An Evolutionary Approach for 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 Levine
Centre for Intelligent Systems
and their Applications
The University of Edinburgh
AT, Crichton Street, EH8 9LE
johnl@inf.ed.ac.uk

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

***Abstract*— In this paper we introduce the first stage of experiments on neuro-evolution mechanisms applied to predator/prey multi-character computer games. Our test-bed is a computer game where the prey (i.e. player) has to avoid its predators by escaping through an exit without getting killed. 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. Their efficiency is based on cooperation which emerges from an abstract type of partial interaction with their environment. In addition, investigation of behavior generalization demonstrated the crucial contribution of playing strategies in the development of successful predator behaviors.**

However, emergent well-behaved opponents trained off-line with fixed strategies do not make the game interesting to play. We therefore 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 the predators' performance while altering their behavior as long as the game is played. Computer game opponents following this on-line learning approach show high adaptability to changing player strategies, which provides evidence for the approach's effectiveness and interest against human players.

I. 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 [1]. 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.

In this paper, we introduce a predator/prey computer game named 'Dead End' for emerging complex and cooperative behaviors among agents through evolutionary procedures. In Dead End, the player's ('Player's') goal is to escape through an Exit in a square-shaped stage while avoiding being killed by eight opponent characters named '*Dogs*'. On the other hand, the *Dogs* are aiming to kill (by 'touching') the Player as soon as possible. The game is over when either the Player escapes through the Exit or the *Dogs* manage to kill the Player. In that case, the game restarts from the same initial positions for the *Dogs* and a randomly chosen position for the Player at the bottom of the stage. Since there are eight *Dogs* on the game field, they are designed to be slower than the Player so that the game is fairer to play. The name Dead End is devised to demonstrate the situation in which the Player finds itself at the beginning of each game whereas the game's fundamental concepts are inspired from previous work of Yannakakis et al. [2].

There are some examples, in the predator/prey domain literature, of researchers attempting to teach a controller to drive an agent in order to avoid being eaten by predators. Koza [3] considers the problem of controlling an agent in a dynamic non-deterministic environment and, therefore, sees the Pac-Man prey/predator computer game as an interesting multi-agent environment for applying off-line learning techniques based on genetic programming. The same application domain has been used for analyzing size and generality issues in genetic programming [4].

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 [5], among others, have designed an environment similar to the Dead End game (i.e. Serengeti world) in order to examine different breeding strategies and coordination mechanisms for the predators. Finally, there are examples such as Haynes' [6] and Miller's [7] work in which both the predators' and the prey's strategies are co-evolved in grid-based and continuous environments respectively.

Similar to Luke and Spector [5], we view Dead End from the *Dogs'* perspective. Our first aim is to emerge effective complex teamwork behaviors by the use of an off-line training approach, based on evolutionary computation techniques, applied to homogeneous neural controlled agents [8]. *Dogs* have to demonstrate good cooperative strategies in order to kill the Player and/or to defend the Exit. Such behaviors can be aggressive, defensive, or a hybrid of the two. Given the specific game, we believe that 8 predators are enough for cooperative behaviors to emerge. Furthermore, given the off-line emergent behaviors, we investigate their generalization against playing strategies other than the strategy of the Player they have been trained against.

However, playing a computer game like Dead End against

well-playing opponents of fixed hunting behaviors cannot be regarded as interesting. 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 and maintaining the *Dogs's* performance (starting from well performing behaviors trained off-line), as well as changing the *Dog's* behaviors — which increases the game's interest as long as the game is being played. In our Dead End predator/prey computer game we require *Dogs* 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).

This paper is organized as follows. In Section II, we present a detailed description of the Dead End game as well as its characters' controllers. In Section III, we discuss the difficulties of the problem as well as some issues of interest of our approach for the multi-agent computer games field. The off-line and on-line machine learning mechanisms used are analytically described in Section IV and Section V respectively. Results obtained from this work are presented in Section VI. Finally, the most important conclusions of the Dead End research are summarized in Section VII.

II. The Dead End Game

In this section, we present a detailed description of the Dead End game and its two main characters, *Dogs* and the Player. As previously mentioned, the Dead End game is investigated from the viewpoint of *Dogs* and more specifically how *Dogs'* emergent adaptive behaviors can be effective against skilled Players as well as contribute to the interest of the game. Dead End is a two-dimensional, multi-agent, grid-motion, predator/prey game. The game field (i.e. stage) is a two-dimensional square world that contains a white rectangular area named "Exit" (see Fig. 1) at the top. For the experiments presented in this paper we use the 16 cm $\times$ 16 cm stage presented in Fig. 1, which is divided into grid squares (of length 0.5 mm).

