Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)

Evolutionary Design of Specialization

A.E. Eiben, G.S. Nitschke, M.C. Schut
Department of Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: gusz@cs.vu.nl, nitschke@cs.vu.nl, schut@cs.vu.nl

***Abstract*— In this research, a neuro-evolution method called Collective Neuro-Evolution (CONE), is introduced for the design of neural controllers (agents) operating in collective behavior task domains. The efficacy of the CONE method for facilitating emergent behavioral specialization for the benefit of increasing task performance is tested in a pursuit-evasion and collective gathering task. For a comparative study, a conventional neuro-evolution method was applied to the same tasks. In both tasks, the CONE method derived behavioral specialization in groups of agents resulting in higher task performances, where as the conventional neuro-evolution method was unable to derive specialization resulting in comparatively lower task performances.**

I. Introduction

Research in simulated and physical collective behavior systems, has often attempted to replicate the success of certain biological social systems [1] at decomposing the labor of a group into composite specialized and complementary roles so as to accomplish global goals that could not otherwise be accomplished by individuals, and to increase global task performance. The mechanisms motivating emergent specialization have been studied in biological [1], artificial life [2], and multi-robot [3] systems. However, collective behavior design methods for harnessing and utilizing emergent specialization for the benefit of problem solving and increasing task performance in such systems is currently lacking.

This paper describes a comparative study testing two neuro-evolution methods, for the purpose of controller design, where the aim was to demonstrate the benefit of emergent specialization in different collective behavior tasks[1].

A neuro-evolution method, termed herein as CONE: *Collective Neuro-Evolution* is introduced, and compared with a conventional neuro-evolution method in two separate collective behavior task domains. First, a pursuit-evasion task using small groups of simulated *Khepera* robots, where the task was for a group of predator robots to immobilize a prey robot. Second, a collective gathering task using a simulated swarm of *Unmanned Autonomous Vehicles* (UAV's), where the task was to discover as many features of interest in an environment unknown to the UAV's, given limited energy and life spans.

[1]The terms used herein are defined as follows. *Task*: what has to be done, *activity*: what is being done, *role*: the task assigned to an individual within a set of responsibilities given to a group of individuals, *caste*: a group of individuals specialized in the same role [3].

A. Neuro-Evolution for Collective Behavior Design

Neuro-evolution (NE) is an approach that combines techniques native to both neural networks and evolutionary computation research, for the purpose of evolving weights of neural controllers [4]. Recently NE has been successfully applied to controller design in a range of collective behavior task domains that have included multi-agent computer games [5], RoboCup soccer [6], and multi-robot controller design [2]. NE is most appropriately applied to complex problems that are neither effectively addressed via pure evolutionary computation methods [7] or neural processing approaches [8].

NE methods that are pertinent to this research include, SANE: *Symbiotic, Adaptive Neuro-Evolution* [9], and ESP: *Enforced Sub-Populations* [10], which has achieved particular success in collective behavior task applications.

SANE differed from other NE systems in that it evolved a population of neurons instead of complete networks. These neurons were combined to form hidden layers of feed-forward networks that were then evaluated in a given task domain. SANE has been effectively applied to single agent control tasks, such as playing Othello, robot arm control, and classical pole-balancing [10], and has proved faster and more efficient than reinforcement learning methods such as *Adaptive Heuristic Critic*, and *Q-Learning*, as well as conventional NE [9].

ESP is an extension of the SANE method, that similarly evolves neurons instead of complete networks. However, ESP creates one sub-population of neurons for each hidden layer neuron in a fully connected feed-forward network, where neurons could only be recombined with other neurons in the same sub-population. Dissimilar to SANE, ESP has been effectively applied to collective behavior tasks such as multi-agent computer games [5], Keep-away RoboCup soccer [6], and pursuit-evasion games [11]. In such tasks, the correct input-output mappings for controllers is not known *a priori*.

B. Research Goals, Hypotheses and Specialization

1) ***Research Goal****:* To conduct a comparative study in order to evaluate the CONE versus a conventional NE method for deriving (simulated) robot controllers in collective behavior tasks. Evaluation criteria included task performance and the level of specialization in a group of robots. One contribution of this research was the provision of a NE method, which facilitated emergent specialization so as to increase the performance of a group of robots in collective behavior tasks.

Previous research has elucidated that there exist particular types of task environments where behavioral specialization increases task performance. Specialization was been highlighted as being beneficial in pursuit-evasion [12] and collective gathering [13] task domains. Given this, our research hypothesis was formulated as follows.

*2) **Research Hypothesis**:* The CONE method was appropriate for deriving specialized agents that would yield a high task performance, in collective behavior task domains.

In order to test this hypothesis in both task domains, the task performance and emergent specialization observed was compared to that of a conventional neuro-evolution method.

*3) **Specialization**:* Specialization was defined at the *individual agent* level. An agent was considered to be specialized if a given behavioral role was assumed for the majority (*more than 50%*) of the agents lifetime.

In the case of the pursuit-evasion experiments, a behavioral role was defined according to observed emergent behavior. Emergent behavior was correlated with average sensor and motor activation values being within a particular range, for a given observed behavior. In the case the collective gathering experiments, a behavioral role was defined according to which one, out of a set of possible actions, an agent executed for the majority of its lifetime.

