Designing Multi-Rover Emergent Specialization

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

ABSTRACT

We compare the efficacy of the *Enforced Sub-Populations* (ESP) and *Collective Neuro-Evolution* (CONE) methods for designing behavioral specialization in a multi-rover collective behavior task. These methods are tested for *Artificial Neural Network* (ANN) controller design in an extension of the multi-rover task, where behavioral specialization is known to benefit task performance. The task is for multiple simulated autonomous vehicles (rovers) to maximize the detection of points of interest (red rocks) in a virtual environment. The task requires rovers to collectively sense such points of interest in order for them to be detected. Results indicate that the CONE method facilitates a level of specialization appropriate for achieving a significantly higher task performance, comparative to rover teams evolved by the ESP method.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms

Algorithms

Keywords

Neuro-Evolution, Collective Behavior, Specialization

1. INTRODUCTION

Research in simulated and physical collective behavior systems, has often attempted to replicate the success of certain biological social systems at decomposing the labor of a group into composite specialized and complementary roles in order 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 [7], multi-agent [13], 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 are currently lacking.

This paper describes the application of CONE (section 4) and ESP [8] to the *extended multi-rover task.* This task extends that described by [1] and requires solutions for controlling teams of simulated planetary exploration rovers that seek to maximize the value of points of interest (*red rocks*) detected in a continuous simulation environment. Rover ANN controllers are evolved in order to control rover sensor and actuators functionality, for the purpose of maximizing red rock detection. The research of [10] elucidated that behavioral specialization (at either the individual rover or team level) is beneficial for task performance in this extension of the multi-rover task. Furthermore, [10] demonstrated that heuristic based systematic search methods are not appropriate for achieving optimal performance in the extended multi-rover task, and that such heuristic methods are significantly out-performed by NE multi-agent control methods.

1.1 Neuro-Evolution (NE)

NE is selected as the collective behavior design approach, given that NE has been successfully applied to complex collective behavior control tasks, for which there is no clear mapping between sensory inputs and motor outputs [8]. Furthermore, NE is most appropriately applied to multi-robot tasks for which there is no clear mapping between sensory inputs to motor outputs [2]. NE methods have been successfully applied to the multi-rover task [1]. However, extending this task to include the notion of using NE to facilitate emergent specialization, in order to increase task performance, has not yet been investigated.

1.2 Research Goal

To conduct a comparative study in order to evaluate the CONE versus the ESP methods for deriving ANN controllers in the multi-rover task. We aim to support the research hypothesis that the CONE method derives a level of behavioral specialization appropriate for the attainment of a high level of task performance comparative to related methods.

1.3 Task Performance Evaluation

The evaluation criterion for individual and rover team task performance is the total *value of red rocks detected.* To avoid the problem where each rover evolves ANN controllers that maximize their own fitness function, yet the system as a whole achieves low values of the global fitness function, a difference evaluation function [1] is used (section 5.8).

1.4 Research Hypothesis

In a multi-rover task, which benefits from specialization, CONE derives a degree of specialization appropriate for achieving a higher task performance comparative to related methods. To test this hypothesis, the task performance and emergent specialization observed using the CONE and ESP methods for ANN controller design is compared.

1.5 Specialization

Specialization is measured at the ANN behavioral level, and defined according to the frequency with which a rover switches between executing distinct actions during its lifetime [7]. The lower the frequency of changing actions, the higher the degree of specialization. Given that a rover can execute v distinct actions, where one action can be executed per iteration in a lifetime of T iterations, the degree of specialization (S) is calculated as the frequency with which a rover switches between each of its v actions (f(v)) divided by T. That is, S is calculated as in equation 1.

$$S = 1 - \frac{f(v)}{T} \tag{1}$$

A value of S close to one indicates a high degree of specialization. That is, where a rover specializes to primarily perform one action, and switches between this and the other v-1 actions with a low frequency. A value of S close to zero indicates a low degree of specialization. That is, a rover is non-specialized, performing the v actions, and switching between these actions with a high frequency. For simplicity, if $S > 0.5$ then a rover is labeled *specialized*. If $S \leq 0.5$ then a rover is labeled *non-specialized*. The specialization label corresponds to the most executed action. The possible specialization labels are: *Long Range Detectors*, *Medium Range Detectors*, *Short Range Detectors*, or *Movers* (section 5.7).