The characters visualized in the Dead End game (as illustrated in Fig. 1) are a dark grey circle of radius 0.75 cm representing the Player and 8 light grey square (of dimension 1.5 cm) characters representing the *Dogs*.

The relationship between the *Dogs* and the Player is mutually highly competitive. The aim of a Player is to reach the Exit, avoiding the *Dogs*. On the other hand, the aims of the *Dogs* are to defend the Exit and/or to catch the Player. In Dead End, if a Player succeeds in arriving at the Exit, this event is described as a *win*. Additionally, if a *Dog* manages to catch a Player, this event defines a *kill*. If there is neither a Player win nor a kill for a predetermined large period of time, then the outcome of the game is a *win* again. After either a win or a kill, a new game starts.

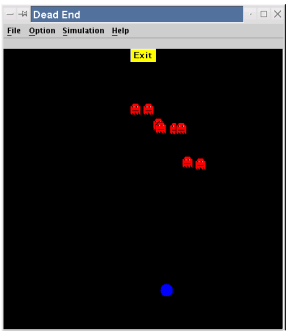

Fig. 1. A snapshot of the Dead End game

The Player moves at four thirds the *Dogs'* maximum speed and since there are no dead ends, it is impossible for a single *Dog* to complete the task of killing it. Since the Player moves faster than a *Dog*, the only effective way to kill the Player is for a group of *Dogs* to hunt cooperatively. It is worth mentioning that one of the *Dogs'* properties is permeability. In other words, two or more *Dogs* can simultaneously occupy the same position on the game field.

The simulation procedure of the Dead End game is as follows. Player and *Dogs* 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 Player and *Dogs* gather information from their environment and take a movement decision, up, down, left or right.
2) If the game is over (i.e. Player escapes through the Exit, Player is killed, or the simulation step is greater than a predetermined large number), then a new game starts from the same initial positions for the *Dogs* but from a different, randomly chosen position, at the bottom of the stage for the Player.
3) Statistical data, such as the outcome of the game (kill or win) and the distance between *Dogs* and the Player at each simulation step, are recorded.

A. The Player

The difficulty of the Dead End game is directly affected by the intelligence of the Player. Its nature is significant because *Dogs'* emergent behavior is strongly related to their competitive relationship against it. To develop more diverse agents' behaviors, different playing strategies are required. We therefore chose three fixed *Dog*-avoidance and/or Exit-achieving strategies for the Player, differing in complexity and effectiveness. Each strategy is based on decision making applying a cost or probability approximation to the player's four directions.

As previously mentioned, the Player starts a game at a random position at the bottom of the game field and its aim is to reach the game's Exit by avoiding the *Dogs*. The non-deterministic initial position is devised to provide *Dogs* with diverse examples of playing behaviors to learn from.

1) Randomly-moving (RM) Player: A Randomly-moving Player takes a movement decision by selecting a randomly (i.e. uniformly distributed) picked direction at each simulation step of the game. After performing a predetermined number of movement decisions, it then proceeds directly to the Exit. The total number of such random movements for the experiments presented here is 500.

2) Exit-achieving (EA) Player: An Exit-achieving Player moves directly towards the Exit. Its strategy is based on moving so as to reduce the greatest of its relative distances from the Exit.

3) Cost-based path planning (CB) Player: A cost-based path planning Player constitutes the most efficient *Dog*-avoiding and Exit-achieving strategy of the three different fixed-strategy types of Player. A discrete Artificial Potential Field (APF), specially designed for the Dead End game, controls the CB Player's motion. The essence of the APF is that points along the Players's path to its Exit are considered to be attractive forces (i.e. low moving cost points), while obstacles (i.e. *Dogs*) in the environment are repulsive forces (i.e. high moving cost points) [9]. The overall APF causes a net force to act on the Player, which guides it along a *Dog*-avoidance, Exit-achievement path. For illustration, consider the CB Player as a small cube that slides down the surface plotted in Fig. 2.

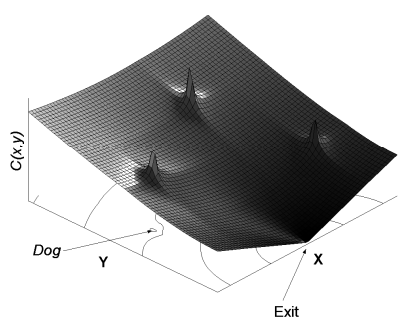

Fig. 2. APF of the CB Player; situation of three obstacles — *Dogs* ($N = 3$).

