Neuro-Evolution versus Particle Swarm Optimization for Competitive Co-evolution of Pursuit-Evasion Behaviors

Leo. H. Langenhoven, *Student Member, IEEE*, and Geoff. S. Nitschke

Abstract— **This paper presents a study that compares the efficacy of *Neuro-Evolution* (NE) versus *Particle Swarm Optimization* (PSO) for evolving *Artificial Neural Network* (ANN) controllers in an unsupervised adaptation process. The research objective is to ascertain which adaptive method is most appropriate for deriving agent behaviors in a competitive co-evolution pursuit-evasion task. This task requires one predator agent to capture one prey agent in a simulation where behavior adaptation is guided by an arms race of competitive co-evolution. Results indicate that NE was overall more effective at deriving pursuit and evasion behaviors according to the task performance measures defined for this study.**

I. Introduction

The use of competitive co-evolution to facilitate emergent behavior [15], [31] via harnessing *arms race* dynamics [5], [9] is a well explored research area in pursuit-evasion tasks [30], [24], [16] and related predator-prey simulations [4], [23], [20]. However, the application of *Particle Swarm Optimization* (PSO) as a means of agent controller adaptation within competitive co-evolution pursuit-evasion simulations has received relatively less research attention. Furthermore, there has been little research that compares PSO and *Neuro-Evolution* (NE) as an *Artificial Neural Network* (ANN) controller adaptation mechanism in pursuit-evasion tasks.

Simulated competitive co-evolution systems often attempt to replicate an *arms race* [31], [33] process in order to develop increasingly sophisticated behaviors. Various approaches to agent behavior adaptation in pursuit-evasion tasks have been studied within a competitive co-evolution context. For example, Koza [18] applied genetic programming techniques to the co-evolution of behaviors in a two-player pursuit-evasion task. In similar experiments, Reynolds [30] reported the evolution of increasingly sophisticated behaviors in the competitive co-evolution of pursuer and evader agents. Cliff and Miller [4] co-evolved behaviors in evolutionary robotics, where simulated robots co-evolved vision morphologies and behaviors as a means of improving pursuit and evasion behaviors. Floreano and Nolfi [24] evaluated a competitive co-evolution pursuit-evasion task within evolutionary robotics experiments using two mobile robots. The authors found that competitive co-evolution enabled faster evolution of more diverse pursuit and evasion behaviors, comparative to a standard evolutionary approach.

To the best of the authors' knowledge, there has not been any research that compares the efficacy of NE versus PSO for unsupervised controller adaptation in a one predator (pursuer) versus one prey (evader) competitive co-evolution pursuit-evasion simulation. Notable exceptions in closely related research include Di Gesu *et al.* [6], who compared PSO and a *Genetic Algorithm* (GA) in experiments that simulated multiple predators attempting to capture a single prey in the least amount of time. Lee *et al.* [19] compared PSO and a GA for simulating predator-prey dynamics as a means of modeling a genetic regulatory network.

Neuro-Evolution (NE), combined with a simulated competitive co-evolution process has proven to be an effective method for adapting ANN controllers that accomplish a disparate range of tasks including artificial creature design [33], multi-agent computer games [34], and pursuit-evasion using simulated and real robots [24].

Recently, competitive co-evolution versions of PSO have been elucidated as an effective means for solving a disparate range of tasks. For example, evolving agents to play board games [1], [11], [22], game theory strategies [12] and multi-objective function optimization [13], [21], [32]. Also, various versions of non co-evolutionary PSO have been applied to the unsupervised adaptation of ANN controller connection weight values [29], [25].

This paper presents, for a one pursuer, one evader task, a comparison of pursuit and evasion behaviors competitively co-evolved by PSO and NE methods. PSO and NE were selected as the comparative methods since both approaches have been successfully applied as a means of evolving connection weights in ANN controllers. However, the efficacy of both approaches have not been compared in the context of a one pursuer and one evader competitive co-evolution pursuit-evasion task.

A. Research Goals

1) To comparatively evaluate PSO versus NE as methods for evolving ANN controllers in a competitive co-evolution pursuit-evasion task.

2) To measure the diversity of the fittest evolved predator and prey ANN controllers and relate this diversity to the task performances of PSO versus NE evolved pursuit and evasion behaviors.

B. Research Hypotheses

1) Based on previous research [28]. PSO, comparative to NE, will result in overall more effective pursuit and evasion behaviors (according to task performance measures defined for this study).

Leo. H. Langenhoven, and Geoff. S. Nitschke are with the Computational Intelligence Research Group, Department of Computer Science, University of Pretoria, South Africa (email: {llangenhoven, gnitschke}@cs.up.ac.za).

2) Based on previous research [26], PSO will tend to evolve ANN controllers with a higher connection weight diversity compared to NE evolved controllers.

II. Methods

The competitive co-evolution approach tested in this study uses two populations of individuals (ANN controllers) that compete against each other, where each population aims to evolve the fittest behavior. One population adapts behaviors for predator agents, and the other adapts behaviors for prey. Within each population, behaviors compete for the role of the fittest behavior. The fittest behaviors then compete against each other in the context of the pursuit-evasion task. Each behavior is represented as a particle (PSO) or a genotype (NE). In either case, a particle or genotype is a vector of floating point values that represents the connection weights of a controller.

