Emergent Specialization in the Extended Multi-Rover Problem

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

***Abstract*— This paper introduces the *Collective Neuro Evolution* (CONE) method, and compares its efficacy for designing specialization, with a conventional Neuro-Evolution (NE) method. Specialization was defined at both the individual agent, and at the agent group level. The CONE method was tested comparatively with the conventional NE method in an extension of the multi-rover task domain, where specialization exhibited at both the individual and group level is known to benefit task performance. In the multi-rover domain, the task was for many agents (rovers) to maximize the detection and evaluation of points of interest in a simulated environment, and to communicate gathered information to a base station. The goal of the rover group was to maximize a global evaluation function that measured performance (fitness) of the group. Results indicate that the CONE method was appropriate for facilitating specialization at both the individual and agent group levels, where as, the conventional NE method succeeded only in facilitating individual specialization. As a consequence of emergent specialization derived at both the individual and group levels, rover groups evolved by the CONE method were able to achieve a significantly higher task performance, comparative to groups evolved by the conventional NE method.**

I. Introduction

Research in simulated and physical collective behavior systems, has often attempted to replicate the success of certain biological social systems [7] 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 [22] artificial life [3], and multi-robot [2] 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.

Neuro-Evolution (NE) is a research field that combines techniques native to neural [10] and evolutionary computation [5] research. Recently NE has been applied for the purpose of finding solutions to both discrete and continuous multi-agent systems control tasks [8]. Such applications include agent controller design in multi-agent computer games [3], RoboCup soccer [23], and physical multi-robot systems [2]. NE has been highlighted as being most appropriately applied to multi-agent tasks that are neither effectively addressed via pure evolutionary or neural computation methods [8]. NE methods have been successfully applied to the rover task domain [1], [21]. However, extending this domain to include the notion of using NE to facilitate emergent specialization, in order to increase task performance, has not yet been investigated.

This paper describes the application of CONE and conventional NE to the rover task domain. This task requires solutions for controlling groups of simulated planetary exploration rovers that seek to maximize the number of points of interest, herein termed *red rocks*, discovered in an unexplored environment. An extension to the rover domain was implemented, in which rovers operated in a discrete simulation environment, and used complementary sensors and actuators in order to maximize a global fitness function.

The research of [4] elucidated that behavioral specialization (at either the individual or group rover level) is beneficial for task performance in this extension of the rover domain. Furthermore, it was demonstrated that heuristic based systematic search methods are not appropriate for achieving optimal performance in the extended rover task, and that such non-adaptive heuristic methods are significantly out-performed by NE multi-agent control methods.

For both NE methods, each rover maintained and evolved its own population of genotypes. The CONE method evolved populations of neurons, from which complete neural network controllers were constructed, where as, the conventional NE method evolved populations of complete controllers. We also introduced a heuristic performance benchmark, against which the task performance results of rover groups derived with the NE methods could be compared.

A. 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 (rover) controllers in the rover domain. One contribution of this research was the provision of the CONE method, which facilitated emergent specialization, at both the *individual* agent and *group* level, so as to increase group fitness in collective behavior tasks.

2) ***Task Performance Evaluation****:* The evaluation criterion for individual rover task performance was the number of red rocks detected, moved to, and evaluated, where information pertaining to red rocks was communicated to a lander (base station). The evaluation criterion for the rover group was the number of communications received by the lander. In order to avoid the problem where each rover evolves neural networks that maximize their own fitness function, yet the system as a whole achieves low values of the global fitness function, we used a difference evaluation function [1] to evaluate group fitness.

*3) **Research Hypothesis**:* In the rover task domain, the CONE method is appropriate for deriving specialization at both the individual agent (rover controller), and at the group (composition of specialized rover controllers), where such specialization results in a high group fitness (performance).

In order to test this hypothesis, the task performance and emergent specialization observed using the CONE method for controller derivation was compared to that of a conventional NE method, as well as a non-adaptive heuristic controller method.

*4) **Specialization**:* Specialization was defined at both the *individual* and the rover *group* level. An individual rover was defined as being specialized if a given behavioral role (a given action executed) was assumed for the majority ($\geq$ *50%*) of the rovers lifetime. Likewise, a rover group was defined as being specialized if rovers with a given behavioral role (that is: a individual specialization) constituted the majority ($\geq$ *50%*) of the rovers in the group.

