooperative behaviors.

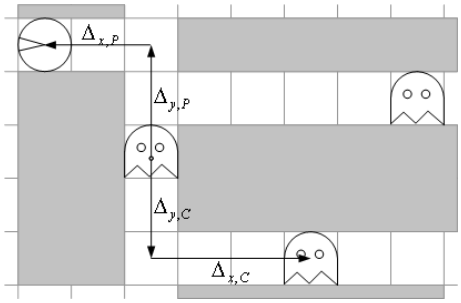


Figure 2: *Ghost's* input data

All input values are linearly normalized into [0, 1] via $0.5[(\Delta_{i,J}/L_i) + 1]$ where $i \in \{x, y\}$, $J \in \{P, C\}$ and L_x, L_y are the width and height of the stage respectively.

Architecture

As previously mentioned, a multi-layered fully connected feedforward neural network has been used for the experiments presented here. The sigmoid function is employed at each neuron.

The connection weights take values from -5 to 5 while the neural network's output is a four-dimensional vector with respective values from 0 to 1 that represents the *Ghost's* four movement options (up, down, left and right respectively). Each *Ghost* moves towards the direction represented by the highest output value.

2.3 Fixed Strategy Ghosts

Apart from the neural controlled *Ghosts*, three additional fixed non-evolving strategies have been tested for controlling the *Ghost's* motion. These strategies are used as baseline behaviors for comparison with any neural controller emerged behavior.

- Random (R): *Ghosts* that randomly (uniform distribution) decide their next available movement. Available movements have equal probabilities to be picked.

- Followers (F): *Ghosts* designed to follow *PacMan* constantly. Their strategy is based on moving so

as to reduce the greatest of their relative distances $(\Delta_{x,P}, \Delta_{y,P})$ from *PacMan*.

- Near-Optimal (O): A *Ghost* strategy designed to produce attractive forces between *Ghosts* and *PacMan* as well as repulsive forces among the *Ghosts*. For each *Ghost* X and Y values are calculated as follows.

$$\begin{aligned} X &= \text{sign}[\Delta_{x,P}]h(\Delta_{x,P}, L_x, 0.25) \\ &- \text{sign}[\Delta_{x,C}]h(\Delta_{x,C} - 1, L_x, 10) \quad (1) \\ Y &= \text{sign}[\Delta_{y,P}]h(\Delta_{y,P}, L_y, 0.25) \\ &- \text{sign}[\Delta_{y,C}]h(\Delta_{y,C} - 1, L_y, 10) \quad (2) \end{aligned}$$

where $\text{sign}[z]{=}z/|z|$ and $h(z, z_m, p) = [1-(|z|/z_m)]^p$. X and Y values represent the axis on which the near-optimal *Ghost* will move. Hence, the axis is picked from the maximum of $|X|$ and $|Y|$ whereas, the direction is decided from this value's sign. That is, if $|X| > |Y|$, then go right if $\text{sign}[X] > 0$ or go left if $\text{sign}[X] < 0$; if $|Y| > |X|$, then go up if $\text{sign}[Y] > 0$ or go down if $\text{sign}[Y] < 0$.

3. Optimal Behavior

When a predator/prey game is investigated from the predator's viewpoint, optimality can be measured in the predators' ability to kill the prey. Thus, a predator's behavior that always manages to kill the prey in such games is obviously a desired behavior in terms of optimality.

Prey-killing ability is the primary factor that determines how good a behavior is (i.e. its performance) in the Pac-Man game as well. Furthermore, the behavior of preventing *PacMan* from eating pellets constitutes an additional factor of the desired optimal behavior. This behavior also implies a fast-killing behavior, which is also desired from optimal predators. Given these, a measure designed to give an approximation of a group of *Ghosts'* performance over a specific number N of games played, is

$$P = \frac{\alpha\frac{k}{N} + \beta\min\{1 + \frac{(e_{min} - E\{e\})}{e_{max}}, 1\}}{\alpha + \beta} \quad (3)$$

where P is the performance of a *Ghost* group behavior taking values from 0 to 1; k is the number of *PacMan* kills within N games; $E\{e\}$ is the average number of pellets eaten by *PacMan* over the N games; e_{min}, e_{max} are the lower and upper bound of the eaten pellets e respectively (in this paper $e_{min} = 70$ and $e_{max} = 187$); α, β are weight parameters (in this paper $\alpha = \beta = 1$).

