Neuro-evolution Methods for Designing Emergent Specialization

Geoff S. Nitschke

Computational Intelligence Group, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands,
nitschke@cs.vu.nl

Abstract. This research applies the *Collective Specialization Neuro-Evolution* (CONE) method to the problem of evolving neural controllers in a simulated multi-robot system. The multi-robot system consists of multiple pursuer (predator) robots, and a single evader (prey) robot. The CONE method is designed to facilitate behavioral specialization in order to increase task performance in collective behavior solutions. Pursuit-Evasion is a task that benefits from behavioral specialization. The performance of prey-capture strategies derived by the CONE method, are compared to those derived by the *Enforced Sub-Populations* (ESP) method. Results indicate that the CONE method effectively facilitates behavioral specialization in the team of pursuer robots. This specialization aids in the derivation of robust prey-capture strategies. Comparatively, ESP was found to be not as appropriate for facilitating behavioral specialization and effective prey-capture behaviors.

1 Introduction

Design principles that facilitate emergent behavioral specialization have been studied in artificial life [10] and multi-robot systems [1] research. However, collective behavior design methods for harnessing and utilizing emergent specialization for the benefit of problem solving are currently lacking.

Pursuit-evasion is a collective behavior task that is commonly used within artificial life research to test both non-adaptive (typically game theoretic) and adaptive (typically learning and evolution) methods for agent controller design. This paper describes a pursuit-evasion game, where a team of pursuer robots (herein called: *predators*), are required to collectively immobilize one evader robot (herein called: *prey*). The paper compares two different *Neuro-Evolution* (NE) methods, the CONE and ESP methods, for designing effective collective prey-capture behaviors. The effectiveness of emergent prey-capture behaviors are examined with respect to the role of behavioral specialization.

Research Goal: To demonstrate that the CONE method is appropriate for deriving behavioral specialization in a team of predators, where such specialization gives rise to successful collective prey-capture behaviors. Success of prey-capture behaviors is measured in terms of the simulation time for which a prey is immobilized (captured).

Hypothesis 1: CONE will facilitate emergent behavioral specialization that will enable the derivation of high performance prey-capture behaviors.

Hypothesis 2: The ESP method is not as appropriate for facilitating behavioral specialization in the pursuit-evasion task, and will thus yield comparatively low performance prey-capture behaviors.

Behavioral Specialization: Using methods from related work [8] we were able to ascertain which sensory activation and motor output value ranges corresponded to an observed behavior. Specifically, we measured the portion of a predators lifetime that light sensors (for prey detection) and infrared sensors (for proximity detection) were activated (within a given range). That is, sensor and motor activations of individual predators, within a given range of values, were found to produce specific observed behaviors. In some cases, these specific behaviors collectively produced an effective prey-capture behavior. Sensory-motor activation instances that had been identified with an observed behavior were summed over the course of a predators lifetime. If the sum of these particular activation instances was $\geq 50\%$ of the predators lifetime (that is: the total amount of activation instances possible) the corresponding observed behavior was labeled as *specialized*.

Task: The task was for a predator team to maximize the time for which a prey is immobilized. A control experiment (described in related work [7]) demonstrated that at least two predators are required to immobilize a prey.

Team Fitness Calculation: Predator teams were evaluated according to the total time for which the team was able to immobilize a prey. Specifically, a global fitness function calculated the average time for which a prey was immobilized. This average was calculated over the lifetime of a given predator team, as well all experimental runs. A fitness estimation method known as *fitness sharing* [2] was used in this calculation. This assumed that each predator in the team contributed equally to the capture of a prey, and thus each predator received an equal fitness reward when a prey was immobilized. Specifically, each predator in the team received a reward equal to the time for which a prey was immobilized.