Emergent specialization, facilitated by the NE methods was measured at both the individual and rover group level. At any simulation time step, a rover could execute one action, which equated to using one of two sensor types (*detection* or *evaluation*), or one of two actuator types (*movement* or *communication*). If a rover spent the majority of its lifetime executing a single action, it would be labeled as an *evaluator*, *detector*, *communicator*, or *mover* (table IV). Likewise, rover groups were defined as *evaluator*, *detector*, *communicator*, or *mover* groups if a majority of the group consisted of rovers with a given individual specialization.

For defining a rover domain performance benchmark, individual rover specialization was specified *a priori*, and a parameter calibration method (section V-B) used to evolve a non-specialized rover group with a high task performance.

II. Collective Neuro-Evolution

The CONE method is an extension of both the SANE [11] and ESP [8] methods. A key difference between the CONE and other NE methods [11], [8], is that it creates n separate genotype sub-populations (neurons) for n neural controllers (phenotypes) operating in the task environment.

One advantage of the CONE method 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 based NE methods) and encourages emergent controller specialization given that evolution occurs within separate sub-populations of genotypes. That is, it has been highlighted that organizing the genotype population into separate niches (sub-populations), either dynamically [18], or *a priori* [17] facilitates specialization, and protects innovation (emergent behaviors) within the specialized niches of the genotype space.

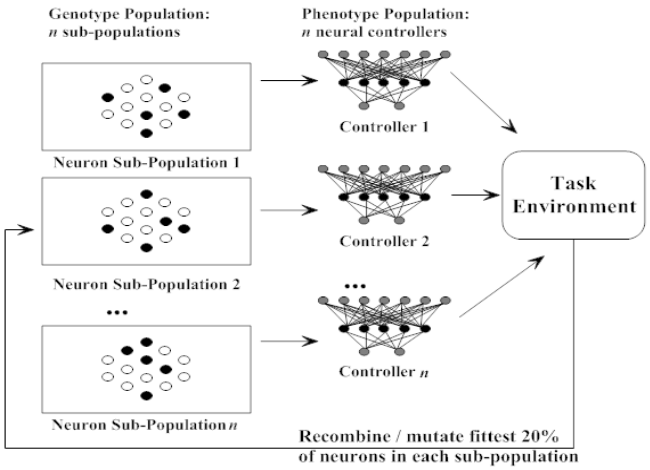

Fig. 1. Collective Neuro-Evolution (CONE) Method (Section II).

A. Genotypes

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

1) Recombination of genotypes: In both the CONE and conventional NE methods, each child genotype was produced using single point crossover [5], and *Burst* mutation with a *Cauchy* distribution [8]. 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.

B. 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 rovers (neural controllers) were constructed via selecting p genotypes (neurons) from each neuron sub-population. 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 rover and environment conditions, such as rover 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 be concurrently evaluated in the task environment 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 a rovers 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 rovers 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 the genotypes in each sub-population and propagate the next generation of each sub-population.
7) p neurons were randomly selected from the fittest 20% of the newly recombined neurons within each of the n sub-populations. These n sets of p neurons were then decoded into n controllers, placed in the task environment, and executed as the next generation. This process was then repeated for r generations.

C. Constructing controllers from neurons

Given that the CONE method operated at the neuron (not the controller [14]) 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 [19], 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.

III. Conventional Neuro-Evolution

The conventional neuro-evolution method was based on the research of [1], [21], where each of the n rovers in the group maintained and evolved its own population of complete neural network controllers. This approach differs from the CONE method (section II-B), which evolved n populations of neurons, from which n neural controllers were constructed. After, randomly initializing (with random weights) n populations, each containing m genotypes (controllers), the conventional NE process operated as follows.

1) Each rover selects the fittest genotype from its population 90% of the time and a random genotype from its population 10% of the time.
2) These fittest n genotypes were decoded into controllers, and placed in the task environment to be tested and evaluated.
3) Each of the m genotypes (representing the population of controllers for a given rover) was systematically decoded into a neural controller and tested, together with n-1 other (randomly) selected genotypes (representing the controllers of other rovers), in the task environment.
4) Each controller was tested for a *lifetime* of q epochs, where each epoch constituted a test scenario that lasted for w iterations of simulation time.
5) 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.
6) The testing and evaluation of the m genotypes in each rovers genotype population constituted one generation of the conventional NE process.
7) 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.
8) This process was repeated for the r generations that the conventional NE method was executed for.