4. Interesting Behavior

In order to find an objective (as possible) measure of interest in the Pac-Man computer game we first need to define the criteria that make a game interesting. Then, second, we need to quantify and combine all these criteria in a mathematical formula. The game should be, then, tested by human players and therefore have this formulation of interest cross-validated against the interest the game produces in real conditions. This last part of our investigation constitutes a crucial phase of future work.

In order to simplify this procedure we will ignore the graphics' as well as the player's contribution to the interest of the game and we will concentrate on the *Ghost's* behavior that effects the game's interest. That is because, we believe, the computer-guided opponent character contributes the vast majority of features that make a computer game interesting.

By being as objective and generic as possible, we believe that the criteria that collectively define interest on the Pac-Man game are as follows.

1. *When the game is neither too hard nor too easy.* In other words, the game is interesting when *Ghosts* manage to kill *PacMan* sometimes but not always. In that sense, optimal behaviors are not interesting behaviors and *vice versa*.

2. *When there is diversity in Ghosts' behavior over the games.* That is, when *Ghosts* are able to find different ways of hunting and killing *PacMan* in each game so that their strategy is less predictable.

3. *When Ghosts' behavior is aggressive rather than static.* That is, *Ghosts* that move towards killing *PacMan* but meanwhile, move constantly all over the game field instead of simply following it. This behavior gives player the impression of an intelligent strategic *Ghosts'* plan which increases the game interest.

In order to estimate and quantify each of the aforementioned criteria of the game's interest, we let the examined group of *Ghosts* play the game N times (each game for a sufficiently large evaluation period of t_{max} simulation steps; this period should be enough for *PacMan* to clear the stage of pellets; in the experiments presented here $t_{max} = 300$ simulation steps) and we record the simulation steps t_k taken to kill *PacMan* as well as the total number of *Ghosts'* visits v_i at each cell i of the game field for each game. Given these, the quantifications of the Pac-Man game's three interest criteria can be presented as follows.

1. According to the first criterion, the estimate of how interesting the behavior is, is given by T in (4).

$$T = [1 - (E\{t_k\}/\max\{t_k\})]^{p_1} \quad (4)$$

where

$$t_{min} \leq t_k \leq t_{max} \quad (5)$$

and $E\{t_k\}$ is the average number of simulation steps taken to kill *PacMan* over the N games; $\max\{t_k\}$ is the maximum t_k over the N games; t_{min} is the minimum number of simulation steps required for *Ghosts* to kill *PacMan* (t_{min} is 30 simulation steps in this paper); p_1 is a weighting parameter (for the experiments presented here $p_1 = 0.5$).

The T estimate of interest demonstrates that the greater the difference between the average number of steps taken to kill *PacMan* and the maximum number of steps taken to kill *PacMan*, the higher the interest of the game. Given (4), both poor-killing ('*too easy*') and near-optimal ('*too hard*') behaviors get low interest estimate values (i.e. $E\{t_k\} \simeq \max\{t_k\}$).

2. The interest estimate for the second criterion is given by S in (6).

$$S = \left(s^2/s^2_{max}\right)^{p_2} \tag{6}$$

where

$$s^2_{max} = \frac{N}{4(N-1)}(t_{max} - t_{min})^2 \tag{7}$$

and s^2 is the sample variance of t_k over the N games; s^2_{max} is the maximum value of s^2; p_2 is a weighting parameter (for the experiments presented here $p_2 = 1$).

The S estimate of interest demonstrates that the greater the variance of the steps taken to kill *PacMan* over N games, the higher the interest of the behavior. Therefore, by using (6) we promote *Ghosts* that produce high diversity in the time taken to kill *PacMan*.

3. A good measure for quantifying the third interest criterion is through entropy of the *Ghosts'* cell visits in a game, which quantifies the completeness and uniformity with which the *Ghosts* cover the stage. Hence, for each game, the cell visits' entropy H is calculated via (8) and it is normalized into $[0, 1]$ via (9).

$$H = -\sum_i \frac{v_i}{V} log\left(\frac{v_i}{V}\right) \tag{8}$$

$$H_n = (H/logV)^{p_3} \tag{9}$$

where V is the total number of visits of all visited cells (i.e. $V = \sum_i v_i$) and p_3 is a weighting parameter (for the experiments presented here $p_3 = 4$).

