Neuro-Evolution Approaches to Collective Behavior

G. S. Nitschke
Department of Computer Science, Vrije Universiteit, Amsterdam
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
nitschke@cs.vu.nl

***Abstract*—This paper is a preliminary study of the types of collective behavior tasks that are best solved by *Neuro-Evolution* (NE). This research tests a hypothesis that for a multi-rover task, the best approach (for deriving effective collective behaviors) is to evolve complete *Artificial Neural Network* (ANN) controllers, and then combine controller behaviors in a collective behavior context. Such methods are called multi-agent *Conventional Neuro-Evolution* (Multi-Agent CNE). This is opposed to methods such as *Enforced Sub-Populations* (ESP) which evolves individual neurons and then combines them to form complete ANN controllers. Single and multi-agent CNE and ESP approaches to evolving collective behavior solutions are tested comparatively in the multi-rover task. The multi-rover task requires that teams of rovers (controllers) cooperate in order to detect *features of interest* in a virtual environment. Results indicate that a Multi-Agent CNE approach derives rover teams with a higher task performance and genotype diversity, comparative to ESP.**

***Index Terms*—Neuro-Evolution, Collective Behavior, Rover.**

I. Introduction

Research in simulated (multi-agent) [20] and physical (multi-robot) [19] systems often attempt to replicate the success of biological collective behavior systems, such as social insect societies, for the purpose of solving collective behavior tasks. *Neuro-Evolution* (NE) has proven to be an effective method for designing *Artificial Neural Network* (ANN) controllers that cooperate in order to accomplish various collective behavior tasks with applications that include multi-agent computer games [3], collective gathering and construction [12], and coordinated movement [2] in multi-robot systems.

A key challenge in applying NE to evolve teams of cooperating ANN controllers that solve collective behavior tasks is selecting an appropriate genotype encoding. Given a direct encoding approach [9], and depending upon the collective behavior task, the evolution of complete controllers [15], versus functional units (for example, neurons [9] and weight sets [8]), can greatly affect evolved controller (and collective behavior) task performance.

Typical *Conventional NE* (CNE) methods directly encode genotypes as complete ANN controllers [21] and then evolve a fittest set of controllers for a given collective behavior task. In solving collective behavior tasks using CNE methods, either a single population [21] or multiple population cooperative co-evolutionary approach [17] is adopted.

In single population CNE methods, a fittest genotype (controller) is selected and copied n times to in order to form a homogenous team of controller clones. Alternatively, n fittest genotypes are selected in order to form a heterogenous team of controllers [18]. Both of these approaches have achieved success in various collective behavior tasks such as coordinated movement [13], and cooperative transport [14]. Multiple population CNE methods use a speciation inspired approach where n ANN controllers are evolved from n separate genotype populations. Such an approach has advantages in deriving teams of controllers to solve collective behavior tasks that require different controllers to adopt complementary and specialized behaviors [10].

Another approach to NE is to evolve functional units of a controller instead of entire ANN controllers. For example, the *Enforced Sub-Populations* (ESP) method works via allocating and evolving a separate neuron population for each of the hidden-layer neurons in a controller. ESP has been effectively applied to non-Markovian control tasks with sparse reinforcement such as double pole balancing, and rocket control [6], as well as to collective behavior tasks such as multi-agent computer games [3] and multi-agent pursuit-evasion tasks [22]. An extension of ESP is Multi-Agent ESP [22]. Multi-Agent ESP allocates one population for the evolution of each controller in a team of controllers. Each population is segregated into multiple sub-populations, where each sub-population evolves each hidden layer neuron in the controller derived from the given population.

An unresolved issue is what NE methods are most beneficial for solving collective behavior tasks. This research is a preliminary investigation of the types of collective behavior tasks for which an NE method that evolves complete controllers is most appropriate, versus, tasks for which an NE method that evolves neurons is most appropriate. In applying NE to solve collective behavior tasks, this is an important question, since methods that evolve and combine neurons are especially susceptible to a loss of genetic diversity within each sub-population, and may consequently evolve neurons that cooperate well with each other, though behave sub-optimally [7].

Furthermore, if a task does not require or benefit from controllers comprised of neurons specialized to different functions, then methods such as ESP will offer no advantage (or even be a disadvantage) in evolving collective behavior solutions. That is, methods such as ESP and Multi-Agent ESP applied to a collective behavior task, may lead to the evolution of neurons that cooperate well together in the form of a controller. However, due to evolution at the neuron level, each of the evolved controllers may converge to a behavior that does not perform well (that is, effectively cooperate with other controllers) in the context of a team.

This paper presents, for a multi-rover task, a collective behavior performance comparison of a single and multiple population implementation of CNE, and an identical implementation of ESP, for evolving teams of controllers.

978-1-4244-2959-2/09/$25.00

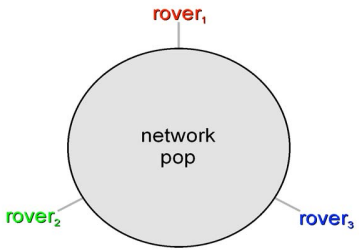

Fig. 1. Example of CNE. CNE is applied to evolve the controllers of three rovers. Each rover's controller is constructed via selecting one of the fittest genotypes (controllers) from the population.