A. Particle Swarm Optimization (PSO)

PSO models a set of potential solutions as a swarm of particles that move about in a virtual search space. Each particle has a *position* and a *velocity* vector that is updated at each algorithm iteration. A velocity update consists of: (1) A *cognitive* component (c_1) which uses each particles personal best (*pbest*) solution, (2) A *social* component (c_2) which uses a neighborhood best solution (*nbest*), and (3) An inertia coefficient (ϕ) which slows particle velocity over time to facilitate swarm convergence. Each algorithm iteration (section II-D), a particle's velocity is added to its current position. This study used a local best *lbest* neighborhood structure [17] for the social component of the velocity update.

We used a competitive co-evolution version of the *Charged PSO* method [2] that implemented two competing swarms. For all experiments, half the particles were given a positive charge equal to one. This meant that half the particles (in a given swarm) explored new solutions while the other half exploited current solutions as a means of controller adaptation. In order to apply PSO to dynamic fitness landscapes (such as observed in competitive co-evolution [31]), the cognitive component needs to be periodically updated [3]. To accomplish this, the fittest *pbest* position vector in the swarm was periodically re-evaluated against the opposing swarm. If the resulting fitness differed by more than 1.0% (of maximum fitness) then all *pbest* vectors in the swarm were re-evaluated. This re-evaluation technique was taken from *Adaptive PSO* [3]. Although this requires more fitness evaluations at each simulation iteration, comparative to the NE method, such an approach prevents noisy fitness evaluations from disrupting the adaptation process and often gives better results [27]. Particle velocity (*VMax*) was initialized to zero and bounded by a fixed range (table I).

B. Neuro-Evolution (NE)

A competitive co-evolution version of *Conventional Neuro-Evolution* (CNE), based on that proposed by Wieland [37], was used. This method directly encoded and evolved complete controllers. That is, one genotype encodes all the parameters (input and output connection weights) of an ANN controller. After all controllers in each population have been evaluated (section II-D), recombination occurs. During recombination, each controller is systematically selected from an *elite portion* (table I) of each population and recombined with a partner controller (randomly selected from the same population). Enough child controllers are produced in order to completely replace each population. A child genotype is produced via recombing two parent genotypes using single point crossover [7], and mutation with a *Gaussian* distribution [7]. The mutation operator changes each gene (connection weight) by a random value in a given range with a fixed degree of probability (table I).

C. Predator and Prey Genotypes and Particles

The term genotype and particle both refer to a string of floating point values (*a*) that represents the connection weight values of a predator or prey ANN controller. Genotype and particle are the terms used when NE and PSO are used for adapting agent behavior, respectively. Where, *a* directly encodes a controller, and is a string of 148 (predators) and 222 (prey) floating point values. Predator controllers consist of weights fully connecting 13 sensory input neurons to eight hidden layer to four motor output neurons (figure 1: left). Prey controllers consist of weights fully connecting 17 sensory input neurons to nine layer neurons to six motor output neurons (figure 1: right). Also, there is one bias neuron for the input and hidden layers for predator and prey controllers (not illustrated in figure 1). Each connection weight is initialized to a value in a fixed range (table I), and can change to any value during the adaptation process.

D. Genotype / Particle Evaluation

For both the NE and PSO methods, each controller in a given population is systematically evaluated against 20 randomly selected opponent controllers. Opponents are selected from: (1) The opposing population, (2) A *Hall of Fame* [31], or (3) In the case of PSO, the opposing population's *pbest* position vectors. For each pairing of controllers, six pursuit-evasion games are played. The evaluation and assignment of fitness to all controllers in both populations is one iteration in the PSO and NE competitive co-evolution process.

III. Pursuit-Evasion Task

The pursuit-evasion task requires one predator agent to *capture* one prey agent. Prey capture occurs when a predator occupies the same grid cell as the prey. One predator and one prey agent are initialized to random positions in the environment, at a minimum Euclidean distance of two, and a maximum distance of 16.

A. Simulation Environment

The simulation environment is a bounded two dimensional grid of 25 x 25 cells. One predator, one prey and one food unit can occupy any *x*, *y* position. At each iteration of a pursuit-evasion game, agents can move in one of eight directions. Predators can move one grid cell per game iteration (using one energy unit). Prey can move one grid cell per game iteration (using two energy units). To give the prey an advantage, it is also able to jump a distance of two cells per game iteration, but at a cost of four energy units. Agents may also opt to stand still, which uses no energy. Each agent begins with 100 energy units. Consumption of one food unit by a prey replenishes six energy units. The predator does not consume food units. Table I presents the simulation, NE, and PSO parameter settings used for the pursuit-evasion task. The parameters ϕ, c_1, and c_2 were selected based on the study of Van den Bergh [35]. Other parameter values were derived experimentally and found to work well for this pursuit-evasion task. Minor changes to these parameter values were found to yield similar results.

The environment is populated with 70 food units, that assume one of two food distributions (for a given experiment). These distributions are referred to as *diagonal* and *corner* (table I). The diagonal distribution uniformly places 70 food units across a strip 70 grid cells wide stretching from the bottom left hand corner of the environment to the top right hand corner. The corner distribution uniformly places 100 food units across a patch of 5 x 5 cells in each corner. We elected to use a greater number of food units for the corner distribution in order to test the impact of more food units upon pursuit and evasion behaviors.

TABLE I
PSO, NE, AND SIMULATION PARAMETERS.