Given the normalized entropy values for all N games, the interest estimate for the third criterion can be presented as their average value $E\{H_n\}$ over the N games. This implies that the higher the average entropy value, the more interesting the game becomes.

All three criteria are combined linearly (10)

$$I = \frac{\gamma T + \delta S + \epsilon E\{H_n\}}{\gamma + \delta + \epsilon} \tag{10}$$

where I is the interest value of the Pac-Man game; γ, δ and ϵ are criterion weight parameters (for the experiments presented here $\gamma = 1, \delta = 2, \epsilon = 3$).

The measure of the Pac-Man game's interest introduced in (10) can be effectively applied to any predator/prey computer game because it is based on generic features of this category of games. These features include the time required to kill the prey as well as the predators' entropy throughout the game field. We therefore believe that (10) — or a similar measure of the same concepts — constitutes a generic interest approximation of predator/prey computer games.

5. Off-line learning

As previously stressed, our primary aim is to generate emergent *Ghost* behaviors that make the game interesting via an on-line learning mechanism. We therefore use an off-line evolutionary learning approach in order to produce some 'good' (i.e. in terms of performance) initial behaviors for the on-line learning mechanism.

The neural networks that determine the behavior of the *Ghosts* are themselves evolved. In the algorithm presented here, the evolving process is limited to the connection weights of the neural network.

The evolutionary procedure, which is based on previous work by Yannakakis et al. (2003), is as follows. Each *Ghost* has a genome that encodes the connection weights of its neural network. A population of 80 (we keep this number low because of the computational cost) neural networks (*Ghosts*) is initialized randomly with initial uniformly distributed random connection weights that lie within [-5, 5]. Then, at each generation:

Step 1: Every *Ghost* in the population is cloned 4 times. These 4 clones are placed in the Pac-Man game field and play N_t games, each one for an evaluation period of t simulation steps (in the experiments presented in this paper $N_t = 10$ games and $t = 300$ simulation steps). The outcome of these games is to ascertain the time taken to kill *PacMan* t_k for each game.

Step 2: Each *Ghost* is evaluated via (11) for each game and its fitness value is given by $E\{f\}$ over the N_t games.

$$f = [1 - (t_k/t)]^{\frac{1}{4}} \tag{11}$$

By the use of the f fitness function, that takes values from 0 to 1, we promote *PacMan*-killing behaviors capable of achieving high performance values P.

Step 3: A pure elitism selection method is used where only the 10% best fit solutions determine the members of the intermediate population and, therefore, are able to breed.

Step 4: Each parent clones an equal number of offspring in order to replace the non-picked solutions from elitism.

Step 5: Mutation occurs in each gene (connection weight) of each offspring's genome with a small probability p_m (e.g. 0.01). A uniform random distribution is used again to define the mutated value of the connection weight.

The algorithm is terminated when a predetermined number of generations g is achieved (e.g. $g = 1000$) and the best-fit *Ghost's* connections weights are saved.

Ghosts play few games (i.e. $N_t = 10$) when evaluated by the off-line learning mechanism. Even though, this evaluation procedure constitutes an approximation of the examined *Ghost's* overall performance in a greater number of games, it keeps the computational cost in low levels.

6. On-line learning (OLL)

This learning approach is based on the idea of *Ghosts* that learn while they are playing against *PacMan*. In other words, *Ghosts* that are reactive to any player's behavior and learn from its strategy instead of being the predictable and, therefore, uninteresting characters that exist in all versions of this game today. Furthermore, this approach's additional objective is to keep the game's interest at high levels as long as it is being played.

Beginning from any initial group of homogeneous off-line trained (OLT) *Ghosts*, the OLL mechanism attempts to transform them into a group of heterogeneous *Ghosts* that are interesting to play against.