2 Neuro-evolution Methods

2.1 CONE: Collective Neuro-evolution

CONE is an extension of both the SANE [6] and ESP [5] methods. A key difference between CONE and other NE methods is that it creates n separate genotype (neuron) sub-populations for n neural controllers operating in the task environment, and is thus best suited for collective behavior tasks. One advantage of CONE is that it expedites artificial evolution, given that the genotype population is organized into sub-populations. Hence, specialized controllers do not have to emerge out of a single population of neurons, and progressive specialization of controllers is not hindered by recombination of controllers with complementary specializations. A second advantage is that it provides more genotype diversity (comparative to single genotype population methods) and encourages emergent controller specialization given that evolution occurs within separate genotype

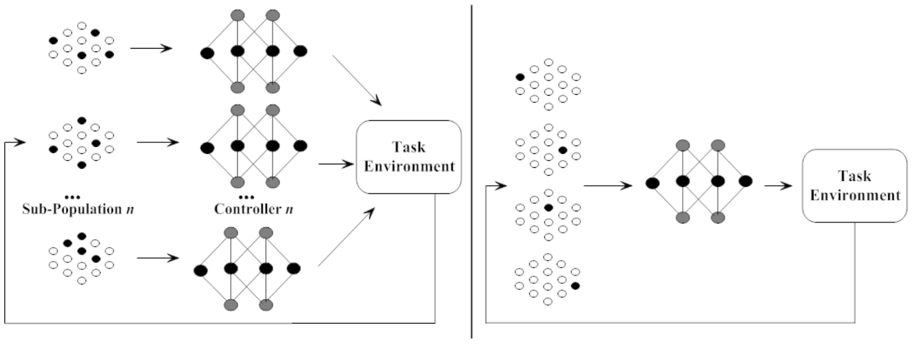

Fig. 1. Left: Example of *Collective Neuro-evolution* (CONE) Method (Section 2.1). **Right:** Example of *Enforced Sub-Populations* (ESP) method (Section 2.2).

sub-populations. Organizing the genotype population into separate niches (sub-populations), either dynamically [10], or *a priori* [9] facilitates specialization, and protects emergent behaviors within specialized niches of the genotype space.

CONE Process. After each of the n sub-populations was initialized with m genotypes, the CONE process (figure 1) was executed as follows.

1. n predator neural network controllers are constructed via selecting p genotypes (neurons) from each sub-population. Each set of p neurons then becomes the hidden layer of each of the n predator neural network controllers.
2. These n controllers are then tested together in a task environment for a *lifetime* of q epochs. An epoch is a test scenario lasting for w iterations of simulation time. Each epoch tested different predator and prey starting positions and orientations in the environment. For each of the q epochs ($q \geq m$, where m is the number of genotypes in a sub-population), each genotype in a given sub-population was selected and tested in combination with p-1 other neurons (in the form of a complete controller) randomly selected from the same sub-population.
3. p neurons from each of the n sub-populations were concurrently evaluated in the task environment and assigned a fitness. Testing of neurons within each sub-population continued until all neurons had been tested at least once.
4. At the end of a predators lifetime (q epochs) an average fitness value was assigned to each of the p neurons that participated in each controller. The average fitness of each neuron was calculated as its cumulative fitness divided by the number of controllers it participated in.
5. The testing and evaluation of the m neurons in each predators genotype sub-population constituted one generation of the CONE process.
6. For each sub-population, recombination and mutation of the fittest 20% of neurons then occurred, where the fittest 20% were arranged into pairs of neurons, and each pair produced 5 child neurons, so as to replace all genotypes in the current sub-populations and propagate the next generation of each sub-population.

7. p neurons were randomly selected from the fittest 20% within each of the n sub-populations. These n sets of p neurons were then decoded into n controllers, and executed in the task environment as the next generation. This process was then repeated for r generations.

2.2 ESP: Enforced Sub-Populations

Enforced Sub-Populations (ESP) has been effectively applied to non-Markovian control tasks with sparse reinforcement such as double pole balancing, rocket control, as well as pursuit-evasion games [5]. ESP differs from other NE methods in that it allocates and evolves a separate neuron population for each of the p hidden-layer units in a neural network. A neuron can only be recombined with other neurons from its own sub-population, thus making it suitable for the evolution of recurrent neural networks (not the case for the SANE method [6]).