PSO Parameters	
Swarm Size	32
Number of PSO iterations	1500
Inertia Weight (ϕ)	0.72
Cognitive Term (c_1)	1.42
Social Term (c_2)	1.42
Velocity Max ($VMax$) / Initialization	4 / 0
Neighborhood topology	*lbest*
Neighborhood size	3
Charged PSO Parameters	
Core Radius (R_c)	10
Perception Limit (R_p)	100
Charge(Q)	1
NE Parameters	
Population size	32
Number of NE generations	1500
Mutation probability	0.05
Mutation rate per gene (σ)	1
Elite portion	0.5
Gene (weight) value initialization	[-1.0, 1.0]
ANN sensory input neurons (Predator/Prey)	13 / 17
ANN hidden layer neurons (Predator/Prey)	8 / 9
ANN motor output neurons (Predator/Prey)	4 / 6
Simulation Parameters	
Iterations per pursuit-evasion game	70
Simulation runs	20
Games played per evaluation	120
Environment width / length	25
Number of food units	70 / 100
Energy per food unit	6
Initial agent positions	Random
Initial agent energy	100
Prey movement cost	2
Prey jump cost	4
Prey jump distance	2
Predator movement cost	1
Predator / Prey movement distance	1
Sensor noise (σ)	0.1
Food distribution	Diagonal / Corner
Hall of Fame size	15
Hall of Fame update (iterations/generations)	50

IV. Predator and Prey Agents

A. Detection Sensors

Four food detection and four opponent detection sensors cover four sensor quadrants. This provides an agent with a 360 degree field of view. Each sensor quadrant is positioned at the front, back, left and right of an agent. A sensor quadrant's maximum length and width are defined as one third of the environment's width. Detection sensors are always active, and sensor values are equal to the Euclidean distance to the closest food unit or opponent.

Each food detection sensor *q*, returns the distance between *this* agent (*v*) and the location of the closest food unit in the quadrant of sensor *q*. If no food units are detected by *q* then the sensor value is equal to the maximum range of *q*.

Each opponent detection sensor *p*, returns the distance between *v* and the location of the closest opponent in the quadrant of sensor *p*. If no opponents are detected by *p* then the sensor value is equal to the maximum range of *p*.

B. Artificial Neural Network (ANN) Controller

A recurrent ANN [8] controller maps sensory inputs to motor outputs for predator and prey agents. The controller for the predator (figure 1: left), fully connects 13 sensory input neurons ([SI-0, SI-12]) to eight hidden layer neurons, to four motor outputs ([MO-0, MO-3]). The controller for the prey (figure 1: right), fully connects 17 sensory input neurons ([SI-0, SI-16]) to nine hidden layer neurons, to six motor outputs ([MO-0, MO-5]). For both controllers, input neurons [SI-0, SI-3] accept inputs from four food detection sensors. Input neurons [SI-4, SI-7] accept inputs from four agent detection sensors. Sensory input neuron SI-8 accepts the opponents last position (one of eight directions) as input.

For the predator, input neurons [SI-9, SI-12] accept motor output layer neuron activation values from the previous game iteration. For the prey, sensory input SI-9 indicates if the prey is situated on a food cell. Input SI-10 indicates the current energy of the agent, and input neurons [SI-11, SI-16] accept motor output layer neuron activation values from the previous game iteration. For both the predator and prey, hidden and output neurons are hyperbolic tangent units [14]. The number of hidden layer neurons were determined experimentally, and found to enable the evolution of effective pursuit-evasion behaviors. All sensor input values are normalized to the active range of the hyperbolic function [-1.5, 1.5], and output values are in the range [-1.0, 1.0]. After sensory input normalization Gaussian noise (table I) was applied to each sensory input value in order to simulate sensor inaccuracies.

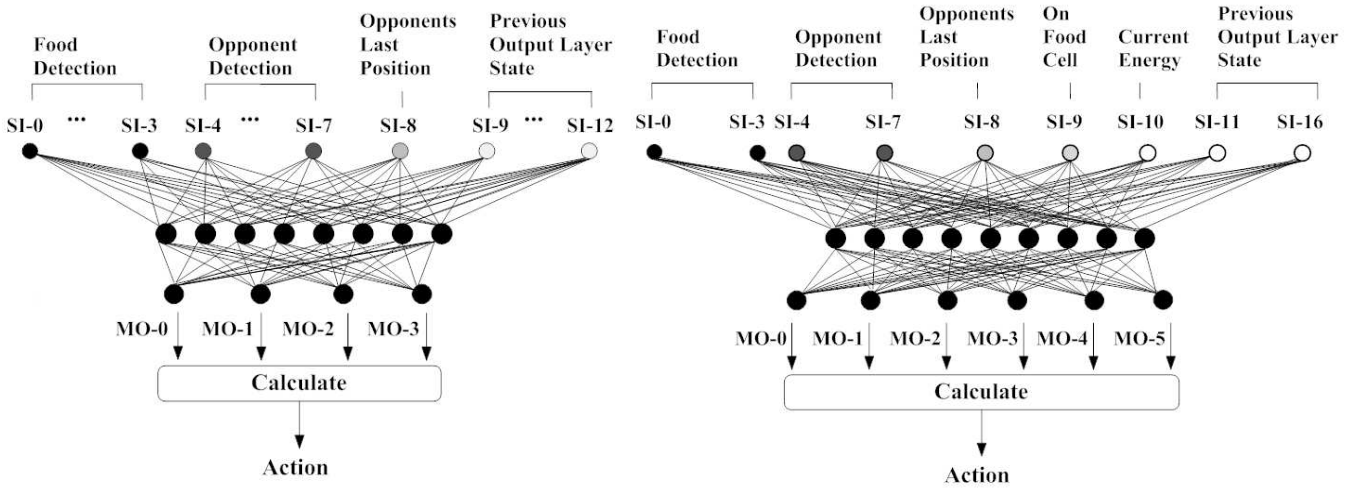

Fig. 1. Predator (left) and Prey (right) ANN controllers. *SI:* Sensory Input, *MO:* Motor Output.