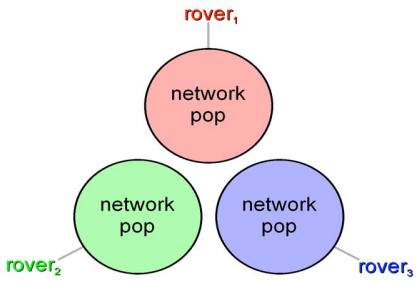

Fig. 2. Example of Multi-Agent CNE applied to evolve the controllers of three rovers. The controller of each rover is constructed via selecting one of the fittest neurons from each population. That is, each rover controller is evolved from its own population.

NE is selected as the collective behavior design approach, given that NE has been successfully applied to complex collective behavior tasks, for which there is no clear mapping between the sensory inputs and motor outputs of controllers that must cooperate [7], [2]. NE methods have also been successfully applied to the continuous multi-rover task [1]. However, extending this multi-rover task as a means of investigating the impact of an NE method that evolves and combines neurons (ESP), versus an NE method that evolves complete controllers (CNE), has not yet been investigated.

A. Research Goal:

To conduct a comparative study in order to evaluate CNE versus ESP as methods for deriving teams of controllers in the multi-rover task. The study is a preliminary investigation into which types of NE methods are appropriate for solving given types of collective behavior tasks. ESP and CNE exemplify two different types of NE methods.

B. Research Hypothesis:

For the multi-rover task, which requires a team of rover controllers to cooperate, the multiple population implementation of CNE (Multi-Agent CNE) is appropriate for deriving rover teams that achieve a higher task performance comparative to CNE, ESP, and Multi-Agent ESP rover evolved teams.

II. Methods

The *Enforced Sub-Populations* (ESP) and *Conventional Neuro-Evolution* (CNE) methods, both directly encode and evolve either neurons (ESP) or complete controllers (CNE). For both CNE and ESP, the crossover and mutation operators are the same. That is, a child genotype is produced via recombing two parent genotypes using single point crossover [4], and *burst* mutation with a *Cauchy* distribution [7]. Mutation changes each gene (connection weight) by a random value in the range [-1.0, +1.0] with a 0.05 degree of probability. Each gene is kept within the range [-10.0, +10.0]. Burst mutation is used to ensure that most weight changes are small whilst allowing for larger changes to some weights.

A. CNE and Multi-Agent CNE

The CNE method is similar to that described by Wieland [21], and operates with either one or n genotype (controller) populations. One genotype encodes all the parameters (input and output connection weights) of a complete ANN controller.

In CNE, one population of controllers is evolved (figure 1: left). Each controller is systematically selected from an *elite portion* (table I) along with n-1 (randomly selected) other controllers, and evaluated in the context of n controllers. The controller is then assigned a fitness. In this research, CNE uses *heterogenous* selection to form a team of n controllers. That is, n controllers are selected from the population's elite portion, where the same controller cannot be selected twice.

In Multi-Agent CNE (figure 2: right) n populations of controllers are evolved. The process used by CNE for selecting a controller from a population is repeated for n-1 other populations. That is, a controller (to be evaluated) is selected from the elite portion of a given population. This controller is then evaluated together with n-1 other controllers. Each of the other n-1 controllers is formed via randomly selecting one controller from each of the other n-1 populations.

For both CNE and Multi-Agent CNE, in order to reduce the chance that a controller is rewarded a high fitness for participating in a *lucky* team, each controller is evaluated r times in the same task trial (together with the same n-1 controllers). An *average fitness* is calculated for each controller as the sum of fitness achieved for all task trials divided by the number of trials the controller participates in.

The key difference between CNE and Multi-Agent CNE is how a team of n controllers is formed. In CNE, a controller team is formed via repeatedly selecting controllers from one population. In Multi-Agent CNE, a controller team is formed via selecting one controller from each of n populations. The evaluation and assignment of fitness to all controllers constitutes one generation in CNE and Multi-Agent CNE.

1) Genotypes: Genotype a (equation 1) directly encodes a controller, where, a is a string of 200 floating point values, representing weights connecting 16 sensory input neurons and 9 motor output neurons to 8 hidden layer neurons (figure 6).

$$\bar{a} =< a_0, a_1, \ldots, a_k > \quad (1)$$

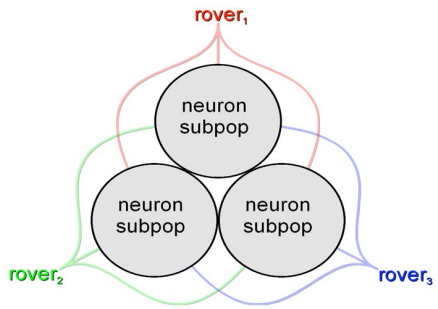

Fig. 3. Example of ESP applied to the multi-rover task. Each rover's controller (containing three hidden layer neurons) is formed via selecting a fittest neuron from each sub-population.