2.3 CONE and ESP: Common Methods

Constructing Neural Network Controllers: In the case of ESP, 1 genotype is selected from each of p=6 sub-populations (table 1), for deriving a neural network controller's hidden layer of p neurons [5]. This is repeated n times for n controllers. There were p sub-populations for n predators, where p equaled the number of hidden layer neurons in a controller. In the case of CONE, complete controllers were constructed via selecting p neurons from each of the n sub-populations (table 1). There were n sub-populations for n predators. Each neuron in each sub-population was assigned to a fixed position in the hidden layer of any given controller. The position that the *ith* neuron (g_i) would take in a hidden layer of p neurons, where g_i was selected from any sub-population of m neurons, was calculated as follows. Each of the m neurons in a sub-population were initially assigned a random and unique ranking in the range [0, m-1]. A sub-population was divided into approximately equal portions (m / p), where if g_i was within the *kth* portion (where: k = [1, p]) then g_i would adopt the *kth* position in the hidden layer. Given that recurrent neural networks were being evolved, CONE only recombined neurons that were in the same sub-population (section 2.1), and assigned to the same hidden layer position.

Recombination and Mutation of Genotypes: Genotypes were encoded as a string of floating point values (table 1), which represented neural network weights connecting all sensory input neurons and all motor output neurons to a given hidden layer neuron. Child genotypes were produced using single point crossover, and *Burst* mutation with a *Cauchy* distribution [5]. Mutation of a random value in the range [-1.0, +1.0] was applied to each gene (connection weight) with a 0.05 degree of probability, and weights of each genotype were kept within the range [-10.0, +10.0] (table 1). Burst mutation was used so that most weight changes were small whilst allowing for larger changes to some weights.

3 Experimental Design, Agents, and Environment

3.1 Experimental Design

In this case, experiments measured the impact of a *neuro-evolution method* and a *group types* upon *prey capture time.*

– *Neuro-evolution Method*: Each predator used a recurrent neural network controller which was adapted with either the ESP or CONE method.
– *Group Type*: Between 2 and 6 predators were tested with 1 prey. These 5 group types (GT) were defined as follows: *GT-1:* 2 predators, *GT-2:* 3 predators,*GT-3:* 4 predators, *GT-4:* 5 predators, *GT-5:* 6 predators.

Table 1. Neuro-evolution parameter settings for the ESP and CONE methods.

CONE and ESP Neuro-evolution Parameter Settings	
Runs per experiment	50
Epochs	50
Iterations per epoch	1000
Mutation probability	0.05
Evaluations per neuron	10
Mutation type	Burst mutation / Cauchy distribution
Mutation range	[-1.0, +1.0]
Weight range	[-10.0, +10.0]
Crossover	Single point
Sensory input neurons	22
Hidden layer neurons	6
Motor output neurons	2
Phenotypes	[2, 6] Recurrent neural networks
Genotype sub-populations	[2, 6] / 6 (ESP)
Genotype representation	Neuron
Genotype length	24
Genotypes	600 per sub-population / 100 per sub-population (ESP)

Each experiment measured prey capture time, given a group type and neuro-evolution method. Prey capture time was measured as an average calculated over the multiple test scenarios that constituted a predator team's lifetime.

3.2 Environment

Predators and the prey move within a discrete environment of 180 x 180 quadrants. Each quadrant is large enough to contain only one predator or prey. If two or more robots attempt to occupy the same quadrant, a collision occurs. As a course encoding of movement, a predator or prey could turn at any angle (in 45 degree increments) up to 180 degrees, either to the left or to the right, with respect to its current heading. Each robot is initialized with a random heading.

A difference calculation in wheel speeds (MO_0 and MO_1) controlled the orientation of any given predator or prey. Obstacles are detected at a maximum range of 4 quadrants with a 360 degree field of detection (figure 2). This field of detection was an area of 9 x 9 quadrants, which is divided into 8 sectors to account for the coverage of different infrared proximity (for predators and prey) or light (for predators only) sensors. If an obstacle was detected on a quadrant divided by two sectors, that is, covered by two proximity (or light) sensors, then both sensors are simultaneously activated (each receiving an equal activation value).

3.3 Predators and Prey: Sensors and Actuators

The sensor and actuator configuration of each predator and prey is assumed to be that of a Khepera mobile robot [8] (figure 2). The prey is equipped with a light on its top (L_0). This light could be detected by the predator light sensors, and was used so each predator could distinguish fellow predators from the prey. Both predators and prey are equipped with 8 infrared proximity sensors ([SI_0, SI_7]. Additionally, each predator is equipped with 8 light ([SI_8, SI_{15}]) sensors, positioned on its periphery. Both, predators and prey are provided with two wheel motors (MO_0, MO_1) that controlled their speed and orientation.