C. Action Selection

At each game iteration, agents can move in one of eight directions. Predator agents are able to move one grid cell per game iteration at a cost of one energy unit. Prey agents can move one grid cell per game iteration at a cost of two energy units, or jump a distance of two at a cost of four energy units. At each game iteration agents may also opt to stand still, which uses no energy. An agent's direction of movement is calculated from the controller motor outputs MO-0, MO-1, MO-2, and MO-3 (figure 1), as follows.

- MO-0, MO-1:
  - If MO-0 $>$ 0.0, and MO-1 $<$ 0.0: then move backwards.
  - If MO-0 $>$ 0.0, and MO-1 $>$ 0.0: then move forward
  - If MO-0 $\leq$ 0.0: then stand still.
- MO-2, MO-3:
  - If MO-2 $>$ 0.0, and MO-3 $<$ 0.0: then move to the left.
  - If MO-2 $>$ 0.0, and MO-3 $>$ 0.0: then move to the right.
  - If MO-2 $\leq$ 0.0: then stand still.

The prey agent includes two additional motor output nodes that operate as follows.

- If MO-4 $>$ 0.0: If the prey is on a food cell, then the prey eats and does not move.
- If MO-5 $>$ 0.2: then the prey jumps.

D. Agent Performance Evaluation

This section details the predator and prey performance measures: (1) *Predator fitness*, (2) *Prey fitness*, (3) *Number of games won*, (4) *Food consumed* and (5) *Time to capture*. The latter two metrics are a measures of prey and predator success, respectively. The predator and prey fitness values were calculated during the PSO and NE adaptation processes, where as the other performance measures are calculated according to the following post-hoc process. First, after the completion of 20 simulation runs of the NE and PSO controller adaptation processes, the fittest PSO evolved predator and NE evolved prey are selected from the PSO and NE runs. Each of the 20 selected PSO predators is played against each of the 20 selected NE prey. Each predator and prey pairing is executed for 100 pursuit-evasion games. No controller adaptation occurs during these games. Second, the fittest NE evolved predator and PSO evolved prey are selected from the same PSO and NE runs. Each of the 20 selected NE predators is played against each of the 20 selected PSO prey. Each predator and prey pairing is executed for 100 pursuit-evasion games. Averages are calculated (over all games played) for the number of games won, food consumed and time to capture task performance measures.

1) Predator Fitness: $g_{\eta,v}$ calculates a fitness score based on the distance between predator v and the prey, and the portion of the environment explored during a game.

$$g_{\eta,v} = C + (1 - (\frac{D}{D_{max}})) + \frac{U}{I} \tag{1}$$

Where, C equals *two* if the predator v captures the prey, otherwise C equals *zero*,
D is the average distance between the predator and the prey during a game,
D_{max} is the maximum distance between the agents,
U is the number of grid cells that the predator visits during its lifetime, where no cell is counted twice,
I is the number of iterations in a game.

Predator v aims to maximize this fitness function.

2) Prey Fitness: $g_{\eta,w}$ calculates a fitness based on how many game iterations it evades capture, and the portion of the environment explored during a game.

$$g_{\eta,w} = (\frac{F}{F_{max}} \cdot 4) + (1 - (\frac{S}{I})) + \frac{I}{I_{max}} + \frac{U}{I} \tag{2}$$

Where, F is the total number of food units consumed,
F_{max} is the maximum number of available food units,

S	is the number of game iterations for which the prey was in the range of predator sensors,
I_{max}	is the maximum number of iterations in a game.

Prey agent w aims to maximize this fitness function.

3) Number of Games Won: A predator wins a pursuit-evasion game if it captures the prey during the game. A prey wins the game if it is not captured and survives the maximum number of iterations in a game (70 iterations). For each of the 100 games played for the fittest predator and prey (evolved by NE and PSO), the number of games won by the predator and the prey is recorded.

4) Food Consumed: The average number of food units consumed during a game provides an indication of the effectiveness of the fittest prey's evolved behavior. An effective prey behavior is one that allows the prey to evade capture for the maximum number of game iterations, and maximize the number of food units consumed.

5) Time to Capture: This is the average number of game iterations taken for a predator to capture the prey, and provides an indication of the efficacy of the fittest pursuit behavior. An effective pursuit behavior is one that captures the prey in the least number of game iterations. Time to capture is calculated over games that lasted for less than 70 game iterations (maximum game length), since the *games won* metric indicates predator failure (in the case of not capturing the prey after 70 iterations) and prey success (in the case of evading the predator for 70 iterations).