The OLL procedure is as follows. An OLT *Ghost* is cloned 4 times and its clones are placed in the Pac-Man game field to play against a selected *PacMan* type of player. Then, at each generation:

Step 1: Each *Ghost* is evaluated every t simulation steps via (12), while the game is played. For the experiments presented in this paper $t = 50$ simulations steps.

$$f' = \sum_{i=1}^{t/2} (d_{P,2i} - d_{P,(2i-1)}) \tag{12}$$

where $d_{P,i}$ is the distance between the *Ghost* and *PacMan* at the i simulation step. This fitness function promotes *Ghosts* that move towards *PacMan* within an evaluation period of t simulation steps.

Step 2: A pure elitism selection method is used where only the fittest solution is able to breed. The best-fit parent clones an offspring with a probability p_c that is inversely proportional to the normalized cell visits' entropy (i.e. $p_c = 1 - H_n$) given by (9). In other words, the higher the cell visits' entropy of the *Ghosts*, the lower the probability of breeding new solutions. If there is no cloning, then go back to Step 1, else continue to Step 3.

Step 3: Mutation occurs in each gene (connection weight) of each offspring's genome with a small probability p_m (e.g. 0.01). A gaussian random distribution is used to define the mutated value of the connection weight. The mutated value is obtained from (13).

$$w_m = \mathcal{N}(w, 1 - H_n) \tag{13}$$

where w_m is the mutated connection weight value and w is the connection weight value to be mutated. The gaussian mutation, presented in (13), suggests that the higher the normalized entropy of a group of *Ghosts*, the smaller the variance of the gaussian distribution and therefore, the less disruptive the mutation process as well as the finer the precision of the GA.

Step 4: The cloned offspring is evaluated briefly via (12) in off-line mode, that is, by replacing the worst-fit member of the population and playing an off-line (i.e. no visualization of the actions) short game of t simulation steps. The fitness values of the mutated offspring and the worst-fit *Ghost* are compared and the better one is kept for the next generation. This pre-evaluation procedure for the mutated offspring attempts to minimize the probability of group behavior disruption by low-performance mutants. The fact that each mutant's behavior is not tested in a single-agent environment but within a group of heterogeneous *Ghosts* helps more towards this direction. If the worst-fit *Ghost* is replaced, then the mutated offspring takes its position in the game field as well.

The algorithm is terminated when a predetermined number of games has been played.

We mainly use small simulation periods (i.e. $t = 50$) in order to evaluate *Ghosts* in OLL. The aim of this high frequency of evaluations is to accelerate the on-line evolutionary process. The same short period is used for evaluating mutated offspring. The two primary objectives of this are: 1) to apply a fair comparison between the mutated offspring and the worst-fit *Ghost* (i.e. same evaluation period) and 2) to avoid undesired high computational effort in on-line mode (i.e. while playing). However, the evaluation function (12) constitutes an approximation of the examined *Ghost's* overall performance for large simulation periods. Keeping the right

balance between computational effort and performance approximation is one of the key features of this approach. In the experiments presented here, we use minimal evaluation periods capable of achieving good estimation of the *Ghosts'* performance.

7. Results and analysis

Results obtained from experiments applied on the Pac-Man game are presented in this section. These include, off-line and on-line learning emergent behavior analysis as well as experiments for testing robustness and adaptability of the on-line mechanism proposed.

7.1 Off-line experiments

The experiment presented in this subsection is focused on producing well-behaved *Ghosts* in terms of the performance measure described in section 3. We train *Ghosts* against all three types of *PacMan* players through the off-line learning mechanism presented in section 5. The off-line learning experiment is described as follows.

- Apply the off-line learning mechanism by playing against each type of *PacMan* players separately.
- *Ghosts* trained against a specific type of *PacMan* player are evaluated by playing 100 non-evolution games against the same *PacMan* type.
- In order to minimize the non-deterministic effect of the *PacMan's* strategy on the *Ghost's* performance and interest values as well as to draw a clear picture of these averages' distribution, we apply the following procedure. Using a uniform random distribution we pick 10 different 50-tuples out of the 100 aforementioned games. These 10 samples of data (i.e. e, k, t_k, v_i) from 50 games (i.e. $N = 50$) are used to determine the *Ghosts'* average performance and interest values. The outcome of this experiment is presented in Table 1.