2. METHODS

The *Enforced Sub-Populations* (ESP) and *Collective Neuro-Evolution* (CONE) methods use the same genotype encoding, crossover and mutation operators, as well as the same mechanism to adapt the number of sub-populations (ANN size) as a means of countering fitness stagnation.

Genotypes: A genotype is encoded as a string of 18 (14 input + 4 output weights) floating point values representing ANN weights connecting all sensory input neurons and all motor output neurons to a given hidden layer neuron.

Crossover and Mutation: A child genotype is produced using single point crossover [5], and *Burst* mutation with a *Cauchy* distribution [8]. Mutation of a random value in the range [-1.0, +1.0] is applied to each gene (connection weight) with a 0.05 degree of probability, and weights of each genotype are 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.

Adapting the Number of Sub-Populations: The sub-population number (ANN size) is adapted whenever the fitness progress of at least one of the fittest 20% of ANN controllers stagnate for 15 generations. The number of sub-populations in population i (the population from which a stagnating ANN is derived), is adapted via applying a neuron lesion mechanism [8]. The lesion mechanism evaluates the contribution of individual hidden layer neurons to overall ANN behavior, and accordingly increases or decreases the number of sub-populations.

3. ENFORCED SUB-POPULATIONS

ESP is a NE method for evolving a ANN controller that is appropriate for solving non-Markovian control tasks with sparse reinforcement such as double pole balancing [8]. ESP allocates and evolves a separate neuron population for each of the u hidden-layer neurons in an ANN. A neuron can only be recombined with other neurons from its own population. Multi-ESP creates n populations for deriving n ANN controllers. 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-ESP there is no recombination between corresponding sub-populations in different populations. Recombination only occurs *within* sub-populations. The example in figure 1 also illustrates a Multi-ESP setup using three populations for deriving three ANN controllers in a collective behavior task. However, the inter-population recombination depicted is not used in Multi-ESP. Multi-ESP is further described in related work [15].

4. COLLECTIVE NEURO-EVOLUTION

CONE is an extension of the ESP [8] and Multi-ESP [15] methods. CONE creates n genotype populations for n ANN controllers operating in a collective behavior task environment. Figure 1 exemplifies the CONE architecture with three populations for deriving three ANN controllers in a collective behavior task. Like related NE methods [8], CONE segregates the genotype space into n populations of genotypes. Each genotype population is further segregated into multiple sub-populations. From each sub-population, one hidden layer neuron is derived and one ANN is then constructed from the set of hidden layer neurons. The number of sub-populations within genotype population i equals the number of hidden layer neurons in an ANN that is derived from genotype population i. Accordingly, ANN-1 and ANN-2 (derived from genotype populations 1 and 2, respectively) consist of three hidden layer neurons, whilst ANN-3 (derived from genotype population 3) consists of four hidden layer neurons. A genotype represents a set of input and output connection weights of one hidden layer neuron. One gene represents one input and one output connection weight. Each gene is a floating point value within a given range.

4.1 GDM: Genotype Difference Metric

One contribution of the CONE method is the implementation of a self-regulating *Genotype Difference Metric* (GDM). The GDM regulates genotype recombination based upon genotype similarity. The GDM facilitates emergent specialized behavior via exploiting genotype similarities between ANN controllers, whilst avoiding deleterious offspring that result from recombining genotypes specialized to different functions. The GDM differs from similar metrics such as the compatibility threshold (δ_t), used by the NEAT method [13], that protect innovation and encourage diversity through speciation. GDM measures the difference between initially segregated populations, and regulates recombination between these populations based upon differences calculated for the fittest genotypes in each population. Where as, NEAT calculates the weight difference between a given genotype and other genotype ANN encodings in an initially unsegregated population. If less than δ_t, then the genotype is placed in an existing species, otherwise another species is created. δ_t

is dynamic, so given a target number of species, the system raises the threshold if there are too many species, and lowers the threshold is there are too few species.