E. Controller Connection Weight Diversity

For simplicity we selected to use a Euclidean distance metric [26] in order to measure connection weight diversity for controllers adapted by PSO and NE. Each solution is a string of floating point values that encodes the connection weight values of the fittest predator and prey controllers evolved in each simulation run. Pairwise diversity is measured by *d*:

$$d(a,b) = \sqrt{\sum_{i=0}^{N}(a_i - b_i)^2}$$

Where, *d*(*a*, *b*) is the pairwise diversity between member *a* and member *b* (section II-C). Solution diversity for a given method is calculated as the average pairwise diversity over the 20 fittest solutions (selected from each simulation run).

V. Experiments

Experiments apply PSO and NE for co-evolving predator and prey controllers in a given environment. All experiments were implemented using the *Computational Intelligence Library* (CIlib)[1]. Each experiment executes the PSO or NE method for 20 simulation runs. A simulation is executed for 1500 iterations (generations). One iteration consists of a given controller being evaluated in 120 pursuit-evasion games against every controller in the opponent population. Each evaluation corresponds to one agent *lifetime*, or the maximum game duration (70 iterations in table I).

The first experimental objective is to maximize:

1) *Predator fitness* and the *number of games won* in environments containing a *diagonal* and *corner* food distribution (section III-A).
2) *Prey fitness* and the *number of games won* in environments containing a *diagonal* and *corner* food distribution (section III-A).

The second experimental objective is:

1) For the predator to minimize the *time to capture* (number of game iterations) the prey.
2) For the prey to maximize the number of *food units consumed* during a game.

A. Method Comparisons

Predator and prey controllers evolved by NE and PSO are compared with respect to the average fitness and number of games won. Also, the average time to capture and the amount of food consumed are used as measures for the efficacy of evolved pursuit and evasion behaviors, respectively.

1) Predator and Prey Fitness: Diagonal and Corner Food Distribution: For each simulation run, the *average* predator (*Ap*) and prey (*Ay*) fitness is calculated (every 10 iterations) over all controllers in a population. Also, the *best* predator (*Bp*) and prey (*By*) fitness is calculated. Figures 2 and 3 present the *average* of *Ap*, *Ay*, *Bp* and *By*, calculated over 20 simulation runs, for the *diagonal* and *corner* food distributions, respectively. Fitness values presented are normalized, so as a value of 1.0 indicates the highest performance, and a value of 0.0 indicates the lowest performance.

To determine if there is a statistical significance of difference between the fitness's of PSO versus NE evolved predators, and PSO versus NE evolved prey, an independent t-test [10] was applied. Statistical tests were applied to fitness data of the predator and prey evolved in the environment with a *diagonal* and the environment with a *corner* food distribution. A statistical significance of 0.05 was selected, and the null hypothesis stated that the data sets do not significantly differ. The statistical comparison found that, for environments containing the diagonal and corner food distributions, predators and prey evolved by PSO yielded a significantly higher task performance (in terms of *Ap* and *Ay* fitness) over predators and prey evolved by NE in the same environments. The comparison also elucidated that PSO predators evolved in environments containing these same food distributions, yielded a significantly higher *Bp* fitness comparative to NE. However, for both food distributions, there was no significant difference between NE and PSO in terms of *By*.

Table II presents the P values calculated for t-tests conducted for *Ap*, *Ay*, *Bp* and *By* comparisons between PSO and NE. In table II, a value of 0.0001 indicates that a value equal to or less than 0.0001 is calculated by the t-test. Values in bold indicate that the null hypothesis is accepted and that there is no significant difference between task performance

[1]The CIlib home page can be found at http://www.cilib.net/.

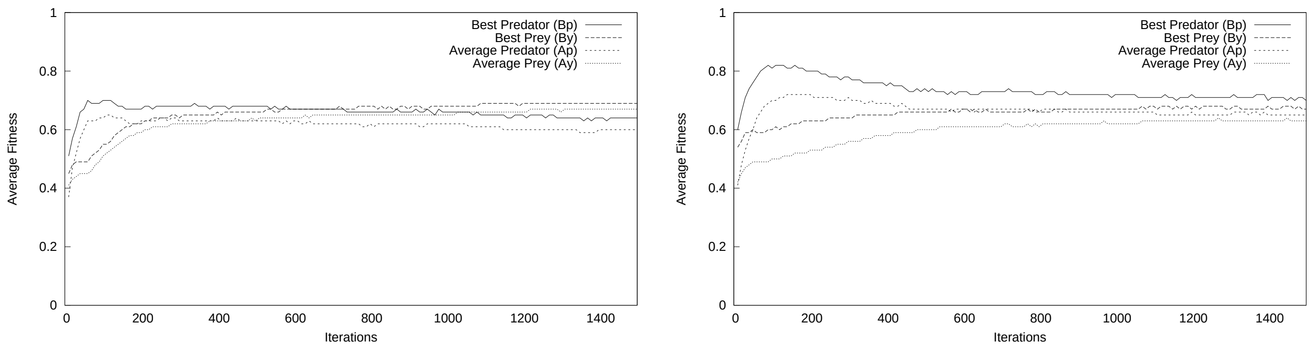

Fig. 2. Fitness of NE (left) versus PSO (right) predator and prey evolved in the environment with a *diagonal* food distribution.