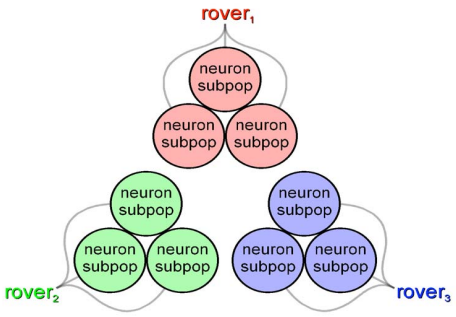

Fig. 4. Example of Multi-Agent ESP applied to the multi-rover task. Each rover's controller is formed via selecting a fittest neuron from each sub-population (for each population). Each controller is evolved from its own population, where each population comprises three sub-populations (the number of hidden layer neurons in a controller).

Genotype *a* consists of *k* genes, where each gene represents an input or output connection weight value.

2) Genotype Evaluations: For CNE and Multi-Agent CNE, each controller in a given population is successively selected and evaluated in the context of a rover team. That is, a given controller is evaluated according to how well it performs together with *n*-1 other (randomly selected) controllers.

3) Recombination: At each generation's end, the elite portion of controllers are recombined (via randomly pairing all parent neurons in each population's elite portion). The parent controllers produce enough child controllers in order to completely replace all controllers in each of the populations.

4) Number of Genotype (Controller) Evaluations: *q* (task trials) x *r* (controller evaluations) x *m* (total number of controllers) *n* (team size). ESP and Multi-Agent ESP (section II-B) use the same number of genotype evaluations.

B. ESP and Multi-Agent ESP

As illustrated in figure 3, ESP allocates and evolves a separate neuron population for each of *u* hidden-layer neurons in an ANN. A neuron can only be recombined with other neurons from its own population. ESP is further described in related work [6].

Multi-Agent ESP [22] is an extension of ESP, and creates *n* populations for deriving *n* controllers. As illustrated in figure 4, each population consists of *u* sub-populations, where the fittest neuron is selected from each sub-population to form a hidden layer of one ANN. This process is repeated *n* times for the *n* controllers. In Multi-Agent ESP there is *no recombination* between (only *within*) sub-populations in different populations.

1) Genotypes: Genotype *a* (equation 1) directly encodes the values that represent the weights connecting all input neurons and all output neurons to one hidden layer neuron (table I), where *a* is a string of 25 (16 input + 9 output weights) floating point values. In genotype a, a_0 is the genotype's tag, and a_i $(1 \leq i \leq k)$ is a neuron connection weight. Each connection weight is a floating point value normalized in order to be within the range: [0, 1]. The genotype's tag specifies which sub-population (that is, which position in the hidden layer of an ANN) the genotype is assigned to. The first gene is not subject to the evolutionary operators or process of ESP.

2) Genotype Evaluation: In ESP [6], each genotype (hidden layer neuron) in each of *u* sub-populations is systematically selected from the sub-population's elite portion (table I), evaluated in the context of a controller and a team of controllers, and assigned a fitness. A given neuron is assigned a fitness according to how well it performs in the context of a controller (cooperates with *u*-1 other randomly selected neurons), as well as how well it performs with *n*-1 other controllers. That is, given that a neuron has been selected for evaluation from one of the *u* sub-populations, and a controller that includes this neuron is constructed, another *n*-1 controllers are constructed. These other *n*-1 controllers are formed via randomly selecting *u* x (*n*-1) neurons from the same *u* sub-populations.

In Multi-Agent ESP [22], the process used by ESP for selecting and constructing a controller from *u* sub-populations is repeated for the *n*-1 other populations. That is, a neuron (to be evaluated) is selected from one of *u* sub-populations (in a given population). The other *u*-1 neurons are randomly selected from the other *u*-1 sub-populations (in the given population). This controller is then evaluated together with *n*-1 controllers, where each of these other controllers is formed via randomly selecting *u* neurons from each of the *u* sub-populations in each of the other *n*-1 populations.

In order to reduce the chance that a neuron is rewarded a high fitness for participating in a *lucky* controller, each genotype is evaluated in *r* times together with *u*-1 other neurons, as well as with the same *n*-1 controllers (in the same task trial). An average fitness is calculated for each neuron as the sum of fitness accumulated for all trials divided by the number of trials the neuron participates in.

The key difference between ESP and Multi-Agent ESP is how each of the *n* controllers is formed. In ESP, each of the controllers is formed via repeatedly selecting neurons from the *u* sub-populations. In Multi-Agent ESP, each of the controllers is formed via selecting *u* neurons from each of the populations.

The assignment of fitness to all neurons (in all *u* sub-populations in each of *n* populations) constitutes one generation in the ESP (Multi-Agent ESP) evolutionary process.

3) Recombination: At the completion of each generation, the elite portion of neurons in each sub-population are recombined (via randomly pairing all parent neurons within each

sub-population's elite portion). The parent neurons produce enough child neurons in order to completely replace all neurons in each of the sub-populations.

4) Number of Genotype (Neuron) Evaluations: n (team size) x u (sub-populations per population) x $\frac{m}{U}$ (neurons per sub-population) x r (neuron evaluations) x q (task trials). Where, U is the total number of sub-populations, and u is the number of sub-populations per population.