This surface is plotted by the CB Player at every simulation step and represents the function $C(x, y)$:

$$C(x, y) = \Delta_E(x, y) + D(x, y) \tag{1}$$

$$\Delta_E(x, y) = \sqrt{(x_e - x)^2 + (y_e - y)^2} \tag{2}$$

$$D(x, y) = \sum_{n=1}^{N} \frac{\rho}{|x_{d,n} - x| + |y_{d,n} - y|} \tag{3}$$

where $C(x, y)$ is the cost of the grid square (x, y); N is the total number of *Dogs* in the game field; (x_e, y_e) are the cartesian coordinates of the Exit's center; ($x_{d,n}, y_{d,n}$) are the current cartesian coordinates of the nth *Dog's* center; (x_p, y_p) are the current cartesian coordinates of the Player's center; ρ is a parameter that defines the height of the *Dog's* cost 'hill' function presented in (3) — for the experiments presented in this paper $\rho = 1000$ (note that the CB Player can 'see' all the *Dogs* while a *Dog* can 'see' only its nearest neighbor — see Section II-B.1).

A CB Player, at each simulation step, calculates the moving cost (see (1)) of each grid square in a circle of radius 2 cm, centered at its current position. Then, the CB player moves 2 cm to the grid square of minimal cost on the perimeter of the circle by following the grid-based trajectory of minimal average cost. While, in theory, APFs may be prone to local minima, in practice, in the dynamic Dead End game, the probability for such cases to occur is significantly low and, therefore, can be ignored.

Any motion strategy that guides a Player to arrive quickly at the Exit, avoiding any *Dogs* and keeping to the straightest and fastest possible trajectory, is definitely a "good" strategy in terms of the Dead End game. Hence, the CB Player presents a "good" behavior in our computer game and furthermore a reference case to compare to human playing behavior.

B. Neural Controlled Dogs

Neural networks are a suitable host for emergent adaptive behaviors in complex multi-agent environments [10]. A feed-forward neural controller is employed to manage the *Dogs'* motion and is described in this subsection.

1) Input: Using their sensors, *Dogs* inspect the environment from their own point of view and decide their next action. Each *Dog* receives input information from its environment expressed in the neural network's input array of dimension 6. The input array consists of the relative distances from (a) the Player in x ($\Delta_{x,P} = x_d - x_p$) and y ($\Delta_{y,P} = y_d - y_p$) axis, (b) the closest *Dog* in x ($\Delta_{x,C} = x_d - x_c$) and y ($\Delta_{y,C} = y_d - y_c$) axis and (c) the Exit in x ($\Delta_{x,E} = x_d - x_e$) and y ($\Delta_{y,E} = y_d - y_e$) axis; where (x_d, y_d), (x_p, y_p), (x_e,y_e) and (x_c,y_c) are the cartesian coordinates of the current *Dog's*, the Player's, the Exit's and the closest *Dog's* current position respectively. *Dog's* input includes information for only one neighbor *Dog* as this constitutes the minimal information for emerging teamwork cooperative behaviors.

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

2) Architecture: As previously mentioned, a multi-layered fully connected feedforward neural network has been used for the experiments presented here (as shown in Fig. 3). The hyperbolic tangent sigmoid function is employed at each neuron.

3) Output: The neural network's output is a two-dimensional vector $[o_1, o_2]$ with respective values from -1 to 1. This vector represents the *Dog's* chosen motion and is converted into cartesian coordinates according to (4) and (5).

$$x_d^{k+1} = \begin{cases} x_d^k, & \text{if } |o_1| \geq |o_2| \quad \text{(4a)} \\ x_d^k + o_2 s, & \text{if } |o_1| < |o_2| \quad \text{(4b)} \end{cases}$$

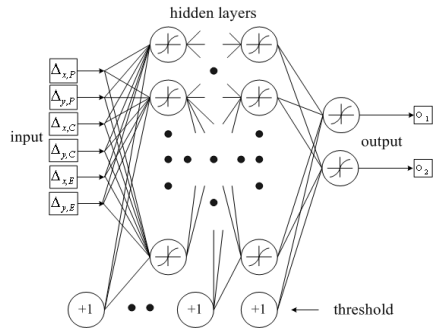

Fig. 3. Multi-layered fully connected feedforward neural network controller

$$y_d^{k+1} = \begin{cases} y_d^k + o_1 s, & \text{if } |o_1| \geq |o_2| \quad (5a) \\ y_d^k, & \text{if } |o_1| < |o_2| \quad (5b) \end{cases}$$

where (x_d^k, y_d^k) are the cartesian coordinates of the *Dog's* center at simulation step k; s is the *Dog's* maximum speed — in the experiments presented here $s = 1.5$ cm/simulation step (this being the 3/4 of the Player's speed).