II. CONE: Collective Neuro-Evolution Method

A. Contributions of CONE

The CONE method was an extension of both the SANE [9] and ESP [10] methods. A key difference between the CONE and other NE methods [9], [10], was that it created n separate sub-populations of genotypes[2] (neurons) for n neural controllers (phenotypes) operating in the task environment.

One advantage of the CONE method was that it expedited artificial evolution, given that the genotype population was organized into sub-populations. Hence, *specialized controllers* did not have to emerge out of a single population of neurons, and progressive specialization of controllers was not hindered by recombination of controllers with complementary specializations.

A second advantage was that it provided more genotype diversity (comparative to conventional NE methods) and encouraged emergent controller specialization given that evolution occurred within separate sub-populations of genotypes. That is, it has been highlighted that organizing the genotype population into separate niches (sub-populations), either dynamically [14], or *a priori* [15] facilitates specialization, and protects innovation (emergent behaviors) within the specialized niches of the genotype space.

A third advantage of the CONE method was that it could be applied to execute *online* [14] as well as *offline* [2] in collective behavior tasks that require fast adaption of controllers, or yield poor performance if controllers are evolved offline, and then placed in a task environment.

[2]Throughout this article the terms *genotype* and *neuron*, as well as *controller* and *agent* are used interchangeably.

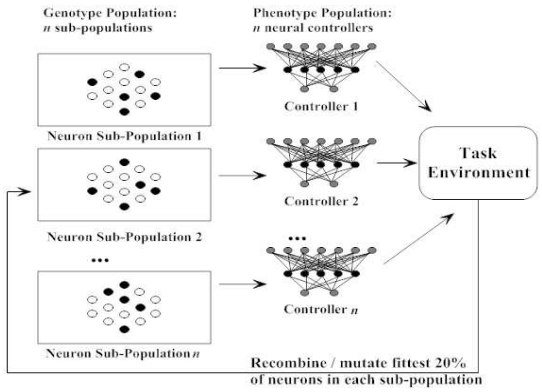

Fig. 1. CONE: Collective Neuro-Evolution. See section II for details.

B. Genotypes

For both the CONE (figure 1) and conventional NE (figure 2) methods, the populations of genotypes were encoded as a string of floating point values (table I), which represented neural network weights connecting all sensory input neurons and all motor output neurons to a given hidden layer neuron.

C. CONE Process

As illustrated in figure 1, after each of the n sub-populations, were randomly initialized with m genotypes the process of the CONE method was as executed follows.

1) n agents (neural controllers) were constructed via selecting p genotypes (neurons) from each sub-population of genotypes. These p neurons then became the hidden layer of each of the n controllers, which were subsequently placed in the task environment. The group of controllers was thus heterogeneous, given that each was constructed via selecting a set of p hidden layer neurons from each of the n sub-populations. Evolutionary operators were not applied between the n sub-populations.
2) The n controllers were tested together in the task environment for a *lifetime* of q epochs, where an epoch was a test scenario lasting for w iterations of simulation time. Each epoch tested different task dependent agent and environment conditions, such as agent starting positions and locations of resources in the environment. For each of the q epochs (where $q \geq m$ genotypes in a sub-population), each genotype in a given sub-population was selected and tested in combination with p-1 other neurons (thus forming a controller) randomly selected from the same sub-population.
3) Thus p neurons from each of the n sub-populations would concurrently be evaluated in the task environments and assigned a fitness. Testing of neurons within each sub-population would continue until all neurons had been tested at least once.
4) At the end of an agents lifetime (q epochs) a fitness value was assigned to each set of p neurons that participated in each of the controllers. The assigned fitness of each

set of p neurons was calculated as the average of fitness values attained over all epochs of an agents lifetime.

5) For each sub-population, recombination and mutation of the fittest 20% of genotypes then occurred, where the fittest 20% were arranged into pairs of genotypes, and each pair produced 5 child genotypes, so as to propagate the next generation of each sub-population.
6) A single genotype was randomly selected from the fittest 20% of the newly recombined genotypes within each of the n sub-populations. These n selected genotypes were then decoded into controllers, placed in the task environment, and executed as the next generation. This process was then repeated for r (table I) generations.

D. Recombination of genotypes: Crossover and Mutation

Each child genotype was produced using single point crossover [7], and *Burst* mutation with a *Cauchy* distribution [10]. As illustrated in table I 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]. Burst mutation was used to ensure that most weight changes were small whilst allowing for larger changes to some weights.

E. Constructing controllers from neurons

Given that the CONE method operated at the neuron (not the controller [16]) level, a controller was constructed via selecting p neurons from one sub-population of neurons.

The setting of specific neurons in specific hidden layer locations has the well investigated consequence that different neurons become specialized for different controller sub-tasks [17], over the course of a NE process. Hence, 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.

F. Fitness Calculation

At the end of each generation (section II-C) a fitness value was assigned to each of the n controllers, where each of the neurons participating in each controller was assigned an equal portion of the fitness value. These individual neuron fitness values were then assigned back to the sub-population corresponding to each of the controllers.