The GDM calculates the *Genetic Distance* (GD) between two genotypes g_a and g_b. The GD is an average Euclidian based weight distance [14] between g_a and g_b, where g_a and g_b contain the same number of genes, and have the same minimum and maximum weight values. A normalized GD value is initialized within the range: [0.0, 1.0]. GDM regulated recombinations only occur between *corresponding* sub-populations sp_a and sp_b, in different populations (section 4.4). The GD between g_a and g_b is calculated as follows.

$$\delta(g_a, g_b) = \bar{W} \tag{2}$$

Where, $\bar{W}$ is the average weight difference between genes in g_a and g_b. That is:

$$\bar{W} = \sum_{i \epsilon N} \frac{|g_a(w_i) - g_b(w_i)|}{N} \tag{3}$$

Where, N is the size of any genotype g selected from a given sub-population, and w_i is the connection weight value of gene i. The GDM only operates on genotypes within the *elite portion* of sp_a and sp_b. The term *elite portion* refers to the fittest 50% of genotypes (m_h) in a given genotype population. The average weight difference between sub-populations sp_a and sp_b ($\bar{SP}$) is thus calculated as follows.

$$\bar{SP} = \sum_{i \epsilon m_h} \frac{\bar{W}}{m_h} \tag{4}$$

Where, m is the number of genotypes in sp_a and sp_b, and $m_h = \frac{m}{2}$.

4.2 Genotype Recombination and Mutation

Recombination occurs between two parent genotypes and results in the production of one offspring genotype. Prior to recombination, parent genotypes are split into two sets of genes. The first set represents the input connection weights of the encoded neuron. The second set represents the output connection weights of the encoded neuron. These corresponding sets of genes are then recombined using one-point crossover [5]. The two recombined sets of genes are then joined in order to form a complete offspring genotype. After genotype recombination, mutation is applied to each gene of each genotype with a predefined degree of probability. *Burst* mutation with a *Cauchy* distribution [8] is used in order to ensure that most weight changes are small whilst allowing for larger changes to some weights. Recombination operators are applied in the following cases.

1. All genotypes in all sub-populations have been tested and evaluated in a given number of task trials (that is, one generation of the CONE evolutionary process).

2. If the elite portions within corresponding sub- populations are within a given *GD* (section 4.1). In this case, the fittest 50% portions of all corresponding sub-populations that are within the given GD value are recombined. If more than two corresponding sub- populations are within the GD value of each other then sub- populations are randomly paired and recombination occurs between each pair of sub-populations. If there is a remainder sub-population after the random pairing, then recombination occurs *within* this remainder sub-population. If at least three sub-populations have the same *genetic* difference between each other, then the elite portions of two randomly selected sub-populations are recombined. Recombination then occurs *within* the elite portions of any leftover subpopulations.

If the elite portion of a given sub-population is not within the *GD* of any corresponding sub-population recombination occurs *within* the sub-population. Pairs of genotypes are randomly selected from within the elite portion of the sub-population. All pairs produce enough offspring genotypes to replace the current sub-population. One-point crossover was selected as a genotype recombination mechanism given its simplicity, general effectiveness as a genetic operator [5]. Two parent genotype recombination was selected given empirical evidence supporting the efficacy of two-parent recombination in a broad range of fitness landscapes [4], [3].

4.3 Sub-Population Recombination

The GD value is adapted in two successive stages. First, as a function of genetic relatedness between corresponding sub-populations and the fitness progress of each ANN controller. Second, as a function of behavioral relatedness between ANN controllers and the fitness progress of each ANN controller. In both cases, a heuristic for GD regulation either increases or decreases the value of GD by a portion of 1%. This did not change the GD value by too much per regulation, thus missing potentially useful GD values, and is also not too small so as to slow or inhibit the finding of an effective GD value.

4.3.1 Recombinations for Regulating Recombination

Two heuristics are applied for regulating the GD value as a function of genotype recombinations and fitness progress.

1. If the number of recombinations between corresponding sub-populations is increasing over the previous V + W generations, and fitness stagnates or is decreasing over this same period, then decrement the GD value.

2. If the number of recombinations between corresponding sub-populations is decreasing or stagnates over the last V + W generations, and fitness stagnates or is decreasing over this same period, then increment GD.

4.4 Corresponding Subpopulations

For recombination that occurs *between* sub- populations, sub-populations must be *corresponding.* Corresponding sub-populations are situated in different genotype populations, where there are n genotype populations. All genotypes within any given sub-population have the same tag. A genotype's tag specifies what position a neuron (decoded genotype) assumes in the hidden layer of an ANN. That is, sub-population j within populations i and k, where $i \neq k$ and $k \leq n$, are *corresponding.* This is the case if genotypes contained within sub-population j (of populations i and k) represent neurons assigned to the same hidden layer position of ANN controllers i and k.