C. Fixed strategy Dogs

Apart from the neural controlled *Dogs*, an additional fixed non-evolving strategy has been tested for controlling the *Dogs'* motion. *Dogs* of this strategy are called 'Followers' and they are designed to follow the Player constantly by moving at half the Player's speed (i.e. 1.0 cm/simulation step). Their strategy is based on moving so as to reduce the greatest of their relative distances $(\Delta_{x,P}, \Delta_{y,P})$ from the Player. This strategy is used as a baseline behavior for comparison with any emergent neural controller behavior.

III. Difficulty of the problem

Dead End is a hard environment for an agent to achieve behaviors of high performance because of the following distinct features:

- It is a fully dynamic multi-agent environment, in which each *Dog* moves continuously in the game field while interacting with seven other *Dogs* and the Player.
- No agent has full information: *Dogs* can 'see' the Player and at most z (z is 1 in this paper) other *Dogs* while advanced Players can see all the *Dogs'* positions but not their future movements.
- Communication between the *Dogs* is limited to each being able to 'see' the position of z nearest neighbor *Dogs* — cooperative action must be built on this implicit and partial communication.
- A *Dog's* input is discontinuous because its nearest neighbor(s) alter; hence the values of the relative coordinates $(\Delta_{x,C}, \Delta_{y,C})$ also change, in a discontinuous fashion.

The basic concept and features of Dead End make it interesting to the multi-agent computer games field. Its key features are that cooperative behavior amongst the *Dogs* is necessary and is supported only by implicit partial communication and the on-line real-time learning mechanism that we propose allows the *Dogs* constantly to adapt their collective strategies as they interact with the Player, contributing to the interest of the game.

IV. Off-line learning

As previously stressed, our primary aim is to generate emergent interactive *Dog* behaviors worth playing against. 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 *Dogs* are themselves evolved. In the algorithm presented here, the evolving process is limited to the connection weights of the neural network.

The evolutionary procedure is as follows. Each *Dog* has a genome that encodes the connection weights of its neural network. A population of 40 (we keep this number low because of the computational cost) neural networks (*Dogs*) is initialized randomly with initial uniformly distributed random connection weights that lie within [-5, 5]. Then, at each generation:

Step 1: Every *Dog* in the population is cloned 8 times. These 8 clones are placed in the Dead End game field and play the game against a selected Player type for an evaluation period T (e.g. 125 simulation steps). The outcome of this game is to ascertain the total number of wins (W) and kills (K).

Step 2: Each *Dog* is evaluated via (6)

$$f = \alpha K - \beta W \quad (6)$$

where K and W are the total numbers of kills and wins respectively; α is the reward rate of a kill; β is the penalty rate of a win. By using (6), we promote *Dogs* (their N clones) that are able to kill the Player as many times as possible as well as to defend the Exit successfully during an evaluation period. We expect that by adjusting α and β, *Dogs* of different behaviors will be emerged.

Step 3: A pure elitism selection method is used where only the 20% 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. Alternatively, uniform and Montana and Davis [11] crossover operators have been used at this step but proved unsuccessful. The explanation is the disruptive feature of crossover operators when dealing with distributed knowledge representation (e.g. neural networks). That is, crossover among parts of different successful neural networks is very likely to lead into unsuccessful offspring [8]. Results obtained from experiments with crossover operators are not presented in this paper.

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. $g = 300$) and the best-fit *Dog's* connection weights are saved.

Dogs play for a small period (i.e. $T = 125$ simulation steps) when evaluated by the off-line learning mechanism. This evaluation procedure constitutes an approximation of the examined *Dog's* overall performance in larger evaluation periods and keeps the computational cost low.

V. On-line learning

This learning approach is based on the idea of *Dogs* that learn while they are playing against the Player. In other words, *Dogs* that are reactive to any player's behavior and learn from his strategy instead of being predictable and, therefore, uninteresting characters for game-playing.