Although this fitness estimation method, known as *fitness sharing* [18] was convenient for deriving the contribution of each neuron to a controller, it was problematic in that it potentially prevented the selection of the best neurons across successive generations. However, this was offset by the advantage that there was no disparity between controller fitness and the fitness of individual neurons. The same fitness sharing method was also applied at the controller level, meaning that, when necessary (section IV-B), we assumed each controller contributed to group performance equally and should thus be rewarded with an equal fitness share.

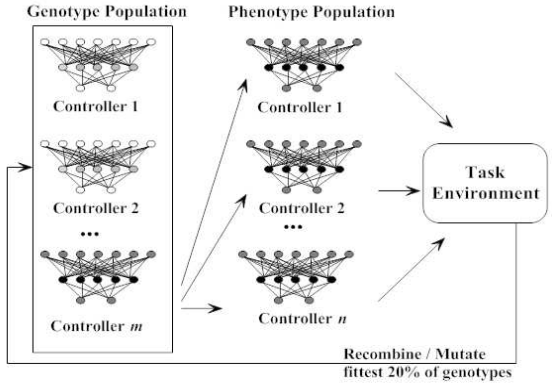

Fig. 2. Conventional neuro-evolution. See section III for details.

III. Conventional Neuro-Evolution

The conventional NE method was adapted from that used for previous evolutionary robotics experiments [19], and as illustrated in figure 2 used only a single population of genotypes. After, randomly initializing a population of m genotypes, the conventional NE process operated as follows.

1) Initially, n genotypes were randomly selected from the population of m genotypes, and decoded into n agents (neural controllers).
2) These n controllers were then placed in the task environment, to be tested and evaluated.
3) Each controller was tested for a *lifetime* of q epochs, where each epoch constituted a test scenario (section II-C) that lasted for w iterations of simulation time.
4) At the end of each controllers lifetime (q epochs), a fitness value was assigned to the genotype corresponding to each controller. The fitness assigned to a genotype was calculated as the average of all fitness values attained for all epochs of its lifetime.
5) Each of the m genotypes was systematically decoded into a neural controller and tested, together with n-1 other (randomly) selected genotypes, in the task environment. The testing of all m genotypes in the population constituted one generation of the NE process.
6) The fittest 20% of genotypes were then arranged into randomly selected pairs, and each pair recombined to produce 5 child genotypes each, so as to replace the current genotype population.
7) n genotypes were then randomly selected from the fittest 20% of the next generation of genotypes. Each selected genotype was decoded into its corresponding controller and placed in the task environment.
8) This process was repeated for the r generations that the conventional NE method was executed for (table I).

CONE and Conventional Neuro-Evolution Parameter Settings		
	Pursuit-Evasion	Collective Gathering
Runs per NE method	20	20
Generations	500	500
Epochs	50	50
Iterations / Epoch	1000	1000
Mutation probability	0.05	0.05
Mutation range	[-1.0, +1.0]	[-1.0, +1.0]
Weight range	[-10.0, +10.0]	[-10.0, +10.0]
Crossover	single point	single point
Hidden neurons	5	10
Phenotypes	[1, 6] Controllers	100 Controllers
Genotype length	18 (16 + 2) weights	18 (14 + 4) weights
Genotypes	100 / 600 (CONE)	100 / 10000 (CONE)

TABLE I

NEURO-EVOLUTION PARAMETER SETTINGS FOR THE PURSUIT-EVASION VERSUS THE COLLECTIVE GATHERING TASK DOMAINS.

IV. PURSUIT-EVASION TASK

The pursuit-evasion task was a game played by the simulated robots, where it was the collective task of pursuers (herein called: predators) to immobilize one evader (herein called: prey). A control experiment using a single predator and single prey, demonstrated that cooperation, between at least two predators, was needed to accomplish this task [12]. Collective behavior was only evolved for the predator team, and each prey was able to move 20 percent faster than the predators. The behavior of each prey was not evolved, but instead used a previously evolved obstacle avoidance behavior.

Experiments were conducted in simulation using an extended version of the EvoRobot Khepera simulator [20], where each predator and prey was embodied as a Khepera mobile robot [21]. Figure 3 presents the prey as being equipped with 8 infrared proximity sensors (SI_0..SI_7), as well as a light (L_0) on its top. This light could be detected by predator light sensors, and allowed each predator to distinguish fellow predators from a prey. Each predator had the same set of sensors and actuators: 8 infrared proximity (SI_0, $S1_2$, SI_4, SI_6, SI_8, SI_{10}, SI_{12}, SI_{14}) and 8 light sensors (SI_1, $S1_3$, SI_5, SI_7, SI_9, SI_{11}, SI_{13}, SI_{15}) positioned on its periphery.

The simulation environment corresponded to a 1000cm x 1000cm arena with no obstacles. When the predator and prey robots were placed in the environment, sensory input was received via the input units, and activation values were passed to the two motor units. The activation value of the two motor units (MO_0, MO_1) was used to move the robots, thus changing the sensory input for the next simulation cycle. This cycle was then repeated, continuing until the end of robots lifetime.