4.5 CONE Process

1. *Initialization.* n genotype populations are initialized with u sub-populations. Each sub-population contains m genotypes. Sub-population j contains genotypes corresponding to neurons assigned to position j in the hidden layer of an ANN derived from genotype population i, where $j \leq u$, and $i \leq n$.

2. *ANN Testing.* ANN controllers are derived and tested by systematically selecting each neuron from each of the u sub-populations (within a given genotype population) and testing it with u-1 other neurons (in the context of a complete ANN). The other neurons are randomly selected from the other u-1 sub-populations, such that no neuron is selected more than once. This same process is repeated n-1 times for deriving another n-1 ANN controllers from the other n-1 genotype populations, such that n ANN controllers are collectively tested and evaluated in the task environment. Each ANN is tested for a *lifetime* of q epochs, where $q \geq m * n$. An epoch is a task trial that is executed for w simulation iterations. Each epoch tests different task and environment dependent conditions.

3. *Evaluation.* After each epoch, each ANN is assigned a fitness value *f*, which is also assigned to each of the neurons participating in the hidden layer of an evaluated ANN. The fitness assigned to a neuron is cumulative with the number of epochs that the neuron participates in. An *average neuron fitness* is calculated as a neuron's accumulated fitness divided by the number of epochs the neuron participates in.

4. *Processing.* The *testing* and *evaluation* of neurons continues until all neurons within all sub-populations (within all genotype populations) have been tested at least r times and assigned a fitness. As with SANE [9] the testing and evaluation of each neuron r times constitutes one generation.

5. *Recombination.* Each generation, all genotypes are ranked by fitness, and recombination occurs *within* and *between* sub-populations (section 4.2). A sufficient number of genotype offspring are produced in order to replace current sub-populations.

6. *Mutation.* At the end of each generation, a mutation operator (section 4.2) is applied with a given probability to each gene of each genotype.

7. *Check for fitness stagnation.* If fitness within at least population (ANN controller) has not progressed in V generations, then adapt the GD (section 4.3).

8. Reiterate steps [2, 7] until a desired collective behavior task performance is achieved.

5. EXTENDED MULTI-ROVER TASK

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 (section 5.8). The complexity of the multi-rover task described in [1] is extended to include multiple

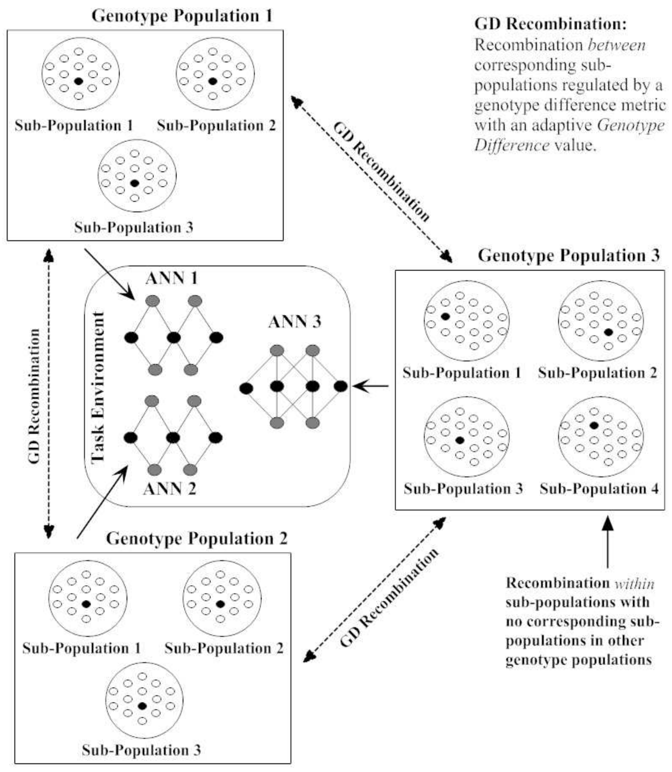

Figure 1: An example of *Collective Neuro-Evolution.*

rover actions (section 5.7). Each action yields a cost and benefit tradeoff, giving each rover the possibility of exploiting specialization to a given action as a means of increasing its own task performance. Related research in the extended multi-rover task [10], has highlighted that behavioral specialization increases task performance at both the individual and rover team level. Furthermore, these case studies demonstrated that applying a systematic search method for controlling a rover team 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, sensor and actuator capabilities.