Beginning from any initial off-line trained (OLT) group of homogeneous *Dogs*, the on-line learning mechanism attempts to transform them into a group of heterogeneous *Dogs* of high performance that are interesting to play against. The interest is produced from potential behavior alterations during the game. The on-line learning procedure is as follows. An OLT *Dog* is cloned 8 times and its clones are placed in the Dead End game field to play against a selected Player type. Then, at each generation:

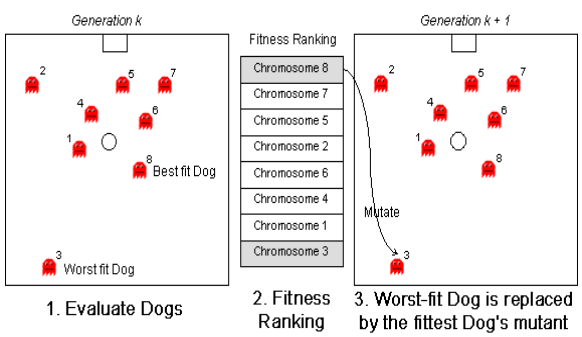

Fig. 4. The on-line learning mechanism

Step 1: Each *Dog* is evaluated every T simulation steps via (7), while the game is played; T is 25 simulation steps in this paper.

$$f' = \frac{1}{1 + \sum_{k=1}^{T}(|x_d^k - x_p^k| + |y_d^k - y_p^k|)} \tag{7}$$

where (x_p^k, y_p^k) are the cartesian coordinates of the Player's center at simulation step k. By using (7), we individually promote each *Dog* that attempts to stay as close as possible to the Player during an evaluation period.

Step 2: If the average fitness of the population is greater than a fixed threshold value then, go to Step 1 else, continue.

Step 3: A pure elitism selection method is used where only the fittest solution is able to breed. The best-fit parent clones an offspring that replaces the worst-fit member of the population. This offspring takes the worst-fit member's position in the game field.

Step 4: Mutation occurs in each gene (connection weight) exactly as in the off-line learning algorithm presented in Section IV.

The algorithm is terminated when a predetermined number of generations g is achieved (e.g. $g = 5000$) and all 8 *Dogs'* connection weights are saved. Fig. 4 illustrates the main steps of the on-line learning algorithm.

We mainly use small simulation periods (i.e. $T = 25$) to evaluate *Dogs* in on-line learning. The aim of this high frequency of evaluations is to accelerate the on-line evolutionary process. However, the evaluation function (7) constitutes an approximation of the examined *Dog'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. We, therefore, use minimal evaluation periods capable of achieving good estimation of the *Dogs'* performance.

VI. Results

In this section, we present experimental results obtained from the Dead End computer game. In subsection VI-A, we introduce a method for measuring the performance of any emergent *Dogs'* behavior. In subsection VI-B, behaviors obtained from the off-line learning mechanism are analyzed and their generalization capability is investigated. Finally, in subsection VI-C the overall effect of the on-line learning mechanism, as well as the robustness and adaptability it demonstrates, are described.

A. Performance Measurement

We introduce an efficient method for testing and comparing different learning attempts' ability to emerge successful controllers. We record the total number of both kills K and wins W of the examined team of *Dogs*, against a specific Player, by placing these agents in Dead End and letting them play the game for $12.5 \cdot 10^3$ simulation steps, since we believe it is a long enough period for testing a playing-behavior of a team of *Dogs* in an efficient way. This evaluation is called a *trial*. We, then, calculate the following performance value:

$$P = 100\frac{K}{K + W} \tag{8}$$

where P is the performance value; K and W are the total number of kills and wins in a game of $12.5 \cdot 10^3$ simulation steps respectively. This performance measurement quantifies the Player-killing (K) percentage over the total number of games played ($K + W$).

B. Off-line learning experiments

The experiment presented in this subsection is focused on producing well-behaved *Dogs* in terms of the performance

measure described in Section VI-A. We train *Dogs* against all three fixed-strategy types of Player through the off-line learning mechanism presented in Section IV. In this experiment we select $\alpha = \beta = 1$ in fitness function (6) — providing equal opportunities for promoting both Player-hunting and Exit-defensive behaviors. The off-line learning experiment is described as follows.

- Apply the off-line learning mechanism by playing against each type of Player separately. Since the off-line learning initialization phase picks random numbers for the initial connection weights' values, it constitutes an important factor for any learning attempt. Therefore, we repeat the learning attempt (run) 30 times — we believe that this number is adequate to illustrate a clear picture of the emergent behavior — with different initial conditions.
- For each run we pick the best (in terms of the optimization function used) team of homogeneous *Dogs*. Each team of *Dogs* trained against a specific type of Player is evaluated by playing against the same Player type (following the procedure described in Section VI-A). Their performance measurement is given by the average of the performance values $E\{P\}$ obtained over the 30 different trials.
- Evaluate non-evolving randomly generated (i.e. untrained) as well as Player-follower *Dogs* (i.e. Followers) against every Player type by following the procedure presented in Section VI-A (run 30 different trials and calculate their average performance). The outcome of this experiment is presented in Table I.