A. Neural Controllers for Pursuit-Evasion

Predator and prey behavior was controlled by feed-forward neural controllers, with a single hidden layer connecting sensory inputs and motor outputs. Since only predator controllers were evolved, only the predator controller is illustrated in figure 4. The neural controller of each predator robot consisted of 16 sensory neurons, which encoded the state of infrared and light sensors, and 2 motor neurons, which encoded the speed of the two wheels, where sensory inputs and motor outputs were fully connected to 5 hidden layer neurons.

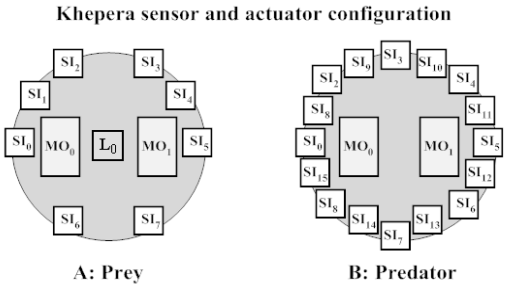

Fig. 3. Sensory-motor configuration of simulated Khepera robots. See section IV for explanation.

A predators controller was represented as a set of 5 genotypes (hidden layer neurons). Where a single genotype was encoded as a string of 18 connection weights. That is, 16 weights (IW_0..IW_{15}) connecting 16 sensory input neurons, and 2 weights (OW_0..OW_1) connecting 2 motor output neurons to a given hidden layer neuron. Hence a complete controller was encoded as 90 (5 x 18) connection weights, where weights connecting sensory input and motor output neurons to hidden layer neurons changed only according to the NE method applied (sections II and III). The NE parameters utilized for the comparative NE methods are presented in table I. The genotype to phenotype mapping scheme for the predators (prey controllers were not subject to a NE process) was a direct one-to-one mapping, where each connection weight corresponded to a floating-point number in the interval [-10, +10].

A prey controller consisted of 8 sensory inputs encoding infrared neuron states, and 2 motor neurons encoding the speed of two wheels. Sensory input and motor output neurons were fully connected to 5 hidden layer neurons.

For both predators and prey, the activation value of each output unit was used to update the speed of the corresponding wheel every 100 milliseconds of simulation time. To allow the prey to move 20% faster than the predators, the activation value of each of its output units was multiplied by 1.2, before setting the wheel speed. Further descriptions of predator and prey controllers, as well as the previously evolved prey obstacle avoidance behavior is given in related work [12].

B. Predator Fitness Rewards

The predator team was rewarded a fitness equal to the total time for which it was able to immobilize a prey. This fitness value was shared equally (section II-F) between all predator controllers in a team.

V. PURSUIT-EVASION GAME EXPERIMENTS

For both NE methods, 5 different group configurations of predators and prey were tested. These group configurations were named and defined as follows. *Group type 1*: 2 predators and 1 prey; *Group type 2*: 3 predators and 1 prey; *Group type 3*: 4 predators and 1 prey; *Group type 4*: 5 predators and 1 prey; *Group type 5*: 6 predators and 1 prey. The performance measure used was the average prey-capture time, where the average was taken over all predators lifetimes, and all experimental runs. The NE parameter settings specified in

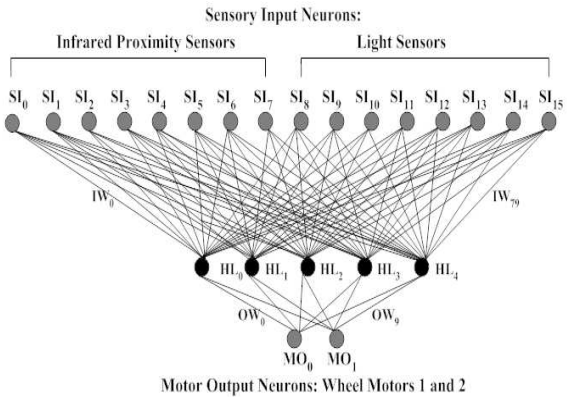

Fig. 4. Neural controller of predator robots. See section IV-A for details.

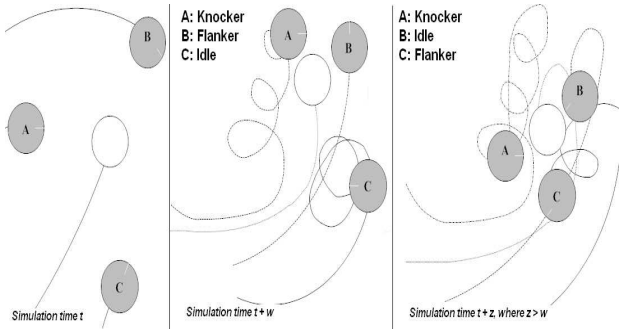

Fig. 5. Role switcher strategy. Predators B and C alternated between specialized roles of flanker and idle, so as to increase strategy effectiveness.

table I were selected given the success of related parameter settings in previous evolutionary robotics experiments [19].

A. Evolved Prey-Capture Behavior

1) Conventional NE Method: For all experimental runs of the conventional NE method, and all group types tested (section V), two cooperative prey capture strategies[3], termed: *entrapment* and *encirclement* consistently emerged.