III. Extended (Continuous) Multi-Rover Task

The extended multi-rover task is an extension of the continuous multi-rover task described by Agogino and Tumar [1]. Each rover attempts to maximize the value of red rocks detected over the course of its lifetime as well as to maximize the value of red rocks detected over the course of the rover team's lifetime. The extended multi-rover task includes the possibility for a rover to execute multiple actions (section III-G). These actions give rovers the possibility of exploiting specialization to a given action as a means of increasing its own (and the team's) task performance. Related research in the extended multi-rover task [11], elucidated that behavioral specialization increases task performance at both the individual and rover team level. Furthermore, this related research demonstrated that a systematic search method for controlling a rover team is not appropriate for attaining an optimal task performance in the extended multi-rover task, where, rover's are constrained by limited energy, and sensor and actuator capabilities.

Table I presents the multi-rover simulation parameters. These include NE and rover parameter settings such as battery energy, sensor and actuator costs, sensor range and speed of rover movement. The range values presented for rover movement correspond to percentage values of the environment size, where the environment width and height are equal for these experiments. Environment parameter settings are those that define the simulation environment, such as the number of rovers, width and height of the environment and the individual and total value of red rocks in the environment.

Multi-Rover Simulation Parameters	
Number of generations	1000
Number of epochs per generation	32
Number of task trials per epoch	4
Iterations per trial (Rover lifetime)	1000
Number of rovers	24
Number of red rocks	720
Total red rock value	36000
Individual red rock value	50
Red rock deviation from centroid positions	Variable
Number of red rock sensors per rover	4
Number of rover sensors per rover	4
Red rock / rover sensor range	0.08
Red rock / rover sensor accuracy	1.0
Red rock detection sensors cost	0.05
Rover detection sensors cost	0.0
Rovers required to detect a red rock	2
Number of ANN sensory input neurons	16
Number of ANN hidden layer neurons	8
Number of ANN motor output neurons	9
Elite portion	0.25
Mutation probability per gene	0.05
Rover movement range	0.01
Rover movement cost	1.0
Rover initial energy	1000 units
Initial rover positions	Random
Environment width	1.0
Environment length	1.0

TABLE I
Multi-rover simulation parameters.

Red Rock/Rover Detector Sensor Setting	Red Rock/Rover Type Detected	Accuracy	Range	Cost
Setting A	A	1.0	0.08	0.05
Setting B	B	1.0	0.08	0.05
Setting C	C	1.0	0.08	0.05

TABLE II
Red Rock / Rover detection sensor settings.

A. Collective Red Rock Detection

Red rock value detection requires that at least two rovers concurrently activate their red rock detection sensors (figure 5). Each rover's sensors' must be activated with a setting that corresponds to the type of red rock being detected (table II). If at least two rovers collectively detect a red rock, it is marked as *detected* and its value recorded. The red rock is then removed from the environment so it is not detected again.

B. Continuous Simulation Environment

The simulation environment is a two dimensional continuous plane, where one rover can occupy any x, y position. Movement is calculated in terms of real valued vectors. Distance calculation to other rovers and red rocks uses the squared Euclidean norm bounded by a minimum distance [1].

C. Red Rock Value Distribution

The environment is populated with 720 red rocks, distributed according to a two dimensional *Gaussian mixture model* [16]. The mixture model is specified with four centroids, set in static locations, where the radius of each determines the spatial distribution of red rocks. Distributions with various radii (ρ) were tested[1], and $\rho = 0.65$ was selected, given that this radius, for all NE methods applied to controller evolution, enabled the derivation the highest performing rover teams. The radii values are percentages of environment size, where the environment width and height are equal. Each red rock has an integer value equal 50, and locations are random within a given distribution and thus initially unknown to rovers.

[1]A total of 90 ρ values ranging from 0.05 to 0.95 (in increments of 0.01) were tested. These ρ values were selected since they produced red rock distributions that ranged from four clusters to approximately uniform.

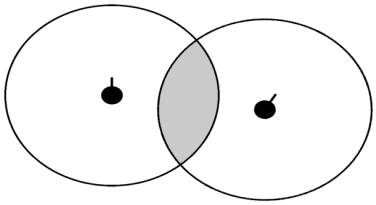

Fig. 5. Collective red rock detection. To detect the value of a red rock of a given type, at least two rovers are required to activate their detection sensor settings with a setting that corresponds to the red rock type (illustrated as the joint area in the two detection fields).

D. Red Rock Detection Sensors

Four red rock detection sensors covering four sensor quadrants provide a rover with a 360 degree *Field Of View* (FOV). Red rock detection sensors need to be explicitly activated with one of three settings, and have a fixed accuracy, range, and cost (table II). Sensor activation consumes one simulation iteration. Detection sensor settings are: *A*, *B*, and *C*, which allow a rover to detect type *A*, *B*, and *C* red rocks, respectively. *Accuracy* denotes the degree of probability with which red rocks are detected. *Range* is defined as a portion of the width of the environment, where width and length are equal. *Cost* is the energy used when the red rock detection sensors are activated.