TABLE I

Multi-rover NE and simulation parameter settings.

Neuro-Evolution and Simulation Parameter Settings	
Environment size	200 x 200
Points of interest (Red rocks)	40000
Rover battery	1000 units
Cost per action	1 unit
Runs per experiment	20
Generations	500
Epochs	50
Iterations per Epoch	1000
Mutation probability	0.05
Mutation range	[-1.0, +1.0]
Weight range	[-10.0, +10.0]
Crossover	single point
Hidden neurons	10
Phenotypes	100 Controllers
Genotype length	25 (21 + 4) weights
Genotypes	10000

IV. Discrete Rover Problem

A. Extended Rover Task

We propose an extended version of the multi-rover task [1], [21], in which the complexity of the rovers and environment is increased. As in the original rover problem, each rover attempts to maximize its own *private evaluation function* (subsection IV-E.2) in order to maximize a *global evaluation function* (subsection IV-E.1). The private evaluation function calculates the number of red rocks detected in a rovers lifetime. The global evaluation function calculates the number of red rocks detected by a rover group over the course of the groups lifetime.

However, each rover has the possibility of selecting between multiple actions (as opposed to only a move action in the original rover problem) at a given simulation iteration (section IV-C). Previous case studies conducted in the extended multi-rover problem [4], have highlighted that behavioral specialization increases task performance. Furthermore, these case studies demonstrated that applying a systematic search method to the rover group is not appropriate for attaining a high degree of task performance in the extended multi-rover problem. That is, the constraints of limited rover energy and sensor capabilities[1] mandated the use of an adaptive search method.

B. Environment

Adapted from the research of [1], this task simulation was characterized by a set of rovers operating on a discrete two dimensional plane (200 x 200 quadrants) which served as a *survey area* for a simulated search and find mission. At the start of a simulation 100 rovers and 1 lander were initialized in a random quadrant (x, y, where: $0 \leq$ x, y < 200). A maximum of four rovers could occupy any given quadrant. On the plane, red rocks were distributed according to a two dimensional *Gaussian mixture model* [16]. 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. 10 radii (ρ = 50, 45, 40, 35, 30, 25, 20, 15, 10, 5) were tested, such that red rock distributions generated ranged from approximately uniform (*low degree of structure*) through to a clustered (*high degree of structure*). 40000 red rocks were distributed such that a red rock could be placed at each possible x, y, where $0 \leq$ x, y < 200. Red rock locations and distributions were initially unknown to the rovers. The lander had no active role in the discovery of red rocks. Its role was to act as a base station that kept a record of the number of successful red rock evaluations (the global performance measure of the rover group).

C. Rover Sensors and Actuators

Rover morphology was defined by two sensor types (detection and evaluation) and two actuator types (movement and communication). This selection of sensors and actuators was based upon design proposals for autonomous rovers [20] that are capable of *detecting* and performing some preliminary categorization of red rocks using directional visual sensors (detection sensor), *moving* (movement actuator) in order to *evaluate* selected red rocks using a physical contact sensor (evaluation sensor), and then *communicating* pertinent red rock data to a base station (communications actuator).

1) Movement Actuator: At any given simulation time step a rover could activate its movement actuator (wheels) to move to an adjacent quadrant. To calculate distances between two quadrants (*p, q*) with discrete positions of (*u, v*) and (*w, z*) respectively, on the two dimensional plane, the following metric was used.

$$\delta(p,q) = min(|(u,v) - (w,z)|) \tag{1}$$

2) Communications Actuator: At any given simulation time step a rover could activate its communications actuator (radio) in order to communicate the value of evaluated red rocks (cumulative since the last communication) to the lander. Communication is broadcast with a fixed radius, so the location of the lander does not need to be known by each rover. If no red rocks had been evaluated since the last communication, nothing would be communicated.