In the encirclement strategy, at least three and at most four predators moved to circle the prey, each moving in the same direction in close proximity to the prey. The predators would gradually move closer to the prey, eventually immobilizing it. The strategy was only successful for immobilizing a prey for relatively short periods of time, given that the predators were not able to coordinate their movements for extended periods.

Similarly, the entrapment strategy utilized at least three and at most four predators, where all moved simultaneously towards a prey from different directions in order to immobilize it within a triangular or square formation. As with the encirclement strategy, all predators remained in close proximity to the prey, except that they would *knock* against it in order to prevent its escape. The entrapment strategy was also hindered by a lack of coordination between the predators, and the number of prey capture instances was less than the encirclement strategy. Although in an instance when a prey was cooperatively trapped, prey capture time was longer comparative to the encirclement strategy.

2) CONE Method: For all experimental runs, and all group types tested using the CONE method, only one emergent cooperative prey capture strategy (termed: *role switcher*) was consistently observed. Here, predators adopted and kept complementary behavioral roles throughout their lifetimes, where these behaviors were either switched on (active) or off (idle).

Figure 5 illustrates the role switcher strategy. After detecting the prey at simulation time t, predators A and B, moved to intercept the prey at simulation time $t + w$. Predator B switched off its flanking behavior to become idle, and predator C switched from an idle to a flanking behavior, whilst predator A retained its knocker behavior, in order to prevent evasion of the prey when it turned about to move in the opposite direction at simulation time $t + z$, where $z \geq w$. Thus, particular predators assumed particular roles (either a *flanker* or *knocker*), where these behaviors were switched on or off (idle), in response to prey movements. This adoption of roles, maintained throughout the lifetime of the predators increased the effectiveness of the role switcher strategy, and thus the performance of the CONE method. Also, the role switcher strategy was consistently effective at immobilizing a prey, for all group types tested, where as this was not the case for the entrapment and encirclement prey-capture strategies that emerged under the conventional NE method.

3) Comparisons: In order to determine the statistical significance of difference between the prey-capture time results of the conventional NE and CONE methods presented in figure 6, an an independent t-test [22] was applied to each data set. Both data sets were found to conform to normal distributions via applying the Kolmogorov-Smirnov test [22] (P = 0.99, 0.98 for the conventional and CONE methods respectively). For each t-test, 0.05 was selected as the threshold for statistical significance, and the null hypothesis was stated as the two data sets not significantly differing. The t-test yielded P = 0.0003, indicating rejection of the null hypothesis, and supporting the observation that the CONE method yielded a greater performance comparative to the conventional NE method.

This comparison, and the observation that the role switcher strategy (emergent under the CONE method) comprised complementary specialized behaviors, supported our research hypothesis (section I-B.2) for the pursuit-evasion task.

4) Measuring Specialization: Importantly, the role switcher strategy exhibited complementary specialized behaviors at the controller level, where as the entrapment and encirclement prey-capture strategies did not. Using methods from related work [19] 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). These activation instances were then named according to the corresponding observed behavior (flanker, blocker and idle), and their activation times summed. If the

[3]All emergent prey capture strategies observed for both the CONE and conventional NE methods are detailed in [12].

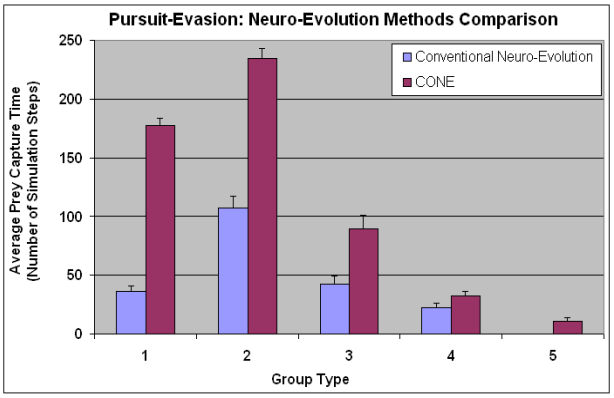

Fig. 6. Performance results of the conventional NE and CONE methods in the pursuit-evasion task. Results are displayed for 5 different group types.

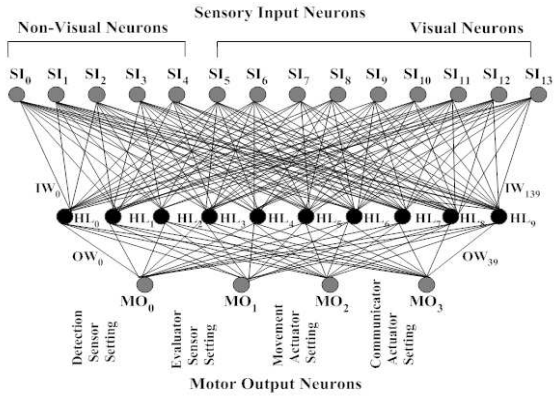

Fig. 7. Neural controller of explorer agents. See section VI-C for explanation.

portion summed $\geq 50\%$ of the predators lifetime the observed behavior was labeled as specialized (section I-B.3). Flanker and blockers were classified and labeled as such via simply observing predator behavior during the simulation.