When an obstacle came within range of a given proximity sensor, that sensor was activated with a value proportional to the distance to the obstacle. When a prey came within range of a predators light sensor, that sensor was activated with a value proportional to the distance to the prey. Sensor values were normalized within the range [0.0, 1.0] for the purposes of being acceptable as neural network inputs. Motor output values were normalized within the range [-10.0, 10.0].

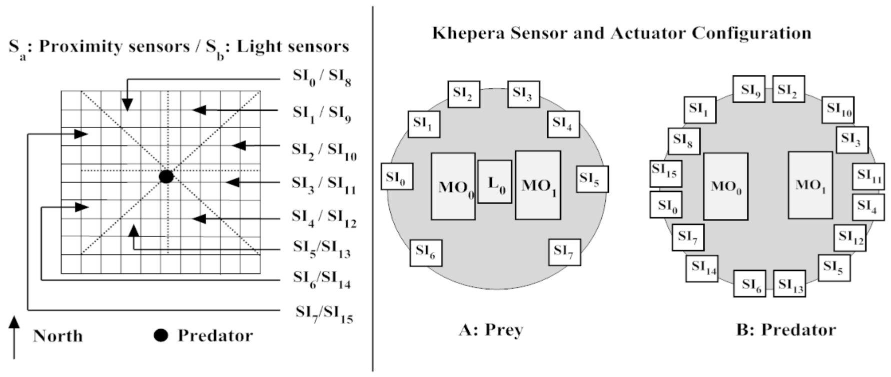
S_a: Proximity sensors / S_b: Light sensors
SI0 / SI8
SI1 / SI9
SI2 / SI10
SI3 / SI11
SI4 / SI12
SI5/SI13
SI6/SI14
SI7/SI15
North
● Predator
Khepera Sensor and Actuator Configuration
SI0 SI1 SI2 SI3 SI4 SI5 SI6 SI7
MO0 L0 MO1
A: Prey
SI0 SI1 SI2 SI3 SI4 SI5 SI6 SI7 SI8 SI9 SI10 SI11 SI12 SI13 SI14 SI15
MO0 MO1
B: Predator

Fig. 2. Left: Sensory range of proximity and light sensors in the discrete simulation environment (prey is limited to proximity sensors). **Right:** Sensor and actuator configuration for predator (A) and prey (B) Khepera robots. See section 3.3 for details.

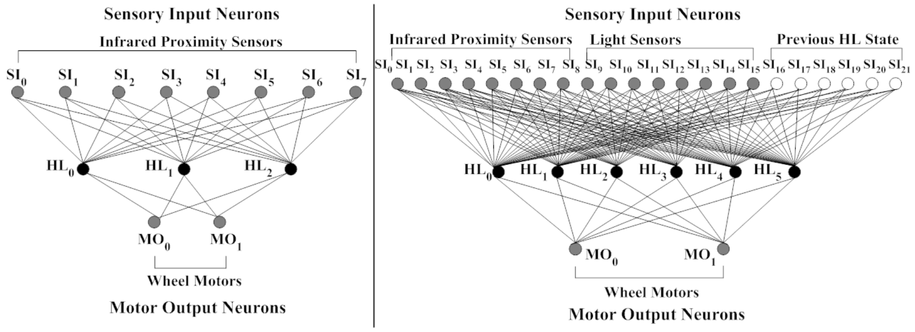

Fig. 3. Left: Prey feed-forward neural network controller. **Right:** Predator recurrent neural network controller. See section 3.4 for details.