Experiment Parameters:

- Controller: 2 hidden layer feedforward neural network; 5 and 4 neurons in the first and second hidden layer respectively.
- Learning mechanism: population size 40; number of generations $g = 300$; evaluation period $T = 125$ simulation steps; mutation probability $p_m = 0.01$.

TABLE I

THE EFFECT OF OFF-LINE TRAINING ON THE DOGS' AVERAGE PERFORMANCE ($E\{P\}$) AND SAMPLE VARIANCE (s^2) OVER 30 LEARNING ATTEMPTS

	Untrained		Followers		OLT	
Player	$E\{P\}$	s^2	$E\{P\}$	s^2	$E\{P\}$	s^2
RM	75.58	163.98	98.54	0.13	97.80	0.41
EA	62.46	214.58	78.94	0.65	88.12	0.59
CB	17.76	288.03	71.51	5.70	94.60	1.38

As can be seen from Table I, there is a significantly large performance improvement of the OLT *Dogs* in comparison to the untrained or even the Follower *Dogs* against all three types of Player. However, in the case where Dogs are trained against the RM Player, the off-line learning mechanism fails to produce *Dogs* more effective than the Followers. This is likely because the more random the Player's strategy becomes, the more effective the Following strategy is against the neural-controlled trained *Dogs*. A Player that takes random movement decisions becomes quite unpredictable for neural controlled *Dogs* and furthermore, hard to learn to kill. This is not the case for Followers whose performance suggests that the problem becomes fairly easy when a RM Player plays the game and therefore, there is no need for a machine learning application for such playing strategies. In other words, the more advanced and effective the Player's strategy becomes (up to the most complex human playing), the harder the problem and more efficient the proposed off-line learning approach.

By supplying equal opportunities for the emergence of Player-hunting as well as Exit-defensive behaviors (i.e. $\alpha = \beta = 1$ in (6)) we come across various types of OLT *Dogs'* game-playing strategies. The most typical emergent behaviors are pure Exit-defensive or pure Player-hunting behaviors but hybrids also occur frequently. The off-line learning mechanism, in the majority of cases, produces *Dogs* that defend the Exit and/or hunt the Player in a cooperative fashion. As stressed before, opponents in this game have to learn to cooperate in order to be successful (achieve a high performance value) against any playing strategy.

1) Behavior generalization: In this subsection we present an experiment to elucidate how a *Dog* trained off-line against a Player type behaves in a game against a different Player type. This experiment gives an indication of the *Dogs'* ability to generalize against different Player strategies as well as to adapt to a new environment. The experiment is as follows.

- Use the 30 teams of OLT Dogs against each Player type presented in the experiment of Section VI-B.
- Evaluate each one of these teams of OLT *Dogs* against all three types of Player by following the procedure described in Section VI-A. Table II presents the outcome of this experiment.

TABLE II

PERFORMANCE GENERALIZATION OF DOGS TRAINED OFF-LINE

	Playing against					
	RM		EA		CB	
	$E\{P\}$	s^2	$E\{P\}$	s^2	$E\{P\}$	s^2
OLT/RM	97.80	0.41	88.06	0.54	56.09	16.21
OLT/EA	91.98	8.27	88.12	0.59	50.43	12.24
OLT/CB	91.78	14.31	74.12	0.71	94.60	1.38
Mean	93.85	15.20	83.46	45.45	67.04	395.06

According to Table II, in most cases, OLT *Dogs* against a specific Player seem to get lower average performance values when rivaling a Player other than the Player they have been off-line trained against. This is the case in OLT *Dogs* against the RM Player (appear as OLT/RM in Table II) which produce bad generalizations against the other two playing strategies. OLT *Dogs* against the EA Player manage to perform well when playing against the RM player (i.e. $E\{P\} = 91.98$); however, they perform poorly (i.e. $E\{P\} = 50.43$) when playing against the CB player.

On the other hand, *Dogs* trained off-line against CB Players showed good overall performance against all types of Players. Even though OLT/CB *Dogs* don't achieve high average performance value when playing against the EA (i.e. $E\{P\} = 74.12$) they achieve the highest mean of the average performance values against all players (i.e. the mean of the $E\{P\}$ values on each row of Table II). Therefore, among the three fixed-strategy Players, the CB Player provides the best off-line training for the opponent agents. This suggests that when *Dogs* learn from more complex and effective types of Players, they tend to generalize better.