3) Detection Sensors: At each simulation iteration, the rovers sensed the environment through 16 detection (visual) sensors that accepted continuous inputs (figure 2). From a rovers perspective, the environment was divided up into 8 quadrants adjacent to the rovers position. Two visual sensors were applied per quadrant. The first visual sensor, for quadrant q, returned the sum of red rocks in quadrant q divided by their discrete distance to the rover.

$$s_{1,q,\eta,t} = \sum_{q \varepsilon V_q} \frac{V_q}{\delta(L_q L_{\eta,t})} \tag{2}$$

Where, V_q was the number of red rocks on quadrant q, L_q was the location of quadrant q, and $L_{\eta,t}$ was the location of rover η at time t.

The second sensor returned the sum of discrete distances from a rover in quadrant q to all the other rovers at time t.

$$s_{2,q,\eta,t} = \sum_{\eta' \varepsilon N_q} \frac{1}{\delta(L_{\eta'} L_{\eta,t})} \tag{3}$$

Where, N_q was the set of rovers in quadrant q.

4) Evaluation Sensor: All red rocks on quadrant q ($\Re$ = $[r_0..r_k]$) were evaluated given that rover η and $\Re$ both occupied quadrant q. Red rocks either had a value of 0 or 1. If the value of $\Re$ for quadrant $q \geq 1$, then this value would then be marked for communication to the lander (next time the communications actuator was activated). The red rocks were then marked as *evaluated* so as they would not be evaluated again. After rover η had evaluated $\Re$ on quadrant q, it would receive an immediate private fitness reward calculated according to the value of $\Re$ on quadrant q (section IV-E).

[1]Rover parameter settings, such as battery energy, sensor and actuator costs, communication range and speed of movement are specified in previous work [4].

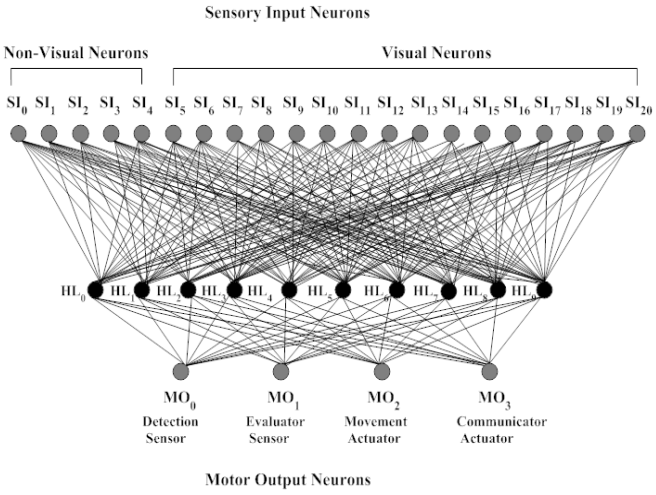

Fig. 2. Rover neural network controller (Section IV-D).

D. Rover Neural Controllers

Figure 2 presents a rover controller as a *Multi-Layer-Perceptron* (MLP) consisting of 21 sensory input nodes ($SI_0..SI_{20}$) and 4 motor output nodes ($MO_0..MO_3$) connected to 10 hidden layer nodes ($HL_0..HL_9$). $SI_0..SI_{20}$ accepted continuous inputs which represented 16 detection sensor inputs (8 adjacent quadrants with 2 detection sensors per quadrant), plus 5 non-visual sensors. This sensor vector constituted the state space for a rover. At each time step the rover used its current state to compute one of four possible actions. An action was either: *detect*, *evaluate*, *move*, or *communicate*. Each rover MLP performed this mapping from a rovers current state to its next action.

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 from the previous simulation iteration, respectively. Red rock evaluation was the value of red rocks on the quadrant that the rover was occupying during the previous simulation iteration, where this value was calculated by the rover's private fitness function (section IV-E). If the evaluation sensor was not activated during the previous simulation iteration then zero was returned. Previous motor outputs were teaching inputs [15] which influenced the next motor outputs. The 16 visual input nodes ($SI_5..SI_{20}$) represented the number of red rocks, and rovers detected in the 8 adjacent quadrants in the environment. That is, to the *north*, *south*, *east*, *west*, *northwest*, *northeast*, *southwest*, and *southeast*. All sensory input values were normalized.