3.4 Predators and Prey: Neural Network Controllers

The prey neural network consists of 8 sensory input neurons and 2 motor output neurons, fully connected to a hidden layer of 3 neurons (figure 3). Prey sensory inputs encode the state of 8 infrared proximity sensors, and 2 motor outputs encode the speed of 2 wheels. The output values of the 2 motor neurons are multiplied by 1.2. This sets a prey's speed to be 20% faster than the predators. The prey controller was evolved for static and dynamic obstacle avoidance, before being tested with a predator team. A recurrent neural network was selected as the predator controller in order to emulate short term memory [3]. A hidden layer of 6 sigmoidal units fully connects 22 sensory input neurons to 2 motor output neurons (figure 3). Predator sensory input neurons encode the state of 8 infrared proximity sensors and 8 light sensors ([SI_0, SI_{15}]), as well as previous hidden layer ([SI_{16}, SI_{21}]) activation values. Motor outputs (MO_0, MO_1) encode the speed of the 2 wheels. Further details are presented in related work [7].

3.5 Evolution of Predator Controllers

The CONE and ESP methods were applied to n predator neural network controllers. The goal was to adapt controllers over the course of evolutionary time so as to derive collective prey-capture behaviors that maximize prey-capture time. The parameter settings used for the CONE and ESP methods are presented in table 1. These parameters were selected given the success of related parameter settings in previous evolutionary robotics experiments [8].

CONE: When the CONE method was applied to evolve predator controllers, between 2 and 6 genotype sub-populations were created. These sub-populations represented the genotype space of between 2 and 6 predator neural network controllers. Each sub-population was initialized with 600 genotypes.

ESP: When the ESP method was applied to evolve predator controllers, 6 genotype populations were created so as to represent the genotype space of the 6 hidden layer neurons in a predator neural network controller. Each genotype population was initialized with 100 genotypes.

Genotypes (Neurons). For both CONE and ESP, each genotype represented the connection weights of a neuron that potentially participated in the hidden layer of any given predators neural network. Each genotype was encoded as vector of 24 floating point values. That is, 22 input connection weights plus 2 output connection weights. In order to construct a single predator neural network controller, 6 neurons were selected from a given genotype sub-population (1 genotype from each of the 6 sub-populations in the case of ESP). This set of neurons then constituted the hidden layer of 1 predator neural network. The process was then replicated between 2 and 6 times for each predator in the team.

4 Results: Evolved Collective Prey-Capture Behavior

Collective prey-capture behaviors, utilizing at least 3 predators and at most 4 predators, consistently emerged in the later stages of both the ESP and CONE evolutionary processes (> 200 generations). These collective behaviors were termed *role-switcher* and the *pursuer-blocker*. The former emerged under ESP, where as, both emerged under CONE. Figure 6 presents the average prey capture time, and number of instances of emergent specialization (corresponding to prey capture behaviors) for all group types.

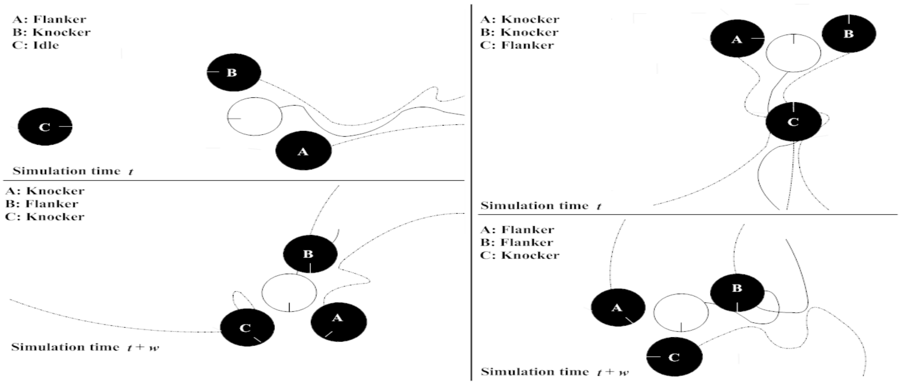

Fig. 4. Two versions of the *role switcher* prey capture behavior using 3 predators.

4.1 ESP: Role-Switcher Prey Capture Behavior

Two versions of the role switcher collective behavior emerged. In each case, predators in the team adopted 1 of 3 specialized behavior roles, termed: *flanker*, *knocker* and *idle*. However, at least 1 predator adopted 2 of these behavioral roles, which it would switch between in order to maintain the role-switcher prey-capture behavior. The role switcher behavior was most effective for teams of 3 or 4 predators. Teams of 2 predators were insufficient to immobilize a prey for more than a few simulation iterations, and teams of 5 and 6 predators often caused physical interference between each other, and thus caused the prey-capture behavior to fail. Figure 4 (left and right hand side) illustrate the two versions of the role switcher prey-capture behavior, occurring over w simulation time steps. The role switcher behavior has been observed in related research [7], and is thus not elaborated upon here.