More analytically, in Table 1, both the performance and the interest values of six different *Ghosts'* behaviors against all three different *PacMan* types as well as their average values $E\{\}$ are presented for comparison. In the first three rows of Table 1 the fixed strategy *Ghosts'* (i.e. R:Random, F:Followers, O:Near-Optimal) performance and interest values are presented against each type of *PacMan* player. Furthermore, P and I values of three different emergent behaviors against each *PacMan* type (9 in total) are presented. These behaviors can be described as follows.

- Blocking (B): These are the OLT *Ghosts* that achieve the best performance ($P > 0.75$) against each *PacMan* type. Their behavior is characterized as 'Blocking' because they tend to wait for *PacMan* to enter into a specific area that is easy for them to block and kill.
- Aggressive (A): These are suboptimal OLT *Ghosts* that achieve lower performance ($0.6 < P \leq 0.75$) in comparison to the blockers. Their behavior is characterized as 'Aggressive' because they tend to follow *PacMan* all over the stage in order to kill it.
- Hybrid (H): These are suboptimal OLT *Ghosts* that achieve the lowest performance ($P \leq 0.6$) in comparison to the aforementioned B and A *Ghosts*. Their behavior is characterized as 'Hybrid' because they tend to behave as a Blocking-Aggressive hybrid which proves to be ineffective at killing *PacMan*.

	Trained off-line by playing against					
	CB		RB		ADV	
	P	I	P	I	P	I
R	0.446	0.429	0.4	0.43	0.411	0.424
F	0.776	0.705	0.839	0.733	0.822	0.722
O	1.0	0.607	0.96	0.625	1.0	0.582
B	0.987	0.458	0.79	0.555	0.875	0.535
A	0.724	0.646	0.7	0.614	0.749	0.513
H	0.6	0.297	0.587	0.476	0.464	0.357
$E\{\}$	0.755	0.524	0.713	0.572	0.72	0.522

Table 1: Performance (P) and Interest (I) values (average values of 10 samples of 50 games each) of fixed strategy (R, F, O) and OLT *Ghosts* (B, A, H) playing against all three *PacMan* types (CB, RB, ADV). Average P and I values ($E\{\}$) of all six strategies appear in the bottom row. Experiment Parameters: population size is 80, $g = 1000$, $t = 300$ simulation steps, $N_t = 10$ games, $p_m = 0.01$, 5-hidden neurons controller.

The two main criteria in distinguishing the aforementioned behavior types are their performance values and their behavior seen on screen.

Near-optimal and Blocking behavior *Ghosts* achieve high-performance values against all three *PacMan* types whereas their interest value is not as high as their performance value. This illustrates the compromise between optimality and interest we have to make because, in a predator/prey computer game, optimal killing behaviors cannot be interesting behaviors.

On the other hand, Followers and Aggressive behavior *Ghosts* are likely to produce the two most interesting behaviors (among the behaviors examined in Table 1) for the game. Given the highest achieved interest values (Followers — 0.705 against CB, 0.733 against RB and 0.722 against ADV), we can determine the threshold of $I \geq 0.7$ in accepting a team of *Ghosts'* behavior (i.e. a game) as interesting.

Viewing results presented in Table 1 from the *PacMan* type perspective (i.e. average values in the bottom row of

the table), it looks as if the RB is the harder to kill and, meanwhile, the most interesting *PacMan* player for the *Ghosts* to play against. Furthermore, the CB *PacMan* seems to be the easiest to kill whereas, ADV — even though apparently not significantly different from CB — seems to be the least interesting type of *PacMan* for *Ghosts* to play against.

7.2 On-line experiments

As previously mentioned, the off-line learning procedure is a mechanism that produces near-optimal solutions to the problem of killing *PacMan* and minimizing the pellets eaten in a game. These solutions will be the OLL mechanisms' initial points in the search for more interesting games. The OLL experiment is described as follows.

- Pick the nine different emerged *Ghosts'* behaviors produced from the off-line learning experiments presented in subsection 7.1 (i.e. B, A and H behaviors emerged by playing against each *PacMan* type).
- Starting from each OLT behavior, apply the OLL mechanism by playing against each type of *PacMan* player separately. This makes a total of 27 different OLL attempts.
- Calculate the interest of the game every 100 games during each OLL attempt.