2) Motor outputs: MO_0-MO_3 corresponded to the 4 actions a rover 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: The CONE method encoded a rover's controller as a set of 10 genotypes (hidden layer neurons). A single genotype was encoded a set of 25 connection weights. That is, 21 weights ($IW_0..IW_{20}$) connecting 21 sensory inputs, plus 4 weights ($OW_0..OW_3$) connecting 4 motor outputs to a given hidden layer neuron. The conventional NE method worked with genotypes that encoded the 250 connection weights of a complete controller.

E. Rover fitness functions

As with the experiments of [1], [21] we defined a global evaluation function, $G(z)$, which was a function of all environment variables and actions of all the rovers, z. The goal of the rover group was to maximize $G(z)$. However, the rovers did not maximize $G(z)$ directly. Instead each rover η attempted to maximize its private evaluation function $g_\eta(z)$.

It is important to note that $G(z)$ does not guide evolution, but rather provides a measure of rover group performance, based upon the contributions of individual rovers. Instead the private rover evaluation function guided the evolution of each rovers controller. In order to ensure that as each rover improves its private evaluation function it also improves the global evaluation function, a *difference evaluation function* was applied.

1) Global fitness evaluation: The global evaluation (fitness) function is given by $G(z)$, where z is the current state of the system. That is, the position of all the rovers in the group, their internal parameters and the state of the environment. The global evaluation function (G) calculated the sum of the value of red rocks evaluated (communicated to the lander) for all n rovers over the course of their lifetimes. G was defined as follows.

$$G = \sum_t \sum_i \frac{V_i}{min_\eta \delta(L_i L_{\eta,t})} \tag{4}$$

Where, V_i was the number of red rocks on quadrant i, L_i was the location of quadrant i, and $L_{\eta,t}$ was the location of rover η at time t. G is factored given that when rover η acts so as to increase G then g_η increases also. However, G has poor learnability given that its insensitive to individual rover actions and sensitive the actions of the group.

2) Private fitness evaluation: The private evaluation function (P) calculated the value of red rocks evaluated by a given rover over the course of its lifetime. P was defined as follows.

$$P_\eta = \sum_t \sum_i \frac{V_i}{min \delta(L_i L_{\eta,t})} \tag{5}$$

The P evaluation function is equivalent to the global evaluation function when there is only one rover. Since P was not affected by the actions of the other rovers, it had infinite learnability, but was not factored. Whilst any rover could detect red rocks in adjacent quadrants, the evaluation functions were only concerned with red rocks on the same quadrant as a rover (min $\delta(L_i L_{\eta,t})$ = 0). This was stipulated since a rover had to occupy the same quadrant as a red rock in order to evaluate it.

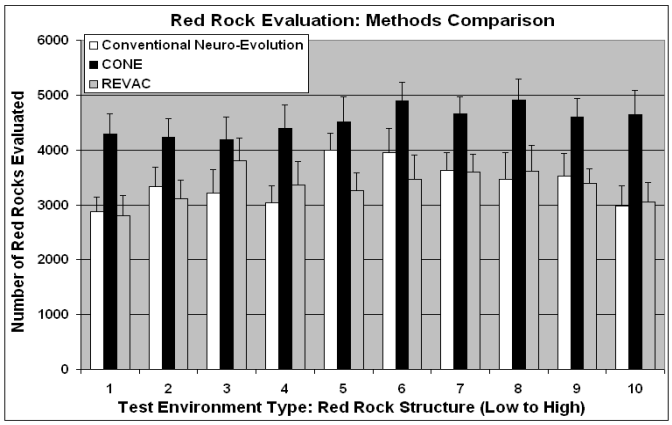

Fig. 3. Results from application of the conventional NE, CONE and heuristic (REVAC) methods to the extended multi-rover task in 10 different test environments (section IV-B).