Hence, the role switcher strategy (emergent under the CONE method) was comprised of predators with complementary and specialized behaviors, and this strategy resulted in a significantly higher prey-capture time comparative to the encirclement and entrapment strategies (emergent under the conventional NE method). Given this, it was theorized that specialization was beneficial for task performance in that it facilitated successful coordination of predator behaviors, reduced physical interference, and thus increased prey-capture time. This was not the case for the encirclement and entrapment strategies where a lack of coordinated behavior and resultant physical interference between predators (as they collectively approached a prey) was observed.

VI. Collective Gathering Task

Inspired by UAV survey and exploration missions of unknown environments [23], a group of 101 simulated UAV's (100 *explorer* agents and 1 *lander* agent) were given the task of maximizing the number of features of interest (herein termed: *red rocks*) discovered within a *survey area* of an environment given limited sensor and actuator capabilities, battery power and mission time. Red rock locations and distributions were initially unknown to the agents. The lander had no active role in the discovery of red rocks. Its role was to act as a base station that received red rock data (locations and value of features of interest) communicated to it, and to recharge explorer agents that successfully accomplished their task.

A. Environment

The simulation environment [4] was represented as a discrete three dimensional environment of 200 *x* 200 *x* 200 voxels. At the start of each simulation, each explorer was initialized in a random voxel (x, y, z, where, $0 \leq$ x, y, z < 200). A maximum of 4 explorer agents could occupy any given voxel. Within the environment, red rocks were distributed according to a two dimensional *Gaussian mixture model* [24]. The mixture model was specified with 4 centroids, set in static locations, where the radius of each determined the spatial distribution of red rock around each. For all collective gathering experiments, five radii were tested, such that red rock distributions generated ranged from approximately uniform (*low degree of structure*) through to a clustered (*high degree of structure*). For a given simulation, 40000 red rocks were distributed over the ground level of the environment. Where, a red rock could be placed at each possible x, y, z, where $0 \leq$ x, y < 200, and z=0.

[4]Demo's, source code and documentation of the simulation environment is available at: www.cs.vu.nl/nitschke/Marsscape/.

B. Explorer Agents: Sensors and Actuators

Agent morphology was defined in terms of 1 *detection sensor*, 1 *evaluation sensor*, 1 *movement actuator*, and 1 *communication actuator*. This selection of sensors and actuators was based upon design proposals for rotorcraft that are to operate as autonomous aerial explorers [25]. Rotorcraft are considered advantageous given their vertical lift capabilities, allowing them to *detect* red rocks in flight using a directional visual sensor, perform some preliminary categorization of red rocks, *move* in order to *evaluate* selected red rocks using a physical contact sensor, and then *communicate* red rock data. Rotorcraft are also able to land to recharge at a base station.

Selection of one of the four possible actions (detect, evaluate, move, communicate) was determined by the motor output node yielding the highest output value (figure 7).

Parameter settings specific to explorer agents, such as battery energy, sensor and actuator costs, communication range and speed of movement are specified in related work [13].

C. Explorer Agents: Neural Controllers

Figure 7 illustrates an explorer agent controller as consisting of 14 sensory input nodes ($SI_0..SI_{13}$) and 4 motor output nodes ($MO_0..MO_3$) connected to 10 hidden layer nodes ($HL_0..HL_9$).

1) Sensory inputs: The 5 non-visual input nodes ($SI_0..SI_4$) took as input the 4 motor output (MO_0-MO_3) values, and the red rock evaluation value from the previous simulation iteration, respectively. These previous motor outputs were teaching inputs [19] which influenced the next motor outputs. The 9

non-visual input nodes (SI_5..SI_{13}) represented the number of red rocks at the agents current position, and at each of the surrounding 8 voxels (on the plane $z = 0$) in the environment. That is, to the *north*, *south*, *east*, *west*, *northwest*, *northeast*, *southwest*, and *southeast*. All values taken by sensory input nodes were normalized.

2) Motor outputs: MO_0-MO_3 corresponded to the 4 actions an agent could select. MO_0 and MO_1 activated the *detection* and *evaluation* sensors, respectively. MO_2 and MO_3 activated the *movement* and *communications* actuators, respectively. The motor output node that generated the highest value was the action selected. All motor output values were normalized.

3) Genotype representation: A controller was represented as a set of 10 genotypes (hidden layer neurons). A single genotype was encoded as a string of 18 connection weights. That is, 14 weights (IW_0..IW_{13}) connecting 14 sensory inputs, and 4 weights (OW_0..OW_3) connecting 4 motor outputs to a given hidden layer neuron. The parameters utilized for the comparative NE methods are presented in table I.

D. Features of Interest: Red Rock

Heuristic behavior (based on controller heuristics described by [13]), complemented explorer agent neural controllers (which learnt *how* best to discover red rocks) so as to ensure particular behavior when red rocks *were* discovered.