In order to calculate the interest of the game we take the *Ghosts'* neural controllers every 100 OLL games and evaluate them by letting them play 100 non-evolution games against the *PacMan* type they were playing against during OLL. We need to minimize the non-deterministic effect of the *PacMan's* strategy on the game's interest and therefore, we use a uniform random distribution to pick 10 different 50-tuples out of these 100 games. These 10 samples of data (i.e. t_k, v_i), of 50 games each (i.e. $N = 50$), are used to determine the games' average as well as boundary values of interest described in section 4. The outcome of this experiment is presented in Table 2 and Figure 3.

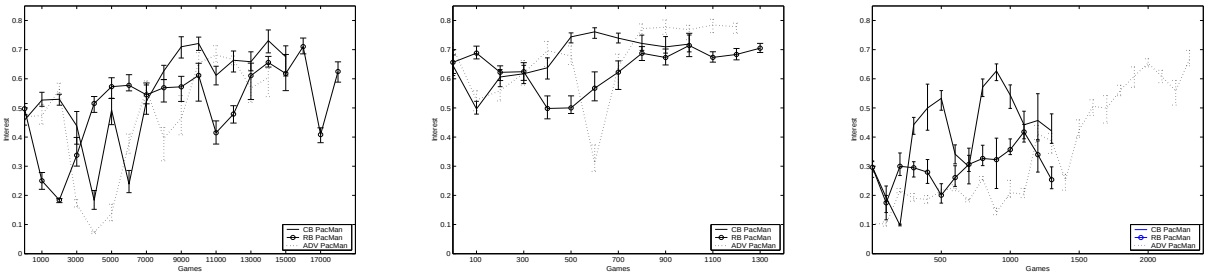
Figure 3 illustrates the overall picture of the OLL experiments. The evolution of interest over the OLL games of each one of the nine different OLT behaviors is presented in a sub-figure of Figure 3. For each sub-figure, three lines are illustrated, representing the interest values of the OLL attempt playing against the three different *PacMan* types (a total of 27 different OLL attempts).

As seen from Figure 3, the OLL mechanism manages to find ways of increasing the interest of the game no matter what the initial OLT behavior or the *PacMan* player they play against. In all experiments presented here the learning mechanism is capable of producing games of higher than the initial interest as well as keeping that high interest for a long period. There is obviously a slight probability of disruptive mutations (i.e. the higher the game's interest, through the cell visit's entropy value, the less the probability of mutation — see section 6) that can cause undesired drops in the game's interest. However, as seen from Figure 3, the learning mechanism is robust enough to recover from such disruptive phenomena. When the initial *Ghost* behavior is interesting (see Figures 3(b) and 3(e)) then the mechanism is likely to keep the game at these high, or ever higher, levels of interest. Given an interesting initial behavior (e.g. Aggressive behavior, $I > 0.6$) or even a suboptimal H behavior (see Figures 3(c) and 3(f)), it takes some hundreds of games (i.e. around 500 games in most cases) for the learning mechanism to produce games of high interest. On the other hand, it takes some thousand games to transform an uninteresting near-optimal blocking behavior (see Figures 3(a), 3(d) and 3(g)) into an interesting one. That is because the OLL process requires an initial long period to disrupt the features of an uninteresting blocking behavior in order to be able to increase the interest of the game. This long period of disruption appears when the initial on-line *Ghosts'* behavior (B, A, or H) is emerged by playing against ADV *PacMan* as well (see Figures 3(g), 3(h) and 3(i)). This happens likely because off-line training against ADV *PacMan* seems to produce the least interesting games (see Table 1).

It is obvious that a number in the scale of 10^3 constitutes an unrealistic number of games for a human player to play. In other words, it is very unlikely for a human player to play so many games in order to see the game's interest increasing. The reason for the OLL process being that slow is a matter of keeping the right balance between the process' speed and its 'smoothness' (by 'smoothness' we define the interest's magnitude of change over the games). A solution to this problem is to consider the initial long period of disruption as an off-line learning procedure and start playing as soon as the game's interest is increased. How effective will this mechanism be in a potential change from a fixed strategy to a human *PacMan* player? Subsection 7.3 provides evidence in order to support the answer.