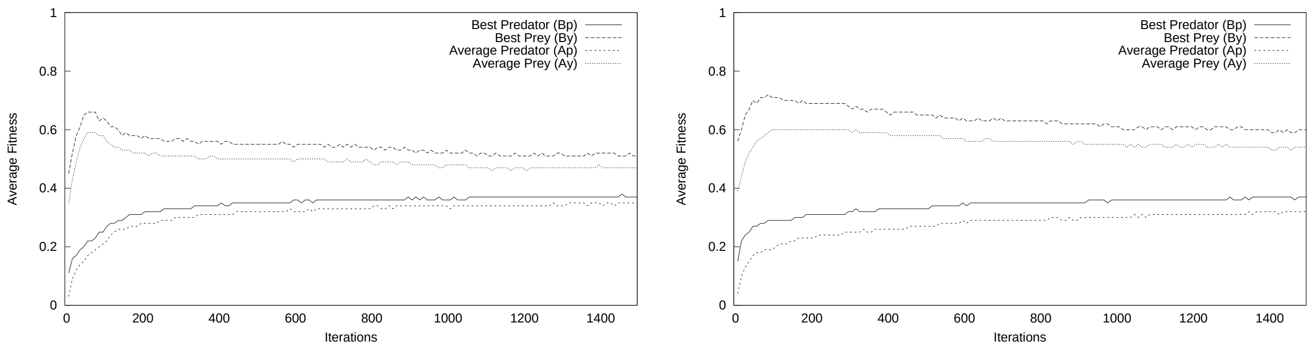

Fig. 3. Fitness of NE (left) versus PSO (right) predator and prey evolved in the environment with a *corner* food distribution.

TABLE II
T-TEST P VALUES FOR FITNESS COMPARISON OF PSO VERSUS NE EVOLVED PREDATORS AND PREY.

Diagonal Food Distribution in Environment	
PSO vs NE Evolved Predator (*Bp*)	0.0001
PSO vs NE Evolved Prey (*By*)	**0.95**
PSO vs NE Evolved Predator (*Ap*)	0.0001
PSO vs NE Evolved Prey (*Ay*)	0.0001
Corner Food Distribution in Environment	
PSO vs NE Evolved Predator (*Bp*)	0.0001
PSO vs NE Evolved Prey (*By*)	**0.27**
PSO vs NE Evolved Predator (*Ap*)	0.0001
PSO vs NE Evolved Prey (*Ay*)	0.0001

TABLE III
TASK PERFORMANCE RESULTS FOR AGENTS EVOLVED IN ENVIRONMENT WITH *diagonal* FOOD DISTRIBUTION. *NA*: NOT APPLICABLE.

NE Predators vs PSO Prey: Diagonal Distribution		
	Prey	**Predator**
Games Won (%)	67.44	41.02
Time to Capture	NA	40.22 (17.44)
Food Consumed	16.93 (7.59)	NA
PSO Predators and NE Prey: Diagonal Distribution		
	Prey	**Predator**
Games Won (%)	58.91	32.65
Time to Capture	NA	39.79 (17.49)
Food Consumed	15.83 (7.75)	NA

results. Values not in bold indicate that the null hypothesis is rejected and there is a significant performance difference.

The other task performance results (in addition to fitness) are presented in tables III and IV. Values in parentheses are standard deviations calculated over all games played. The *games won* result does not have a standard deviation since this measure denotes the percentage of fittest predators and prey (for all games played) that win games.

2) Number of Games Won: Diagonal Food Distribution: As presented in table III, the higher fitness of PSO evolved predators (figure 2) did not translate to PSO predators winning a greater portion of games, comparative to NE evolved predators. That is, NE evolved predators won 41.02% of games versus PSO evolved predators that won 32.65%. Similarly, NE evolved prey won 67.44% of games, versus PSO evolved prey that won only 58.91% of games.

3) Number of Games Won: Corner Food Distribution: Also, the higher fitness of PSO evolved predators (figure 3) did not mean these PSO predators winning a greater portion of games, comparative to NE evolved predators. That is, table IV presents that NE evolved predators won 35.75% of games versus PSO evolved predators that won 22.10%. Similarly, NE evolved prey won 78.20% of games, versus PSO evolved prey that won only 64.27% of games.

4) Time to Capture and Food Consumed: Diagonal Food Distribution: The time to capture results presented in table III indicate that there is no significant difference between the PSO and NE evolved predators. That is, differences between the average amount of time taken for pursuit behaviors to capture a prey (evolved by PSO compared to NE) were negligible. However, this result indicates that both PSO and NE evolved pursuit behaviors that, when successful in

TABLE IV

TASK PERFORMANCE RESULTS FOR AGENTS EVOLVED IN ENVIRONMENT WITH *corner* FOOD DISTRIBUTION. *NA*: NOT APPLICABLE.

NE Predators vs Prey: Corner Distribution		
	Prey	**Predator**
Games Won (%)	78.20	35.75
Time to Capture	NA	36.63 (19.62)
Food Consumed	16.84 (5.67)	NA
PSO Predators vs NE Prey: Corner Distribution		
	Prey	**Predator**
Games Won (%)	64.27	22.10
Time to Capture	NA	38.47 (19.49)
Food Consumed	14.67 (6.03)	NA

TABLE V

CONTROLLER WEIGHT DIVERSITY: 20 FITTEST AGENTS EVOLVED IN 20 SIMULATION RUNS BY NE AND PSO.