When red rocks come within range of a red rock detection sensor, then that sensor is activated with a value inversely proportional to the distance to the closest red rock. That is, red rock detection sensor q, returns the inverse of the distance between *this* rover (v) and the location of the closest red rock in quadrant q (equation 2).

$$S_{1(q,t)} = j \epsilon J_q \frac{1}{\delta(L_{v,t}, L_{j,t})} \tag{2}$$

q is a sensor quadrant,
v is *this* rover,
t is simulation time step t,
j is the closest red rock to rover v,
J_q is the set of all red rock values in quadrant q,
$L_{j,t}$ ($j \epsilon J_q$) is the location of red rock j at time t,
$L_{v,t}$ is the location of rover v at time t.

E. Rover Detection Sensors

Four rover detection sensors covering four sensor quadrants provide a rover with a 360 degree FOV. Rover detection sensors need to be explicitly activated with one of three settings and have a fixed accuracy, range and cost (table II). Sensor activation consumes one simulation iteration. When another rover comes within range of a rover detection sensor, then the sensor is activated with a value inversely proportional to the distance to the other rover. Detection of other rovers also accounts for the *other* rover's red rock detection sensors setting. That is, sensor q returns a value corresponding to the red rock detection sensor setting being used by the closest rover (v'), divided by the distance between *this* rover and rover v' (equation 3).

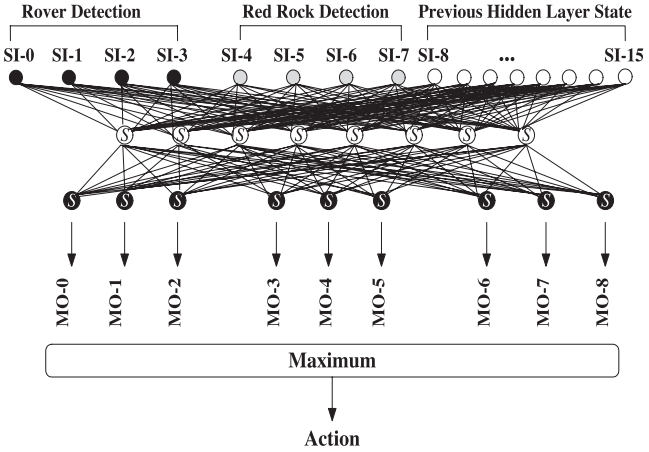

Fig. 6. Rover ANN controller. SI: Sensory Input, MO: Motor Output.

$$S_{2(q,v,t)} = \frac{dv'}{\delta(L_{v'}, L_{v,t})} \tag{3}$$

q is a sensor quadrant,
v is *this* rover (the rover that is detecting other rovers),
t is simulation time step t,
v' is the closest rover to *this* rover in quadrant q,
dv' is the red rock detection sensor setting of rover v'. dv' is: 1, 2, or 3 (setting: A, B, or C, respectively),
$L_{v'}$ is the location of the closest rover in quadrant q,
$L_{v,t}$ is the location of *this* rover at time t.

Sensor values are normalized within the range [0.0, 1.0].

F. Movement Actuators

A rover's heading is calculated from two motor output values in its controller. That is, MO-7 (dx) and MO-8 (dy) in figure 6. A rover's heading is determined by normalizing and scaling these vectors by the maximum distance a rover can traverse in one simulation iteration. That is: $dx = d_{max}$(MO-7 - 0.5), and $dy = d_{max}$(MO-8 - 0.5). Where, d_{max} is the maximum distance a rover can move in one iteration.

G. Artificial Neural Network Controller

A recurrent ANN maps sensory inputs to motor outputs (figure 6), where 16 sensory input neurons ([SI-0, SI-15]) are fully connected to 8 hidden layer neurons. Input neurons [SI-0, SI-3] accept inputs from four rover detection sensors. Input neurons [SI-4, SI-7] accept inputs from 4 red rock detection sensors. Input neurons [SI-8, SI-15] accept hidden layer neuron activation values from the previous simulation iteration. Motor outputs ([MO-0, MO-8]) are fully connected to 8 hidden layer neurons. Hidden and output neurons are sigmoidal units. Outputs are normalized in the range: [0, 1].

1) Action Selection:: Each time step, the motor output (figure 6: MO-0 to MO-8) with the highest value is the action executed.

Neuro-Evolution Method	Diversity	Performance
ESP	0.0835 (± 0.0055)	0.4970 (± 0.0031)
Multi-Agent ESP	0.3768 (± 0.0118)	0.5590 (± 0.0082)
CNE	0.0798 (± 0.0014)	0.3638 (± 0.0049)
Multi-Agent CNE	0.3754 (± 0.0201)	0.6059 (± 0.0056)

TABLE III
AVERAGE TASK PERFORMANCES AND DIVERSITY OF ESP, MULTI-AGENT ESP, CNE, AND MULTI-AGENT CNE EVOLVED TEAMS (STANDARD DEVIATIONS IN PARENTHESES).

1) MO-0: Activate red rock detection sensors (*setting A*).
2) MO-1: Activate red rock detection sensors (*setting B*).
3) MO-2: Activate red rock detection sensors (*setting C*).
4) MO-3: Activate rover detection sensors (*setting A*).
5) MO-4: Activate rover detection sensors (*setting B*).
6) MO-5: Activate rover detection sensors (*setting C*).
7) MO-6: Rover becomes (or remains) idle.
8) MO-7, MO-8: Move. Direction calculated from *dx*, *dy*.