Table 1 presents rover and environment parameter settings. Rover parameter settings are those used by individual rovers, 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	
Rover lifetime	2500 iterations
Rover movement range	0.01
Rover movement cost	1.0
Red rock detection sensor range	Variable
Red rock detection sensor cost	Variable
Red rock detection sensors accuracy	Variable
Rover detection sensor range	0.02
Rover detection sensors cost	0.05
Rover detection sensors accuracy	1.0
Rover initial energy	1000 units
Initial rover positions	Random
Environment width	1.0
Environment length	1.0
Individual red rock value	[1, 10]
Total red rock value in environment	40000
Red rock value distribution	2D Gaussian mixture model

Table 1: Multi-rover simulation parameters.

5.1 Collective Red Rock Detection

Detection of red rock value requires the successful detection of a red rock by at least two rovers during the same simulation iteration. The probability that a rover successfully detects a red rock is determined by a rover's current detection sensor setting (table 2). If at least two rovers concurrently and successfully detect a red rock, it is marked as *detected* and its value recorded. The red rock is then removed from the simulation environment so as it is not detected again.

5.2 Continuous Simulation Environment

The simulation environment is a two dimensional continuous plane, where one rover can occupy any given *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 observation distance δ^2_{min}[1]. Equation 5 specifies this metric.

$$\delta(p, q) = min\|x - y\|^2, \delta^2_{min} \tag{5}$$

5.3 Red Rock Distribution

The simulation environment is populated with 40000 red rocks, distributed according to a two dimensional *Gaussian mixture model* [12]. 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 ten radii ρ = [0.50, 0.45, 0.40, 0.35, 0.30, 0.25, 0.20, 0.15, 0.10, 0.05] are tested. Distributions range from approximately uniform (*environment type 1*) through to clustered (*environment type 10*), respectively. The radii values are percentages of environment size, where the environment width and height are equal. Each red rock has an integer value equal to one, where the sum of all values equals 40000. Red rock locations are random within a given distribution and thus initially unknown to rovers.

[1]A minimum distance prevents singularities [1] when a rover is very close to a red rock.

Red Rock Detector Sensor Setting	**Accuracy**	**Range**	**Cost**
Minimum (Setting 1)	1.0	0.02	0.25
Medium (Setting 2)	0.5	0.03	0.5
Maximum (Setting 3)	0.25	0.05	1.0

Table 2: Red rock detection sensor settings.

5.4 Red Rock Detection Sensors

Each red rock detection sensor covers one quadrant in a rover's sensory *Field Of View* (FOV). Eight sensors provide a 360 degree FOV. Red rock detection sensors need to be explicitly activated with one of three settings. This constitutes one action. Detection sensor settings are: *low*, *medium*, and *high*, where settings determine the sensors' range, accuracy, and cost of usage (table 2). *Accuracy* denotes the degree of probability with which red rocks are successfully detected. *Range* is defined as a portion of the width of the simulation environment, where width and length are equal. *Cost* is the energy used each time the red rock detection sensors are activated. Red rock detection sensor *q* returns the sum of red rock values in sensor quadrant *q*, divided by the squared distance to the rover. Detection sensor values are normalized within the range [0.0, 1.0]. That is:

$$S_{1,q,t} = \sum_{q \epsilon J_q} \frac{1}{\delta(L_{q'} L_{q,t})} \tag{6}$$

Where, J_q is the set of red rock values in sensor quadrant *q* (up to sensor range). L_q is the location of red rock *q*, and $L_{\eta,t}$ is the location of rover η at time *t*. Detection sensor values are normalized within the range [0.0, 1.0]. Red rock detection sensor *q* returns the value of red rocks in sensor quadrant *q* ($\Re = [rv_0, rv_k]$, where there is a maximum of *k* red rocks within each sensor quadrant).

5.5 Rover Detection Sensors

The purpose of rover detection sensors is to prevent collisions between rovers. Eight rover detection sensors covering a 360 degree FOV are constantly active and have a fixed accuracy, range and cost (table 2). Rover detection sensor *q* returns the sum of rovers in sensor quadrant *q*, divided by the squared distance to *this* rover. Rover detection sensor values are normalized within the range [0.0, 1.0]. That is:

$$S_{2,q,i,t} = \sum_{i' \epsilon N_q} \frac{1}{\delta(L_{i'} L_{i,t})} \tag{7}$$

Where, N_q is the set of rovers in sensor quadrant *q*.