1) Discovered red rock heuristics: When an agent discovered a red rock using its detection sensor; red rock location was moved to using its movement actuator; red rock data was then ascertained using its evaluation sensor; red rock data was communicated from the agent to the lander using its communication actuator; red rock value (0 or 1) was communicated back to the agent from the lander; and the evaluated red rock was marked: *evaluated*, so it would not be evaluated again.

2) Red rock value: Agents that had evaluated red rocks would be marked as eligible to receive a recharge reward (reset = 0 after an agent returned to the lander to recharge) and an immediate fitness reward equal to the previous red rock value gathered, where rewards were cumulative. An agent returned to the lander for recharge when its energy level was $\leq 25\%$ of its maximum. Hence, red rock value served two purposes. First, it was equated with controller fitness and thus used by a NE method in selection and propagation of the best performing controllers. Second, it allowed high performance agents to have their lifetimes extended.

E. Individual and Group Fitness Rewards

When an agent *discovered* a red rock, all other agents within its communication range would receive an equal share of a fitness reward (*red rock value*). This reward scheme assumed that red rock data of newly discovered red rocks would always be communicated to other agents (within communication range), thereby increasing group performance. Hence, individual agent fitness was equal to the number of red rocks it had discovered thus far (in its lifetime), where as, group fitness was equal to the total number of communications (concerning red rock with value = 1) received by the lander.

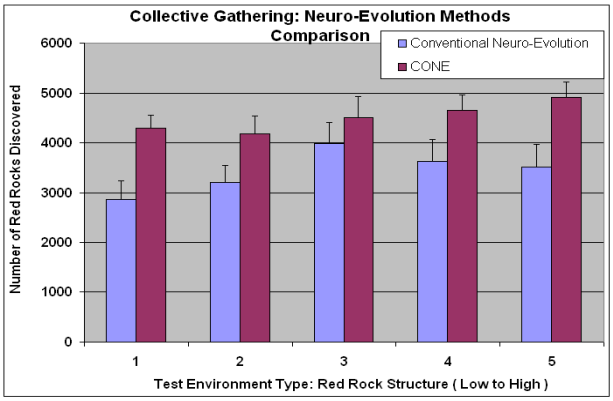

Fig. 8. Performance results (for 5 different test environments) of the conventional NE and CONE methods in the collective gathering task.

VII. Collective Gathering Experiments

For both NE methods, 5 different environment types were tested, where each used a specific red rock distribution (section VI-A). The performance measure was the average number of red rocks discovered, where the average was taken over the lifetime of all agents, and all experimental runs (table I).

A. Evolved Group Compositions

1) CONE Method: Table II presents the highest performing group composition (average of 4907 red rocks discovered) derived by the CONE method (occurring in *environment type 3*). The group was comprised of a majority of *detector* agents (0.52), minor portions of *evaluator* (0.25) and *communicator* (0.22) agents, and agents of no specialization (0.01).

2) Conventional NE Method: Likewise, table II presents the highest performing group composition (average of 3988 red rocks discovered) derived by the conventional NE method (occurring in *environment type 5*) as consisting of three approximately equal portions of agent types, with none having a majority. That is, *detector* (0.37), *evaluator* (0.36), *communicator* (0.15), and agents of no specialization (0.12).

3) Comparisons: In order to draw conclusions from this comparative study, a set of statistical tests were performed in order to gauge task performance differences between respective NE method results. First, it was determined that results of the conventional and CONE methods conformed to normal distributions via applying the Kolmogorov-Smirnov test [22] (P = 1.0, 0.98 for the conventional and CONE methods respectively). Second, to determine the statistical significance of difference between data presented in figure 8 an independent t-test [22] was applied. We selected 0.05 as the threshold for statistical significance, and stated the null hypothesis as the two data sets not significantly differing. The t-test yielded P = 0.0007, indicating rejection of the null hypothesis.

4) Measuring Specialization: In this case we measured specialization at the group level (as well as the agent level). Specialized agents were labeled as: detectors, evaluators or communicators according to which sensor or actuator (excluding movement) was used for $\geq 50\%$ of the agents lifetime.

Group composition derived with conventional NE (A)					
Detectors	Evaluators	Communicators	No Special-ization	DoS	Average RRVG
0.37	0.36	0.15	0.12	3	3988
Group composition derived with the CONE method (B)					
Detectors	Evaluators	Communicators	No Special-ization	DoS	Average RRVG
0.52	0.25	0.22	0.01	5	4907

TABLE II

COMPOSITIONS OF BEST PERFORMING GROUPS USING CONVENTIONAL NE (A) CONE (B) METHODS. RRVG: RED ROCK VALUE GATHERED. DOS: DEGREE OF STRUCTURE.

A caste was likewise labeled if $\geq$ 50% of agents in the group were detectors, evaluators, or communicators. Table II presents the best performing agent group using the CONE method as converging to a detector caste. Comparison of best performing groups for both NE methods, and the specialization (section I-B.3) of the highest performing group to a detector caste, supports our research hypothesis (section I-B.2) for the collective gathering task. The CONE method yielded a detection caste and the conventional NE method yielded no castes (theorized to be a result of using a single genotype population). Hence, specialization (facilitated by the CONE method) was theorized to be beneficial for task performance.