Diagonal Food Distribution in Environment	
Fittest PSO Adapted Prey	166.27
Fittest PSO Adapted Predator	177.80
Fittest NE Evolved Prey	66.45
Fittest NE Evolved Predator	60.93
Corner Food Distribution in Environment	
Fittest PSO Adapted Prey	159.17
Fittest PSO Adapted Predator	179.21
Fittest NE Evolved Prey	63.65
Fittest NE Evolved Predator	57.07

capturing a prey, did so approximately half way through a game's maximum duration. Also, no significant difference was observed for the average number of food units consumed in environments with the fittest PSO and NE evolved prey.

5) Time to Capture and Food Consumed: Corner Food Distribution: As with the time to capture results for predators evolved in the environment containing the diagonal food distribution, there was no significant difference between the time to capture for predators evolved by PSO and NE with corner food distributions (table IV). This result also indicates that both PSO and NE evolved pursuit behaviors that, when successful in capturing a prey, did so approximately half way through a games maximum duration. As was the case for the environment with diagonal food distribution, no significant difference was observed for the average food consumed in environments with the fittest PSO and NE evolved prey.

B. Performance Analysis

As stated in section IV-D, we elected to use multiple behavioral based measures, in addition to predator and prey fitness functions. This was necessary since fitness comparisons only illustrate progress and counter progress of pursuit and evasion behaviors and do not highlight if evolutionary time corresponds to *true* progress. That is, the fitness landscape of both predator and prey populations continuously change due to the Red Queen affect [36].

Results indicate that during the PSO adaptation process, pursuit and evasion behaviors were derived that yielded a significantly higher average fitness comparative to pursuit and evasion behaviors evolved during the NE adaptation process. These results held for predators and prey evolved in environments containing the *diagonal* food distribution (*Bp*, *Ap* and *Ay* in figure 2) and environments containing the *corner* food distribution (*Bp*, *Ap* and *Ay* in figure 3).

In a post-hoc analysis, where the fittest PSO and NE adapted controllers were played against each other in pursuit-evasion games, NE was found to be more effective (comparative to PSO) at deriving pursuit and evasion behaviors. The effectiveness of NE evolved controllers was supported by a greater number of *games won* in environments containing the diagonal (table III) and corner (table IV) food distributions. For both food distributions there was negligible difference between the *food consumed* by PSO and NE adapted prey. Also, there was no significant different between the *time to capture* for PSO and NE adapted predators. This indicates that both PSO and NE adapted prey derived equally effective food foraging behaviors (indicated in tables III and IV). Also, PSO and NE adapted predators derived equally effective pursuit behaviors. This is indicated by comparable time to capture values in tables III and IV. These results partially refute hypothesis 1 (section I-B) given that although predators and prey adapted during the PSO process yielded a higher fitness (according to the predator and prey fitness functions given in section IV-D), NE evolved controllers comparatively won a greater number of games.

Table V presents the diversity in controller connection weight values calculated (over 20 simulation runs) for the fittest predator and prey evolved by NE and PSO. These results indicate that the fittest PSO controllers, on average, maintain a higher weight diversity (more than double), comparative to NE evolved controllers. This higher diversity was true for environments containing both the food distributions. It is theorized that the lower controller weight diversity in the fittest NE evolved predators and prey results in the significantly lower fitness (comparative to PSO) of NE evolved predators and prey (figures 2 and 3); Where predator and prey fitness was calculated according to the fitness functions given in section IV-D. The exception was the average best prey fitness (*By*) measured for PSO and NE. This observation is supported by related work [26], and also supports the second hypothesis defined for this study (section I-B).

VI. CONCLUSIONS

This research detailed a comparative study of two unsupervised learning techniques (PSO and NE) as methods for adapting ANN controllers in a competitive co-evolution task. The task was a one predator versus one prey agent pursuit-evasion task conducted in a discrete simulation environment. The task was for the predator (pursuer) to capture the prey (evader) as quickly as possible, and for the prey to evade capture for the duration of a pursuit-evasion game. Results indicated that pursuit and evasion behaviors adapted during the PSO process yielded a higher average fitness comparative to those evolved by NE. The fittest controllers evolved by PSO also maintained a higher controller weight diversity comparative to those evolved by NE. However, NE evolved pursuit and evasion behaviors won a greater number of pursuit-evasion games when played in a set of post controller adaptation games.

Future work intends to investigate the relationship between high controller connection weight diversity in PSO solutions, and the lower diversity but overall more effective NE evolved behaviors, that were observed in this research. Also, we intend to extend the current one predator and one prey competitive co-evolution task to an n-predator and n-prey collective behavior task in order to examine the impact of cooperative and competitive co-evolution on PSO and NE as controller design methods.

References

[1] A. Abdelbar, O. Soliman, S. Kinawy, and H. Sayed. An evolved seega player capable of strong novice-level play. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, pages 332–336, Montreal, Canada, 2005. IEEE Press.

[2] T. Blackwell and P. Bentley. Dynamic search with charged swarms. In *Proceedings of the Congress on Evolutionary Computation*, pages 19–26, Honolulu, USA, 2002. IEEE Press.

[3] A. Carlisle and G. Dozler. Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of the 5th Biannual World Automation Congress*, pages 265–270, Orlando, USA, 2002. IEEE Press.

[4] D. Cliff and G. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Advances in Artificial Life*, pages 200–218, Heidelberg, Germany, 1995. Springer.

[5] R. Dawkins and J. Krebs. Arms races between and within species. *Proceedings of the Royal Society of London B*, 205(1):489–511, 1979.