4.2 CONE: Pursuer-Blocker Prey Capture Behavior

In addition to the role-switcher behavior, a prey capture behavior called *pursuer-blocker* also emerged under the CONE method. Two versions of the pursuer-blocker collective behavior emerged. Predators assumed 1 of 2 specialized behavioral roles, termed, *pursuer* and *blocker*. Pursuer-blocker was most effective for teams of 3 or 4 predators. Teams of 5 or 6 predators were ineffective due to physical interference that occurred between the predators as they collectively approached the prey, and 2 predators were sufficient for immobilizing the prey for a few simulation iterations only.

Figure 5 (left hand side) illustrates an example of the first pursuer-blocker behavior. Predators A and B are the pursuers, assuming positions behind and to either side of the prey. Predator C assumes the role of the blocker. When the prey moves within light sensor range of predator C, it moves directly towards the prey. Consequently the prey turns to avoid predator C, however its evasion is stopped by one of the pursuing predators. The result is that the prey becomes immobilized between the 3 predators. This pursuer-blocker behavior depended upon at least 2 and at most 3 predators assuming the roles of pursuers. Pursuers needed to maintain a close enough distance to the prey, so as the prey could not escape when it changed its direction of movement. Furthermore the blocker needed to move directly towards the prey when the prey came within its light sensor range. Figure 5 (right hand side) illustrates an example of the second *pursuer-blocker* behavior using 3 predators. The 3 predators assume the pursuer role, pursuing a prey that is moving towards a corner in the environment. When the prey comes close to the corner it turns to avoid the walls, however, such a turn places it in the path of one of the pursuers. The result is that the prey becomes immobilized between the corner and the 3 predators. A prerequisite for the success of this behavior was that the prey be moving towards a corner. This pursuer-blocker behavior also emerged using 2 or 4 predators, but failed with 5 and 6 predators due to interference that occurred between predators as they collectively approached the prey in a corner.

5 Analysis and Discussion

To draw conclusions from this comparative study, a set of statistical tests were used to gauge respective differences between CONE and ESP method results. First, the data distributions for *prey-capture time* results yielded by CONE and ESP, were determined to be normal distributions via applying the Kolmogorov-Smirnov test [4] (P=0.72 and P=0.98, respectively). We then applied an independent t-test [4]; 0.05 was selected as the threshold for statistical significance,

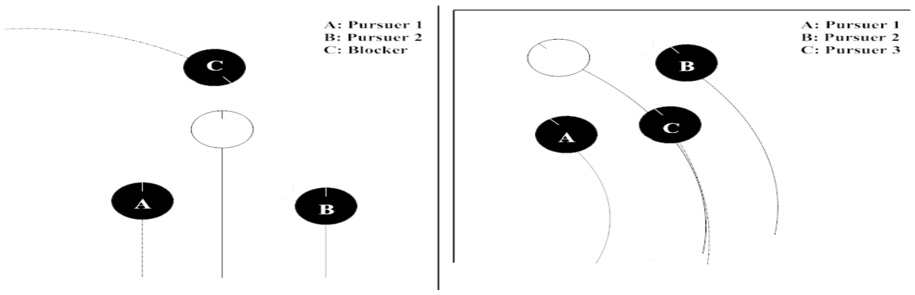

Fig. 5. Two versions of the *pursuer-blocker* prey capture behavior using 3 predators.