Finally, results obtained from off-line learning experiments demonstrate the difference in effectiveness of the fixed playing strategies used. It is obvious that the RM Player (mean performance over all off-line training attempts equals to 93.85; presented in the bottom row of Table II) is the least effective and 'easiest to kill' player, whereas, the EA (83.46) is harder to kill and the CB (67.04) proves to be the most effective playing strategy of all three.

C. On-line learning experiments

The off-line learning procedure is a mechanism that attempts to produce near-optimal solutions to the problem of killing the Player and defending the Exit. These solutions will be the on-line learning mechanisms' initial points in the search for different *Dogs'* behaviors capable of achieving even higher performance values. The on-line learning experiment is described as follows.

- Apply the on-line learning mechanism (described in Section V) to all teams of OLT *Dogs* (Section VI-B) playing against each type of Player separately.
- Evaluate each on-line learning attempt against each Player type by following the procedure presented in Section VI-A. The outcome of this experiment is presented in Table III.

Experiment Parameters:

- Controller: 2 hidden layer feedforward neural network; 5 and 4 neurons in the first and second hidden layer respectively.
- Learning mechanism: number of generations $g = 5000$; evaluation period $T = 25$ simulation steps; mutation probability $p_m = 0.01$.

TABLE III

THE PERFORMANCE EFFECT OF ON-LINE LEARNING ON DOGS TRAINED OFF-LINE

	On-line learning against					
	RM		EA		CB	
	$E\{P\}$	s^2	$E\{P\}$	s^2	$E\{P\}$	s^2
OLT/RM	97.78	0.28	87.36	0.63	53.61	1.12
OLT/EA	96.93	0.33	98.40	0.15	80.31	2.93
OLT/CB	96.67	0.22	76.55	1.48	95.40	2.10
Mean	97.12	0.49	87.43	83.01	76.44	300.72

As can be seen from Table II and Table III, in nearby all cases, there is a big increase of the *Dogs'* average performance values as well as a noticeable decrease in their sample variance when on-line learning is applied to OLT *Dogs* for every Player type they are playing against. This suggests that the evolutionary approach proposed demonstrates a behavior of high robustness which furthermore manages to generate opponents' behaviors of much higher performance values.

The on-line learning mechanism tends to be a highly disruptive procedure (via the mutation operation) for high-performance group behaviors towards individual rewards. Such disruptive mutations can cause undesired drops in the performance of a team of *Dogs*. However, experiments show that *Dogs* trained by individual rewards (while playing) manage to maintain and even increase their group performance.

It is worth mentioning that these high group performances are maintained under continuously changing (on-line) *Dog* behaviors. In order to measure alterations in the *Dogs'* behavior during on-line learning, the following experiment is used. Pick the best, in terms of performance, team of OLT *Dogs* against each Player. Apply the on-line learning mechanism and calculate the average distance the *Dogs* move towards the Player in each generation. The cumulative value of these distance measures provides an effective illustration of the *Dog's* behavior during on-line learning (average distance values, over the generations, are not appropriate for presentation because of being noisy). That is, the steeper the curve is, the more aggressive (i.e. Player-hunting) the *Dogs'* behavior.

The total number of *Dog* behavior evaluation experiments is 9 (i.e. three types of OLT *Dogs* playing against three types of Players — see Table III). The mechanism demonstrated a similar behavior for all 9 different *Dog*-Player combinations but, due to space considerations, we present only three of them (see Figure 5).

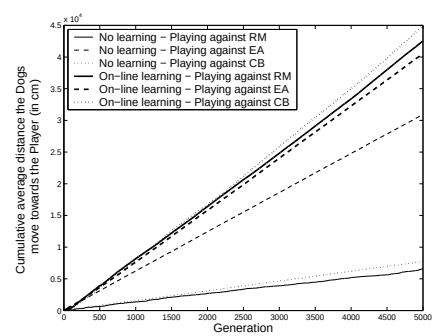

Fig. 5. On-line learning effect into the *Dog's* behavior; Initial behavior: best OLT *Dogs* against EA Player.

As seen from Fig. 5, the on-line learning mechanism alters the behavior of OLT *Dogs* against the EA Player. After 500 on-line learning generations, there is an increase of approximately 507, 17 and 367 percent in the cumulative average distance the *Dogs* move towards the Player when playing against RM, EA and CB respectively. Results obtained show that *Dogs* become more aggressive — this behavioral change occurs very fast

(i.e. from the first 50 generations) — regardless of their initial behavior or the Player they are playing against. Furthermore, *Dogs* manage to find different ways to hunt the Player as well as defend the Exit, while their overall performance is maintained or even increased.