3) Difference fitness evaluation: As elucidated by previous research [1], a difference evaluation function ensured that each local fitness evaluation function (P_η) was both *aligned* with the global evaluation function and was *learnable*. That is, each rover was able to readily observe how their behavior affected their evaluation function. Thus, D does not have as high learnability as P, but is still factored like G. In this case, the difference evaluation function, D, was defined as follows.

$$D_\eta = \sum_t [\sum_i \frac{V_i}{min_{\eta' \delta}(L_i L_{\eta',t})} - \sum_i \frac{V_i}{min_{\eta' \neq \eta \delta}(L_i L_{\eta',t})}]$$

$$= \sum_t \sum_i I_{i,\eta,t(z)} \frac{V_i}{min_\eta \delta(L_i L_{\eta,t})} \quad (6)$$

Where, $I_{i,\eta,t(z)}$ is the indicator function, returning one if and only if η is the rover occupying the quadrant L_i at time *t*. The second term of D is equal to the value of all red rocks collected if rover η were not in the system. For all time steps where η was not the rover occupying the same quadrant as a red rock, the subtraction (resulting in D) returned zero.

V. Heuristic Method

A set of *meta-control heuristics* was applied to the non-adaptive heuristic method experiments. In these experiments a heuristic controller was implemented for each rover in the group, and action selection was executed according to a set of probabilistic preferences (section V-A). Meta-control heuristics were implemented so as to direct rovers to perform particular actions when red rocks were detected. Specifically:

1) When red rocks were detected, the rover would move (in the next simulation iteration) to the quadrant with the highest number of red rocks. If this quadrant was fully occupied by other rovers, the quadrant with the second highest number of red rocks would be moved to, otherwise no move at all would be executed.
2) After red rocks with a value > 0, had been evaluated this value was communicated to the lander (in the next simulation iteration).

TABLE II

Heuristic Controller: Specialization was pre-defined at the individual rover (controller) level via giving a probabilistic preference each of 4 possible actions.

Rover Type	Detect	Evaluate	Move	Communicate
Detector	0.55	0.15	0.15	0.15
Evaluator	0.15	0.55	0.15	0.15
Communicator	0.15	0.15	0.15	0.55
Mover	0.15	0.15	0.55	0.15

A. Heuristic Controller

A heuristic controller was one of four specialized types (table II), and defined by a set of probabilistic preferences for selection of 1 of 4 actions (detection, evaluation, movement, or communication) at each simulation time step.

B. Non-Specialized Group

For the purposes of testing a non-adaptive heuristic controller in the rover domain, we used the REVAC (Relevance Estimation and Value Calibration) method to evolve a rover group composition from the four different specialized rover types (table IV). REVAC is a parameter calibration method which has been successfully applied to calibrate parameters in an agent-based evolutionary economics simulation [12], and for calibrating genetic algorithm parameters for numerous common objective functions [13].

The goal of REVAC was to evolve group compositions of individually specialized rovers. That is, the constituent portions of detectors, evaluators, communicators, and movers. We elected to evolve rover group compositions, and to specify individual rover specialization *a priori* (table II), as experiments that attempted to evolve individual specialization with REVAC did not succeed. For the purposes of these experiments, rovers having individual specializations was a prerequisite for defining group specialization. The calibration resulted in a non-specialized rover group (table III), where this group composition yielded the highest task performance in the calibration of rover group compositions.

C. Benchmark Comparison

The REVAC evolved group composition (table III) was subsequently executed in experiments using the heuristic controller. Since a non-specialized group was evolved, these results constituted a non-specialized task performance benchmark for the rover domain. The highest task performance results were attained in environment type 3, where an average of 3798 red rocks were evaluated (table V). Benchmark task performance results were compared with those of rover groups where neural controllers were applied and adapted with either the conventional NE (section III) or the CONE method (section II).

VI. Experiments

Experiments tested both NE and the heuristic method in 10 different environment types. Each environment type used

TABLE III

NON-SPECIALIZED ROVER GROUP COMPOSITION EVOLVED BY REVAC.

Detector Portion	Evaluator Portion	Mover Portion	Communicator Portion
0.33	0.38	0.26	0.03

a different red rock distribution (section IV-B). Each method was executed for 20 simulation runs. Averages presented in figure 3 were thus calculated over the 20 runs. The goal of the non-adaptive (heuristic) versus adaptive (NE) methods comparison was to elucidate the benefit of specialization (at both the individual and group levels) for task performance. The simulation and NE parameters are presented in table I.