H. Rover Team Fitness Evaluation

This section details the global evaluation function used to evaluate rover team performance, and the private evaluation function used to evaluate individual rover performance (fitness). Global and private evaluation functions calculate fitness as a function of the total *red rock value detected*.

1) Private Fitness Function: $g_{\eta,v}$ calculates the *value of red rocks detected* by the detection sensors (v) of rover η over the course of its lifetime. Equation 4 presents $g_{\eta,v}$.

$$g_{\eta,v} = \sum_{0 \leq t \leq T} \sum_{j \epsilon J_{t,v}} \frac{rv_{j,t}}{min(\delta(L_{v,t}, L_{j,t}))} \quad (4)$$

v is the *value of red rocks detected* by the detection sensors of rover η,
t is simulation time step t,
T is the total number of simulation time steps,
$rv_{j,t}$ is the value of red rock j at time t,
$J_{t,v}$ is the set of red rock values within detection sensor range of rover v and detected by rover v,
$L_{j,t}$ ($j \epsilon J_q$) is the location of red rock j at time t,
$L_{v,t}$ is the location of rover v at time t,
min $(\delta(L_{v,t}, L_{j,t}))$ is the minimum distance between rover v at time t, and red rock j at time t.

2) Global Fitness Function: G calculates the sum of the *value of red rocks detected* by a rover team. The goal of a rover team is to maximize G. However, rovers do not maximize G directly. Instead each rover η attempts to maximize its own private fitness function $g_{\eta,v}$. It is important to note that G does not guide evolution, but rather provides a measure of rover team performance, based upon the contributions of individual rovers. Rather, $g_{\eta,v}$ guides rover controller evolution. For any given experiment, the average *red rock value detected* is calculated over all epochs of all rover lifetime's. The highest *red rock value detected* is selected in order to calculate G (equation 5).

$$G = \sum_{\eta \epsilon V} g_{\eta,v} \quad (5)$$

Where, V is the set of all rovers.

IV. EXPERIMENTS

Experiments apply ESP and CNE for evolving controllers in teams of 24 rovers in a given environment (section III-C). Each experiment executes a method for 20 simulation runs (table I). A simulation consists of 500 generations. A generation corresponds to the *lifetime* of each rover. A rover lifetime lasts for 32 epochs. An epoch is a set of 4 task trials that test different rover starting positions and orientations in the environment. Each task trial consists of 1000 simulation iterations.

A. CNE and Multi-Agent CNE Experimental Setup

In the single population implementation of CNE, one population is initialized with 24000 genotypes. A heterogenous rover team is created via randomly selecting 24 genotypes from the elite portion (table I) of the population. These 24 genotypes are then decoded into a team of 24 rover controllers.

In Multi-Agent CNE, one genotype population is initialized for each of the 24 rovers, where each of the 24 populations contains 1000 genotypes. A heterogenous rover team is created via randomly selecting one genotype from the elite portion of each of the 24 populations. These 24 genotypes are then decoded into a team of 24 rover controllers.

B. ESP and Multi-Agent ESP Experimental Setup

In the single population implementation of ESP, eight sub-populations are initialized, where each sub-population contains 3000 genotypes. A heterogenous rover team is created via randomly selecting one genotype from each sub-population's elite portion. These eight genotypes are then decoded into neurons which form the hidden layer of a controller. This process is repeated 24 times in order to form a rover team.

In Multi-Agent ESP, one genotype population is initialized for each of the 24 rovers, where each of the 24 populations contains eight sub-populations, and each sub-population contains 125 genotypes. A heterogenous rover team is created via deriving one rover controller from each population. That is, for each of the 24 populations, one genotype is randomly selected from the elite portion of each sub-population. These eight genotypes are then decoded into neurons which form the hidden layer of a rover controller.

C. Method Comparisons

Rover teams evolved by CNE, ESP, Multi-Agent CNE and ESP are compared with respect to their average task performance (fitness) and genotype diversity.

1) Performance Analysis: Figure 7 shows the average fitness of teams at each generation when evolved with CNE, Multi-Agent CNE, ESP, and Multi-Agent ESP. Fitness is presented as the portion of red rocks detected by a team, where a value of 1.0 indicates that all red rocks have been detected.

CNE evolved teams yield a higher task performance comparative to Multi-Agent CNE (and a task performance approximately equal to Multi-Agent ESP evolved teams) for the first 250 generations. CNE evolved teams achieved the highest task performance at generation 300, and thereafter task performance remains approximately constant. As expected, CNE quickly converged to an initially effective behavior. This in turn leads to the evolution of a fittest team of genetically similar controllers that converge (early in the evolutionary process) to a sub-optimal solution (table III: 0.3638 of optimal task performance). Poor and steady performance between generations 300 and 1000 supports this statement.

ESP evolved teams did not yield a higher performance, comparative to CNE evolved teams, until after generation 400. Between generations 500 and 1000, ESP evolved teams achieved a task performance higher comparative to CNE and lower comparative to Multi-Agent CNE and Multi-Agent ESP evolved teams. The ESP sub-population architecture was successful in maintaining genotype diversity until generation 400. However, selecting neurons for all controllers from the same set of sub-populations, means that all controllers are likely to converge to a sub-optimal solution, given that each sub-population is also likely to converge to a set of neurons that cooperate well, though ultimately constitute poor performing controllers (in terms of their cooperation in the context of a rover team). This is supported by table III which presents ESP evolved teams as attaining 0.4970 of optimal task performance.