To test the on-line mechanism's ability to adapt to a changing environment (i.e. changing player strategy), the following experiment is used. Beginning from an initial behavior trained off-line we apply the on-line learning mechanism against a specific Player type. During the on-line process we change the type of Player every 50 generations and calculate the average distance the *Dogs* move towards the Player in each generation. The sequence of the Player types is constant and predetermined. The process stops after 500 generations.

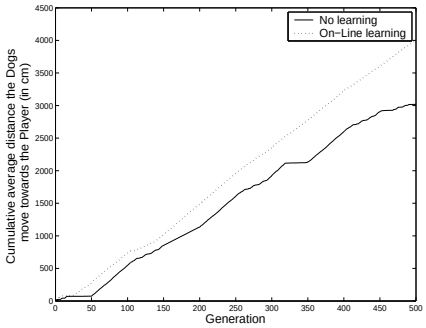

Fig. 6. Continuously changing Player type every 50 generations — in the sequence RM-EA-CB.

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 overall picture of the mechanism's behavior against any sequence of Player types. Due to space considerations we present only one (see Figure 6) out of the 6 experiments here. As seen in Figure 6, the on-line learning mechanism is able to recover from a sudden change in the player's strategy and quickly adapt to the new environment by altering the *Dogs'* behavior. The *Dogs'* average performance values, obtained over 30 trials by following the method presented in Section VI-A, at the end of the experiment are 98.69 against RM ($s^2 = 0.04$), 88.23 against EA ($s^2 = 0.37$), and 79.59 against CB ($s^2 = 2.56$). The mechanism demonstrated a similar adaptive behavior for all 6 different sequences of Player types, illustrating the mechanism's independence of the sequence of the changing Player type.

Results obtained from this experiment provide evidence for the mechanism's ability to adapt to new playing strategies. This fact makes the Dead End game far more interesting and attractive for a human player as the on-line learning mechanism produces opponents worth playing against.

VII. Conclusions

The Dead End predator/prey computer game is devised as an interesting test-bed for studying the emergence of multi-agent complex cooperative behaviors through evolutionary learning mechanisms. We introduced an off-line learning mechanism, from which effective predator behaviors have rapidly emerged. In addition, we demonstrated the influence of the player's strategy on the generalization of opponents' behaviors trained off-line.

Predator strategies in predator/prey computer games are still nowadays based on simple rules which make the game quite 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 his opponents who demonstrate interesting behaviors.

We saw that by using the proposed on-line learning mechanism, maximization of the individual simple distance measure (see (7)) coincides with maximization of the group performance. However, investigation of the heterogeneity's contribution on these results constitutes an important step for future work. Apart from being fairly robust, the proposed mechanism demonstrates fast adaptation to different types of Player (i.e. playing strategies). Therefore, we believe that such a mechanism will be able to produce interesting interactive opponents against even the most complex human playing strategy.

Acknowledgment

The authors would like to thank Jeong Keun Park for his valuable contribution to the graphical representation of the Dead End game.

References

[1] S. Woodcock, "Game AI: The State of the Industry 2000-2001: It's not Just Art, It's Engineering," Game Developer magazine, August 2001.

[2] G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou, "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, June 2003.

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

[4] J. Rosca, "Generality versus size in genetic programming," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. MIT Press, 1996, pp. 381–387.

[5] S. Luke and L. Spector, "Evolving teamwork and coordination with genetic programming," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. MIT Press, 1996, pp. 150–156.

[6] T. Haynes and S. Sen, "Evolving behavioral strategies in predators and prey," in *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, S. Sen, Ed. Morgan Kaufmann, 1995, pp. 32–37.

[7] G. Miller and D. Cliff, "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)*. Cambridge, MA: MIT Press, 1994, pp. 411–420.

[8] Yao, "Evolving artificial neural networks," *PIEEE: Proceedings of the IEEE*, vol. 87, 1999.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. for Robotics Research*, vol. 5, no. 1, pp. 90–99, 1986.

[10] D. H. Ackley and M. L. Littman, "Interactions between learning and evolution," in *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds., Sante Fe Institute Studies in the Sciences and Complexity. Reading, MA: Addison-Wesley, 1992, pp. 478–507.

[11] D. J. Montana and L. D. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kauffman, 1989.