A. Evolved Group Compositions

1) CONE Method: Table V presents the highest performing rover group (4907 red rocks evaluated) derived by the CONE method as being specialized at both the individual and at the group level. That is, this group (labeled a *detector group*) was comprised of the following portions of individual specializations: 52% *detectors*, 25% *evaluators*, 7% *communicators*, and 11% *movers*. 5% of the group did not derive an individual specialization.

2) Conventional NE Method: Table V presents the highest performing rover group (3988 red rocks evaluated) derived by the conventional NE method as being specialized at the individual level but unspecialized at the group level. That is, the group was comprised of the following portions of individual specializations: 37% *detectors*, 36% *evaluators*, 5% *communicators*, and 12% *movers*. 10% of the group did not derive an individual specialization.

B. 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 methods, and the heuristic controller method results. The data sets representing results of the conventional NE, CONE and heuristic methods were found to conform to normal distributions via applying the Kolmogorov-Smirnov test [6]. P = 1.0, 0.98, and 1.0 respectively. To determine the statistical significance of difference between data presented in figure 3 an independent t-test [6] was applied. We selected 0.05 as the threshold for statistical significance, and the null hypothesis was stated as the data sets not significantly differing.

First, we applied the t-test to the results of the heuristic controller method (using the REVAC evolved rover group composition presented in table III) and the conventional NE method. P = 0.32 was calculated, meaning the null hypothesis was accepted and there was no significant difference between these task performance results. This served to partially support the hypothesis that specialization was beneficial for task performance, via illustrating that the conventional NE method (which did not derive a specialized group) performed comparably to a non-adaptive heuristic controller method (using a previously evolved non-specialized group composition).

TABLE IV

EVOLVED ROVER GROUP COMPOSITIONS.

Rover Type	CONE	Conventional NE
Detector	0.52	0.37
Evaluator	0.25	0.36
Communicator	0.07	0.05
Mover	0.11	0.12
No-Specialization	0.05	0.1

Second, we applied the t-test to the conventional NE and CONE method results, and third, to the heuristic and CONE method results. The t-test yielded P = 0.0007 and 0.00003, respectively, indicating rejection of the null hypothesis and thus highlighting a significant difference between the CONE and conventional NE results, as well as between the CONE and heuristic method results (figure 3).

1) Measuring Specialization: Specialization was measured at the group, as well as the individual rover level. Specialized rovers were labeled as: detectors, evaluators, movers or communicators according to which sensor (detection or evaluation) or actuator (movement or communication) was used for $\geq 50\%$ of the rovers lifetime.

A group was likewise labeled *specialized* if $\geq$ 50% of rovers in the group were either detectors, evaluators, movers, or communicators. Table V presents the performance of the best performing group evolved using the CONE method, and labels it as a *detector* group, since (52% of rovers in the group were specialized as detectors (table IV).

Emergent specialization exhibited at both the individual (table IV presents 95% of the group as having an individual specialization) and group levels by the detector group, and its high task performance (comparative to the performance of the non-specialized group derived by the conventional NE method, and the heuristic benchmark) supports our research hypothesis (section I-A.3) for the rover domain.

The conventional NE method was successful in deriving specialization at the individual rover level, where these individually specialized rovers constituted 90% of the group (table IV). However, none of these individual specializations constituted $\geq$ 50% of the rovers in the group. As a result, the rover group derived by the conventional NE method performed comparably to the heuristic method, which used a previously evolved non-specialized group composition.

Supporting the performance gain results of similar neuro-evolution methods [9], the performance advantage yielded by the CONE method is theorized to be consequent of its fine-grained representation of the genotype space (figure 1). That is, the CONE method evolved a population of neurons for each rover controller in the group, where as, the conventional NE method evolved populations of complete controllers. We theorize that, construction of specialized controllers from individually specialized neurons thus made the CONE method appropriate for deriving a sufficient number of specialized controllers so as to warrant specialization at the group level.

TABLE V

PERFORMANCE OF GROUPS (INCLUDING THE TEST ENVIRONMENT THEY PERFORMED BEST IN) CONTROLLED BY THE CONVENTIONAL NE (A) CONE (B), AND HEURISTIC (C) METHODS.