Multi-Agent CNE uses a multi-population cooperative co-evolution architecture that encourages convergence to a team of controllers that yield a steadily increasing task performance. That is, selection of a fittest controller from each population and the collective evaluation of controllers as a rover team facilitates the evolution of a set of complementary behaviors. Collectively, these behaviors constitute a high performance team. The average task performance of Multi-Agent CNE evolved teams are not optimal (table III: 0.6059 of optimal task performance), though, it is significantly higher comparative to CNE, ESP, and Multi-Agent ESP evolved teams.

Multi-Agent ESP also uses a multi-population cooperative co-evolution architecture. Experimental results indicate that, whilst such an architecture is generally beneficial, the evolution of neurons as part of this architecture is not as beneficial as the Multi-Agent CNE network based evolution. That is, from each population, a set of fittest neurons that work well together are selected in order to form a controller. However, these neurons did not produce controllers that were able to cooperate effectively in order to yield a high task performance. This is supported by the significantly lower average task performance (table III: 0.5590 of optimal task performance) of Multi-Agent ESP evolved teams, comparative to Multi-Agent CNE teams.

The Kolmogorov-Smirnov (KS) test [5] confirms that task performance results of CNE, Multi-Agent CNE, ESP, and Multi-Agent ESP evolved teams (figure 7) conform to normal distributions. In order to determine if there is a statistical significance of difference between the average task performances of rover teams evolved by CNE, ESP, Multi-Agent CNE and Multi-Agent ESP, an independent t-test [5] is applied. A statistical significance of 0.05 is selected, and the null hypothesis is stated as the data sets not significantly differing. These t-tests[2] found that teams evolved by Multi-Agent CNE yield a significantly higher task performance over teams evolved by the other methods. This result supports the hypothesis that Multi-Agent CNE, on average, evolves teams yielding a higher task performance comparative to other methods (section I).

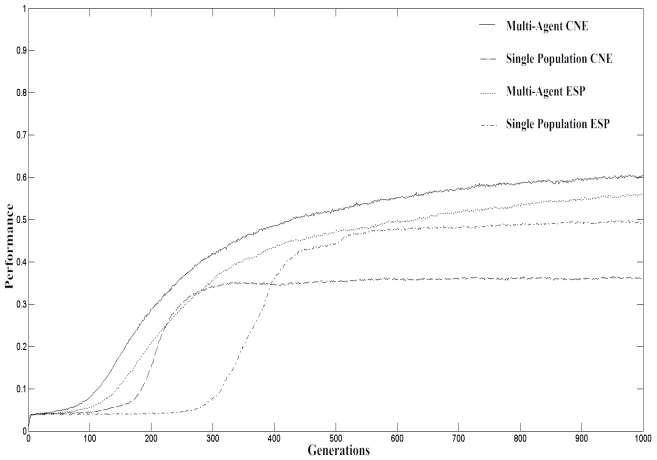

Fig. 7. Progression of average task performance (calculated over 20 simulations) of rover teams evolved by ESP, Multi-Agent ESP, CNE, and Multi-Agent CNE. The curves demonstrate the short term performance gains of CNE and the long term performance gains of Multi-Agent CNE.

2) Genotype Diversity Analysis: The average genotype diversity exhibited by different methods throughout evolution is also investigated. This diversity analysis demonstrates that the Multi-Agent (cooperative co-evolution) methods enable greater diversity between genotypes, which in turn facilitates the evolution of high performance rover teams. For teams evolved by each method, genotype diversity is measured after each generation of the 1000 generations of execution.

To measure the genetic similarity between two genotypes $\bar{a}$ (equation 1) and $\bar{b}$ (identical in structure to $\bar{a}$), a *Genetic Distance* (GD) between $\bar{a}$ and $\bar{b}$ is defined (equation 6).

$$GD(\bar{a}, \bar{b}) = \frac{\sum_{i \in \{1,..,N\}} |a_i - b_i|}{N} \quad (6)$$

Where, N is the genotype length.

Figure 8 presents the average genotype diversity of teams evolved by each method measured at each generation, where a value of 1.0 indicates maximum diversity. The cooperative co-evolutionary methods (Multi-Agent CNE and Multi-Agent ESP) maintain a higher level of diversity over all generations. For the first 200 generations, Multi-Agent CNE evolved teams maintained a higher diversity comparative to Multi-Agent ESP evolved teams. This diversity coincides with a comparatively

[2]KS and t-tests P values are not presented here due to space restrictions.