[6] V. Di Gesu, B. Lenzitti, G. L. Bosco, and D. Tegolo. Comparison of different cooperation strategies in the prey-predator problem. In *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, pages 108–112, Paris, France, 2007. IEEE Press.

[7] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Germany, 2003.

[8] J. Elman. Finding structure in time. *Cognitive Science*, 14(1):179–211, 1990.

[9] S. Ficici and J. Pollack. Challenges in coevolutionary learning: Arms race dynamics, open endedness, and mediocre stable states. In *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, Cambridge, USA, 1998. MIT Press.

[10] B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes*. Cambridge University Press, Cambridge, 1986.

[11] C. Franken and A. Engelbrecht. Comparing pso structures to learn the game of checkers from zero knowledge. In *Proceedings of the Congress on Evolutionary Computation*, pages 234–241, Canberra, Australia, 2003. IEEE Press.

[12] C. Franken and A. Engelbrecht. Pso approaches to co-evolve ipd strategies. In *Proceedings of the Congress on Evolutionary Computation*, pages 356–363, San Diego, USA, 2004. IEEE Press.

[13] C. Goha, K. Tan, D. Liu, and S. Chiam. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, 202(1):42–54, 2009.

[14] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, 1991.

[15] D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1):228–234, 1990.

[16] F. Jiancong, R. Jiuhong, L. Yongquan, and T. Leiyu. A pso solution for pursuit-evasion problem of randomly mobile agents. In *Proceedings of the 2009 Chinese Control and Decision Conference*, pages 4032–4035, Guilin, China, 2007. IEEE Press.

[17] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the Congress on Evolutionary Computation*, pages 1671–1676, Honolulu, USA, 2002. IEEE Press.

[18] J. Koza. Evolution and co-evolution of computer programs to control independent-acting agents. In *Proceedings of the international conference on the simulation of adaptive behaviour*, pages 366–375, Paris, France, 1991. MIT Press.

[19] T. Lee, C. Yeh, and S. Doong. A comparative study of genetic network modeling using predator-prey system. In *Proceedings of the Second International Conference on Innovative Computing, Information and Control*, pages 167–175, Paris, France, 2007. IEEE Press.

[20] X. Li and S. Sutherland. A real-coded cellular genetic algorithm inspired by predator-prey interactions. In *Recent Advances in Simulated Evolution and Learning: Advances in Natural Computation*, pages 191–207. Kluwer, Washington DC, USA.

[21] S. Lu and C. Sun. Quantum-behaved particle swarm optimization with cooperative-competitive coevolutionary. In *Proceedings of the International Symposium on Knowledge Acquisition and Modeling*, pages 593–597, Wuhan, China, 2008. IEEE Press.

[22] L. Messerschmidt and A. Engelbrecht. Learning to play games using a pso-based competitive learning approach. *IEEE Transactions on Evolutionary Computation*, 8(3):280–288, 2004.

[23] S. Nishimura and T. Ikegami. Emergence of collective strategies in a prey-predator game model. *Artificial Life*, 3(1):243–260, 1997.

[24] S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do arm races arise in artificial evolution. *Artificial Life*, 4(4):311–335, 1999.

[25] J. Pugh and A. Martinoli. Multi-robot learning with particle swarm optimization. In *Proceeding of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 441–448, Hakodate, Japan, 2006. ACM Press.

[26] J. Pugh and A. Martinoli. Parallel learning in heterogeneous multi-robot swarms. In *Proceedings of the Congress on Evolutionary Computation*, pages 3839–3846, Singapore, 2007. IEEE Press.

[27] J. Pugh and A. Martinoli. Distributed adaptation in multi-robot search using particle swarm optimization. In *Proceeding of Simulated Adaptive Behavior*, pages 393–402, Osaka, Japan, 2008. Springer.

[28] J. Pugh and A. Martinoli. An exploration of online parallel learning in heterogeneous multi-robot swarms. In *Design and Control of Intelligent Robotic Systems*, pages 133–151. Springer, Berlin, Germany, 2009.

[29] J. Pugh, Y. Zhang, and A. Martinoli. Particle swarm optimization for unsupervised robotic learning. In *Proceeding of the Swarm Intelligence Symposium*, pages 92–99, Pasadena, USA, 2005. IEEE Press.

[30] C. Reynolds. An evolved, vision-based behavioral model of coordinated group motion. In *Proceedings of the international conference on the simulation of adaptive behaviour*, pages 384–392, Cambridge, USA, 1993. MIT Press.

[31] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[32] Y. Shi and R. Krohling. Co-evolutionary particle swarm optimization to solve min-max problems. In *Proceedings of the Congress on Evolutionary Computation*, pages 1682–1687, Honolulu, USA, 2002. IEEE Press.

[33] K. Sims. Evolving 3d morphology and behavior by competition. In *Proceedings of Artificial Life IV*, pages 28–39, Cambridge, USA, 2004. MIT Press.

[34] K. Stanley, B. Bryant, and R. Miikkulainen. Real-time neuro-evolution in the nero video game. *Evolutionary Computation*, 9(6):653–668, 2005.

[35] F. van den Bergh. *An analysis of particle swarm optimizers. PhD Thesis*. University of Pretoria, Pretoria, South Africa, 2002.

[36] L. VanValen. A new evolutionary law. *Evolution Theory*, 1(1):1–30, 1973.

[37] A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks*, pages 673–667, Seattle, WA, USA, 1991. IEEE Press.