			OLL - Playing against		
			CB	RB	ADV
Initial OLT behaviors	CB	B	0.731	0.711	0.682
		A	0.761	0.714	0.772
		H	0.627	0.418	0.609
	RB	B	0.743	0.727	0.724
		A	0.716	0.688	0.705
		H	0.673	0.772	0.676
	ADV	B	0.701	0.574	0.564
		A	0.706	0.591	0.733
		H	0.763	0.595	0.675

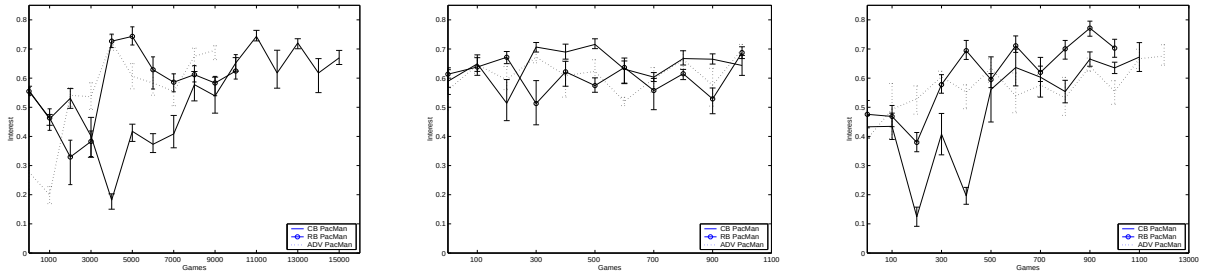
Table 2: Best interest values achieved from on-line learning.



(a) B OLT against CB *PacMan*

(b) A OLT against CB *PacMan*

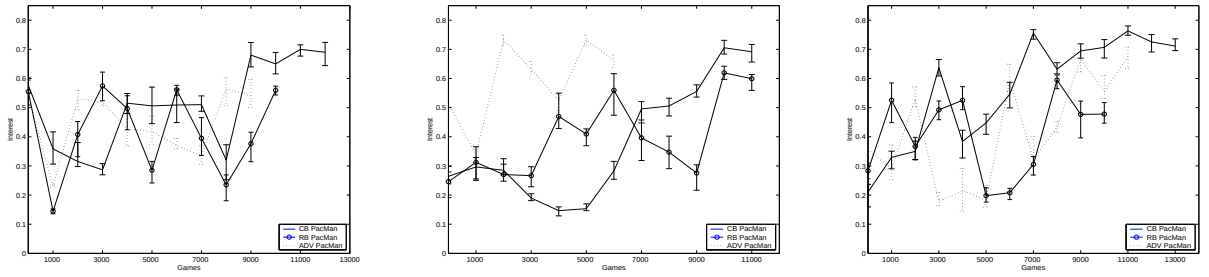
(c) H OLT against CB *PacMan*



(d) B OLT against RB *PacMan*

(e) A OLT against RB *PacMan*

(f) H OLT against RB *PacMan*



(g) B OLT against ADV *PacMan*

(h) A OLT against ADV *PacMan*

(i) H OLT against ADV *PacMan*

Figure 3: Game interest over the number of OLL games. For reasons of computational effort, the OLL procedure continues for a number of games, large enough to illustrate the mechanism's behavior, after a game of high interest ($I \geq 0.7$) is found. Initial *Ghost* behaviors appear in sub-figure captions. Experiment Parameters: $t = 50$ simulation steps, $p_m = 0.01$, 5-hidden neurons controller.

Table 2 presents the best average interest values obtained from the OLL mechanism. It is clear that the OLL approach constitutes a robust mechanism that, starting from near-optimal or suboptimal *Ghosts*, manages to emerge interesting games (i.e. interesting *Ghosts*) in the majority of cases (i.e. in 15 out of 27 cases $I > 0.7$). It is worth mentioning that in 10 out of 27 different OLL attempts the best interest value is greater than the respective Follower's value (i.e. 0.705 against CB, 0.733 against RB and 0.722 against ADV).

7.3 Adaptability

In order to test the OLL mechanism's ability to adapt to a changing environment (i.e. change of *PacMan* strategy), the following experiment is proposed. Beginning from an initial OLT behavior we apply the OLL mechanism against a specific *PacMan* type. During the on-line process we keep changing the type of player as soon as interesting games (i.e. $I > 0.7$) are produced. The process stops when all types of players have played the game.