and the null hypothesis stated that the two data sets did not significantly differ. P=0.00012 was calculated, meaning the null hypothesis was rejected. This partially supported our first and second hypotheses. That is, CONE (and not ESP) would derive prey-capture behaviors with superior task performance. A t-test was then applied to the *instances of specialization* data sets for the CONE and ESP methods; P=0.00025 was calculated, meaning the null hypothesis was rejected. This further supported our first and second hypotheses. That is, the high task performance exhibited under CONE (and not ESP) was due to emergent specialization. To completely support our hypotheses, it is necessary to compare the group types where prey capture time and the number of specialization instances are highest for CONE (figure 6). Instances of emergent specialization were calculated as the number of epochs in a predators lifetime (averaged for all predators in the team) that were labeled as *specialized* (section 1). As illustrated in figure 6 the largest differences between the average number of emergent specialization instances, for CONE and ESP, were for group types 2 and 3. Group types 2 and 3, also yielded the largest difference in prey-capture times. That is, under the ESP method a low number of specialization instances corresponded to low prey-capture times for these group types. It is theorized that the superior performance of prey-capture behaviors derived by CONE, was due to its capability to facilitate more instances of behavioral specialization. Subject to future research, this was attributed to the use of separate neuron sub-populations for deriving complete controllers. This encouraged the derivation of specialized controllers that complemented each other in the task environment.

6 Conclusions

This paper described a comparative study of the ESP and CONE neuro-evolution methods applied to the task of deriving effective collective prey-capture behaviors in a pursuit-evasion game. Prey capture behaviors were evolved for a team of simulated predator robots that attempted to immobilize (capture) a prey robot. The effectiveness of prey-capture behaviors, and hence the fitness of the predator team was measured in terms of prey-capture time. Results indicated, that

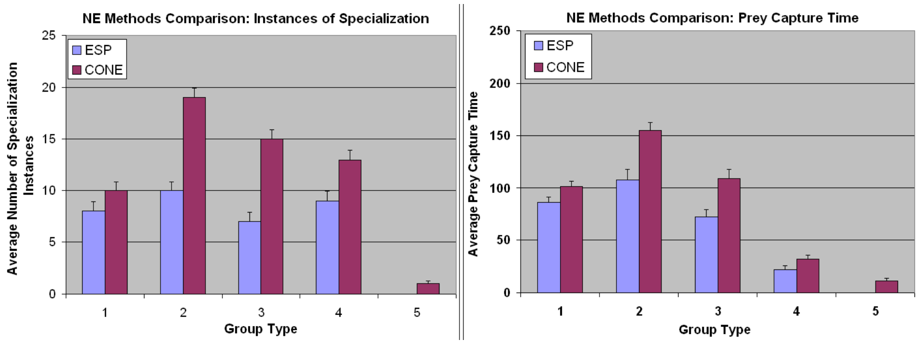

Fig. 6. Neuro-evolution (CONE and ESP) methods comparison. **Left:** Average number of emergent specialization instances. **Right:** Average prey capture time.

the CONE method facilitated emergent behavioral specialization in the predator team. These specialized behavioral roles served to increase the effectiveness of prey-capture behaviors. Comparatively, less instances of emergent specialization were observed when the ESP method was applied for the purpose of deriving collective prey-capture behaviors in a predator team. Given this, and the inferior performance of prey-capture behaviors derived under the ESP method, it is concluded that emergent behavioral specialization is beneficial in the pursuit-evasion task, and that CONE is appropriate for facilitating such specialization.

References

1. G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behavior. *Artificial Life*, 9(1):255–267, 2003.
2. L. Bull and J. Holland. Evolutionary computing in multi-agent environments: Eusociality. In *Proceedings of the Second Annual Conference on Genetic Programming*, pages 347–352, San Francisco, USA., 1997. IEEE Press.
3. J. Elman. Finding structure in time. *Cognitive Science*, 14(1):179–211, 1990.
4. B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, 1986.
5. F. Gomez. *Robust Non-Linear Control Through Neuroevolution. PhD thesis.* Department of Computer Sciences, The University of Texas, Austin, Texas, 2003.
6. D. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22(1):11–32, 1996.
7. G. Nitschke. Designing emergent cooperation: a pursuit-evasion game case study. *Artificial Life and Robotics*, 9(4):222–233, 2005.
8. S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 1(5):75–98, 1997.
9. M. Potter and K. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
10. K. Stanley, B. Bryant, and R. Miikkulainen. Real-time neuro-evolution in the nero video game. *IEEE Transactions Evolutionary Computation*, 9(6):653–668, 2005.