VIII. CONCLUSION

This paper introduced the CONE: *Collective Neuro-Evolution* method, and compared its task performance with a conventional NE method for controller design given two different collective behavior tasks. The CONE method was designed to facilitate emergent behavioral specialization within a group of controllers (agents), and operated under the supposition that if specialization was advantageous, then behavioral specialization would emerge for the benefit of increasing collective behavior task performance. To test this research supposition, the CONE and a conventional NE method were applied to pursuit-evasion and collective gathering tasks.

When the CONE method was applied to the pursuit-evasion task, a group of predator robots adopted specialized behavioral roles which enabled the formation of collective prey-capture strategies capable of effectively immobilizing a prey robot. When the CONE method was applied to the collective gathering task, a group of Unmanned Autonomous Vehicles (UAV's) adopted different frequencies of use of their sensors and actuators so as to form complementary specialized behaviors. For both tasks, in addition to these emergent specialized behaviors a high level of task performance was observed.

The value of behavioral specialization emergent under the CONE method was demonstrated by the result that, comparatively, the conventional NE method yielded a lower performance in both task domains. The higher task performance of controllers derived using the CONE method, was theorized to be consequent of emergent behavioral specialization that enabled the formation of effective collective behaviors yielding the benefit of increased task performance. However, this is subject to current research.

REFERENCES

[1] J. Gautrais, G. Theraulaz, J. Deneubourg, and C. Anderson, "Emergent polyethism as a consequence of increased colony size in insect societies," *Journal of Theoretical Biology*, vol. 215, no. 1, pp. 363–373, 2002.

[2] G. Baldassarre, S. Nolfi, and D. Parisi, "Evolving mobile robots able to display collective behavior." *Artificial Life*, vol. 9, no. 1, pp. 255–267, 2003.

[3] M. Kreiger and J. Billeter, "The call of duty: Self-organized task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, pp. 65–84, 2000.

[4] D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," *Neural Networks*, vol. 13, no. 1, pp. 431–443, 2000.

[5] B. Bryant and R. Miikkulainen, "Neuro-evolution for adaptive teams," in *Proceedings of the 2003 Congress on Evolutionary Computation*. Canberra, Australia: IEEE Press, 2003, pp. 2194–2201.

[6] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving keep-away soccer players through task decomposition." in *Proceeding of the Genetic and Evolutionary Computation Conference*. Chicago: AAAI Press, 2003, pp. 356–368.

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

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

[9] D. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution." *Machine Learning*, vol. 22, no. 1, pp. 11–32, 1996.

[10] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, no. 1, pp. 317–342, 1997.

[11] C. Yong and R. Miikkulainen, *Cooperative coevolution of multi-agent systems. Technical Report AI01-287.* Austin, Texas: Department of Computer Sciences, The University of Texas, 2001.

[12] G. Nitschke, "Designing emergent cooperation: a pursuit-evasion game case study," *Artificial Life and Robotics*, vol. 9, no. 4, pp. 222–233, 2005.

[13] A. Eiben, G. Nitschke, and M. Schut, "Collective specialization of aerial explorer controllers," in *Proceedings of the Second International Workshop on Swarm Robotics*. Rome, Italy: Springer, 2006, p. In Press.

[14] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuro-evolution in the nero video game," *IEEE Transactions Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, 2005.

[15] M. Potter and K. DeJong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[16] S. Nolfi and D. Floreano, "Learning and evolution." *Autonomous Robots*, vol. 7, no. 1, pp. 89–113, 1999.

[17] K. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification." *Journal of Artificial Intelligence Research*, vol. 21, no. 1, pp. 63–100, 2004.

[18] L. Bull and J. Holland, "Evolutionary computing in multi-agent environments: Eusociality," in *Proceedings of the Second Annual Conference on Genetic Programming*. San Francisco, USA.: IEEE Press, 1997, pp. 347–352.

[19] S. Nolfi and D. Parisi, "Learning to adapt to changing environments in evolving neural networks," *Adaptive Behavior*, vol. 1, no. 5, pp. 75–98, 1997.

[20] S. Nolfi, *Evorobot 1.1 User Manual. Technical Report.* Rome, Italy: Institute of Cognitive Sciences, National Research Council, 2000.

[21] F. Mondada, E. Franzi, and P. Ienne, "Mobile robot miniaturization: A tool for investigation in control algorithms," in *Proceedings of Third International Symposium on Experimental Robotics*. Kyoto, Japan.: IEEE Press, 1993, pp. 501–513.

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

[23] S. Thakoor, "Bio-inspired engineering of exploration systems," *Journal of Space Mission Architecture*, vol. 2, no. 1, pp. 49–79, 2000.

[24] P. Paalanen, J. Kamarainen, J. Ilonen, and H. Klviinen, "Feature representation and discrimination based on gaussian mixture model probability densities - practices and algorithms," *Pattern Recognition*, vol. 39, no. 7, pp. 1346–1358, 2006.

[25] L. Young, E. Aiken, G. Briggs, V. Gulick, and R. Mancinelli, "Rotorcraft as mars scouts." in *Proceeding of the IEEE Aerospace Conference*. Big Sky, USA: IEEE Press, 2002, pp. 4–12.