Since we have three types of players, the total number of different such experiments is 6 (all different player type sequences). These experiments illustrate the over-

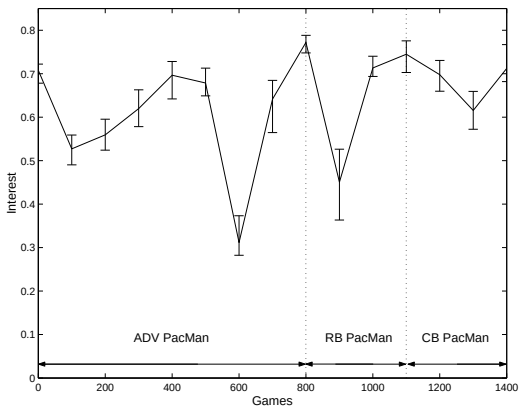


Figure 4: On-line learning *Ghosts* playing against changing types of *PacMan*. Initial behavior: A OLT against CB *PacMan*.

all picture of the mechanism's behavior against any sequence of *PacMan* types. Due to space considerations we present only one (see Figure 4) out of the 6 experiments here. As seen in Figure 4, the OLL mechanism is able to quickly recover a sudden change in the player's strategy (i.e. from ADV to RB and finally to CB *PacMan*) and increase the game's interest at high levels after a number of games has been played (i.e. 100 to 400 games). The mechanism demonstrated a similar adaptive behavior for all 6 different sequences of *PacMan* players which illustrates the mechanism's independence from the sequence of the changing *PacMan's* type.

In addition, by experimenting with all sorts (B, A, H) of initial behaviors, the mechanism's time scale of adaptation to new players is seen to be a function of the initial *Ghosts'* behavior: the more interesting the initial behavior, the faster the adaptation to new players.

Results obtained from this experiment provide evidence for the mechanism's ability to adapt to new types of players as well as its efficiency in producing interesting games against human players.

8. Conclusion

Predator strategies in prey/predator computer games are still nowadays based on simple rules which make the game pretty predictable and, therefore, uninteresting (by the time the player gains more experience and playing skills). A computer game becomes interesting primarily when there is an on-line interaction between the player and its opponents who demonstrate interesting behaviors.

Given some objective criteria for defining interest in predator/prey games presented in this paper we introduced a generic method for measuring interest in such games. We saw that by using the proposed on-line learning mechanism, maximization of the individual simple distance measure (see (12)) coincides with maximization of the game's interest. Apart from being fairly robust, the proposed mechanism demonstrates high adaptability to changing types of player (i.e. playing strategies). Therefore, we believe that such a mechanism will be able to produce interesting interactive opponents (i.e. games) against even the most complex human playing strategy.

On the other hand, for future work, we believe that the methods used need to be tested on more complex Pac-Man stages in order to provide more evidence for their generality, and the interest measure proposed needs to be cross-validated against human players.

References

Ackley, D. H. and Littman, M. L. (1992). Interactions between learning and evolution. In *Artificial Life II*, pages 478–507, Reading, MA. Sante Fe Institute Studies in the Sciences and Complexity, Addison-Wesley.

Haynes, T. and Sen, S. (1995). Evolving behavioral strategies in predators and prey. In *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37. Morgan Kaufmann.

Koza, J. (1992). *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press.

Luke, S. and Spector, L. (1996). Evolving teamwork and coordination with genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156. MIT Press.

Miller, G. and Cliff, D. (1994). Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pages 411–420. MIT Press.

Rabin, S. (2002). *AI Game Programming Wisdom*. Charles River Media, Inc.

Rosca, J. (1996). Generality versus size in genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 381–387, Stanford University, CA, USA. MIT Press.

Woodcock, S. (2001). Game AI: The State of the Industry 2000-2001: It's not Just Art, It's Engineering. Game Developer magazine.

Yannakakis, G. N., Levine, J., Hallam, J., and Papageorgiou, M. (2003). Performance, robustness and effort cost comparison of machine learning mechanisms in *FlatLand*. In *Proceedings of the 11th Mediterranean Conference on Control and Automation MED'03*. IEEE.

Yao, X. (1999). Evolving artificial neural networks. In *Proceedings of the IEEE*, volume 87, pages 1423–1447.