Best Performing Group Derived With Method (A)		
Group Label	Environment Type	Performance
Non-specialized	5	3988
Best Performing Group Derived With Method (B)		
Group Label	Environment Type	Performance
Detector	8	4907
Best Performing Group Derived With Method (C)		
Group Label	Environment Type	Performance
Non-specialized	3	3798

VII. CONCLUSIONS

This paper described a NE method (CONE) that worked via having each agent in a group of agents to evolve its own population of neurons. At each evolutionary step of the CONE method, an agents neural controller was constructed from the fittest set of neurons within a given population. The performance of the CONE method was compared with a conventional NE method, that worked via having each agent in a group of agents evolve its own population of complete neural controllers.

These NE methods were comparatively tested in a collective behavior task domain, where specialization is known to be beneficial for task accomplishment. Specialization was defined at both the individual agent level, and at the agent group level. This was an extension of the rover task domain. The goal was for a group of agents (rovers) to detect and evaluate as many points of interest (red rocks) in a discrete two-dimensional environment, given limited energy, mission time, sensor, and actuator capabilities.

The research hypothesis was that the CONE method was appropriate for deriving emergent specialization for the benefit of increasing task performance. The CONE method was successful at deriving specialization at both the individual and group level, and consequently derived a rover group that yielded the highest task performance. Furthermore, rover groups controlled by the CONE method yielded a significantly higher group fitness (measured by a difference evaluation function) for all environment types tested.

The performance of rover groups controlled by the CONE method was compared with groups controlled by the conventional NE, as well as a heuristic controller. The conventional NE method was successful at deriving specialization at the individual rover level, but not at the group level. Rover groups controlled by the heuristic controller used individual specialization specified *a priori*, but did not use specialization at the group level. The performance of rover groups using the heuristic controller thus represented a non-specialized performance benchmark. The conventional NE and heuristic controlled groups were found to yield a comparative performance. Comparative to rover groups controlled by the CONE method (exhibiting specialization at both the individual and group levels), the inferior performance of rover groups controlled by the conventional NE and heuristic methods was theorized to be a consequence of the lack of group level specialization.

REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1–12, New York, USA, 2004. Springer-Verlag.

[2] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behavior. *Artificial Life*, 9(1):255–267, 2003.

[3] B. Bryant and R. Miikkulainen. Neuro-evolution for adaptive teams. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pages 2194–2201, Canberra, Australia, 2003. IEEE Press.

[4] A. Eiben, G. Nitschke, and M. Schut. Collective specialization for evolutionary design of a multi-robot system. In *Proceedings of the Second International Workshop on Swarm Robotics*. In press, 2006.

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

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

[7] 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*, 215(1):363–373, 2002.

[8] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(1):317–342, 1997.

[9] F. Gomez and R. Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1356–1361, Stockholm, Sweden, 2000. Morgan Kaufmann.

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

[11] D. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22(1):11–32, 1996.

[12] V. Nannen and A. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 183–190, Seattle, USA, 2006. ACM Press.

[13] V. Nannen and A. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the International Joint Conference on Atrificial Intelligence*, pages 975–980, Hyderabad, India, 2007. Morgan Kaufmann.

[14] S. Nolfi and D. Floreano. Learning and evolution. *Autonomous Robots*, 7(1):89–113, 1999.

[15] S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 1(5):75–98, 1997.

[16] P. Paalanen, J. Kamarainen, J. Ilonen, and H. Klviinen. Feature representation and discrimination based on gaussian mixture model probability densities. *Pattern Recognition*, 39(7):1346–1358, 2006.

[17] M. Potter and K. DeJong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[18] 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.

[19] K. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100, 2004.

[20] S. Thakoor. Bio-inspired engineering of exploration systems. *Journal of Space Mission Architecture*, 2(1):49–79, 2000.

[21] K. Tumer and A. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 591–598, Washington DC, USA, 2005. ACM Press.

[22] M. Waibel, D. Floreano, S. Magnenat, and L. Keller. Division of labor and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations. *Proceedings of the Royal Society B*, 273(1):1815–1823, 2006.

[23] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone. Evolving keepaway soccer players through task decomposition. In *Proceeding of the Genetic and Evolutionary Computation Conference*, pages 356–368, Chicago, 2003. AAAI Press.