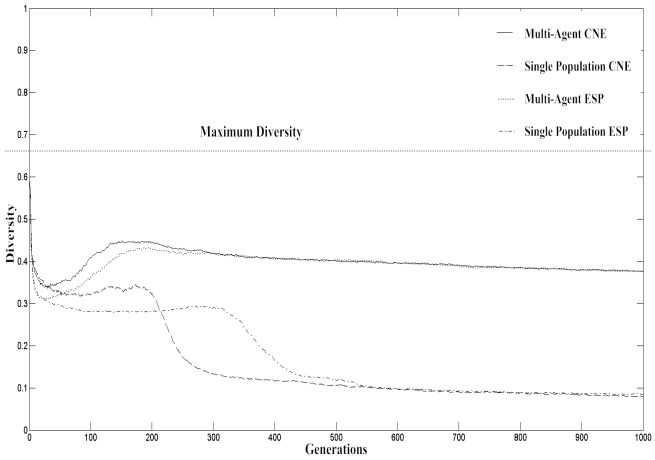

Fig. 8. Progression of average genotype diversity (calculated over 20 simulations) of rover teams evolved by ESP, Multi-Agent ESP, CNE, and Multi-Agent CNE. Multi-Agent CNE and Multi-Agent ESP evolved teams maintain the highest level of genotype diversity throughout evolution.

higher (approximately 10%) average task performance yielded by Multi-Agent CNE evolved teams (figure 7).

This result, supported by related work [9], is theorized to be consequent of the speciated nature of the multi-agent methods, which encourages the evolution of teams that converge to multiple complementary behaviors. This result also indicates that the multi-rover task is suited to multi-agent evolution at the controller level and not at the neuron level. The neuron based evolution used by Multi-Agent ESP, whilst maintaining a comparably high level of diversity, converges to teams that yield a significantly lower average task performance.

Also, figure 8 presents the average genotype diversity maintained by CNE evolved teams as being higher comparative to the average diversity of ESP evolved teams for the first 200 generations. However, between generations 200 and 400 the average diversity of CNE evolved teams is lower than that of ESP evolved teams. Interestingly, the low diversity of CNE evolved teams during this period coincides with an on average 30% higher task performance (figure 7). That is, CNE causes teams to quickly converge to a set of genetically non-diverse yet effective controllers. However, the average task performance (figure 7) and diversity (figure 8) of CNE evolved teams remains constant between generations 300 and 1000.

V. Conclusions

This research was a preliminary exploration into a collective behavior task that benefits from the evolution of genotypes that encode complete ANN controllers, as opposed to genotypes that encode individual neurons. Experiments were conducted that compared the task performances of simulated rover teams operating with ANN controllers. Controllers were evolved with the CNE, Multi-Agent CNE, ESP, and Multi-Agent ESP methods in a multi-rover task. The multi-rover task is a collective behavior task that requires teams of rovers to maximize a value for *features of interest* detected on a virtual landscape, given time, energy, sensor and actuator constraints. Results support the research hypothesis. That is, Multi-Agent CNE is appropriate for deriving teams that yield a higher task performance comparative to teams evolved by the CNE, ESP and Multi-Agent CNE methods. Thus, for this multi-rover task, the ESP approach which evolves neurons separately and then combines them into complete ANN controllers, offers no advantage. The multi-rover task is the first in a set of case studies which investigate the types of collective behavior tasks that benefit from NE at the neuron level versus NE at the controller level.

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 Congress on Evolutionary Computation*, pages 2194–2201, Canberra, Australia, 2003. IEEE Press.

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

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

[6] F. Gomez. *Robust Non-Linear Control Through Neuroevolution*. *PhD thesis*. Computer Science Dept, University of Texas, Austin, USA, 2003.

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

[8] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In *Machine Learning: ECML 2006*, pages 654–662, Berlin, Germany, 2006. Springer.

[9] D. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(1):373–399, 1998.

[10] G. Nitschke and M. Schut. Designing multi-rover emergent specialization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, 2008.

[11] G. Nitschke, M. Schut, and A. Eiben. Collective specialization for evolutionary design of a multi-robot system. In *Proceedings of the Second International Workshop on Swarm Robotics*, pages 189–206, Rome, Italy, September 2006. Springer.

[12] G. Nitschke and D. van Krevelen. Neuro-evolution for a gathering and collective construction task. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, 2008.

[13] S. Nolfi, G. Baldassarre, and D. Parisi. Evolution of collective behaviour in a team of physically linked robots. In *Applications of Evolutionary Computing*, pages 581–592. Springer, Heidelberg, Germany, 2003.

[14] S. Nolfi, J. Deneubourg, D. Floreano, L. Gambardella, F. Mondada, and M. Dorigo. Swarm-bots: Swarm of mobile robots able to self-assemble and self-organize. *Ecrim News*, 53(1):25–26, 2003.

[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. Kälviäinen. Feature representation and discrimination based on gaussian mixture model probability densities. *Pattern Recognition*, 39(7):1346–1358, 2006.

[17] M. Potter and K. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):10–29, 2000.

[18] M. Quinn. A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Proceedings of the Congress Evolutionary Computation*, pages 128–135, Seoul, South Korea, 2001. IEEE Press.

[19] C. Schultz and L. Parker. In *Multi-robot Systems: From Swarms to Intelligent Automata*. Kluwer, Washington DC, USA, 2002.

[20] 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.

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

[22] C. Yong and R. Miikkulainen. *Coevolution of Role-Based Cooperation in Multi-Agent Systems*. *Technical Report AI07-338*. Department of Computer Sciences, The University of Texas, Austin, USA, 2007.