5.6 Movement Actuators

A rover's heading is calculated from two motor output values in the ANN controller (MO-3 and MO-4 in figure 2). These output values determine the vectors *dx* and *dy*. 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: $dx = d_{max}(o_1$ - 0.5), and $dy = d_{max}(o_2$ - 0.5). Where, d_{max} is the maximum distance a rover can move in one simulation iteration, o_1 and o_2 are the values of motor outputs MO-3 and MO-4.

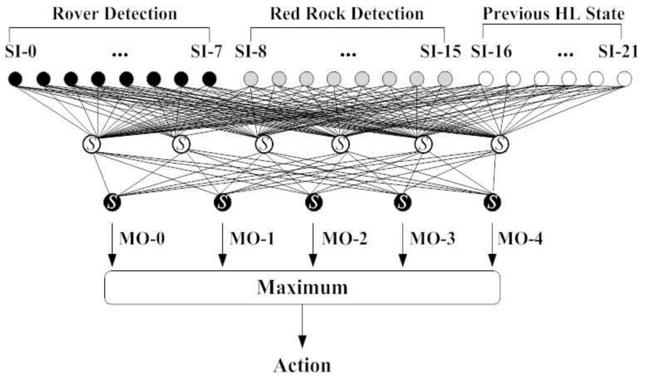

Figure 2: Rover ANN controller. HL: Hidden Layer, SI: Sensory Input, MO: Motor Output.

5.7 Artificial Neural Network Controller

A rover uses a recurrent ANN in order to map sensory inputs to motor outputs. Figure 2 presents 22 sensory input neurons ([SI-0, SI-21]) as being fully connected to six hidden layer neurons. Input neurons [SI-0, SI-7] accept input from each of eight rover detection sensors. Input neurons [SI-8, SI-15] accept input from each of eight red rock detection sensors. Input neurons [SI-16, SI-21] accept hidden layer neurons activation values from the previous simulation iteration. Motor output neurons ([MO-0, MO-4]) are fully connected to six hidden layer neurons. Hidden and motor output neurons are sigmoidal neurons. Motor output values are normalized in the range: [0, 1].

5.7.1 Action Selection

Each time step, the motor output with the highest value is the action executed. The four rover actions are as follows.

1. Activate red rock detection sensors with *setting 1* (MO-0 in figure 2).

2. Activate red rock detection sensors with *setting 2* (MO-1 in figure 2).

3. Activate red rock detection sensors with *setting 3* (MO-2 in figure 2).

4. Move in direction calculated from dx and dy ([MO-3, MO-4] in figure 2).

5.8 Rover Fitness Functions

A global fitness function, $G(z)$, is defined as a function of the total *red rock value detected* by all rovers in a team (section 5.8.1). The goal of a rover team is to maximize $G(z)$. However, rovers do not maximize $G(z)$ directly. Instead each rover η attempts to maximize its own private fitness function $g_\eta(z)$. It is important to note that $G(z)$ does not guide evolution, but rather provides a measure of rover team performance, based upon the contributions of individual rovers. The private rover fitness function guides the evolution of each rover's ANN controller (section 5.8.2).

5.8.1 Global fitness evaluation

$G(z)$ calculates the sum of the *value of red rocks detected* (V), for all n rovers. For any given experiment, an average V is calculated over all epochs of all rover lifetime's. The highest V is then selected from each rover's lifetime in order to calculate G. G is defined in equation 8.

$$G = \sum_n \sum_t \sum_i \frac{V_{i,\eta,t}}{min_\eta \delta(L_i, L_{\eta,t})} \tag{8}$$

Where, $V_{i,\eta,t}$ is the value of red rock i detected by rover η at simulation time t. L_i is the location of red rock i, and $L_{\eta,t}$ is the location of rover η at simulation time t. Where, $t \leq T$, and T is equal to the number of simulation iterations in any given rover's lifetime. $V_{i,\eta,t}$ is an indicator function, and returns the value of red rock i if and only if, at simulation time t, η is a rover where $\delta(L_i, L_{\eta,t}) \leq$ *red rock detection sensor range* (section 5.5), and a *second rover* is also successfully detecting red rock i (section 5.1).

5.8.2 Private fitness evaluation

The private fitness function $g_\eta(z)$ calculates the *value of red rocks detected* (V) by a given rover over the course of the rover's lifetime. An average V is calculated over all epochs of the rover's lifetime. The highest V is then selected in order to calculate g_η. g_η is defined in equation 9.

$$g_\eta = \sum_t \sum_i \frac{V_{i,\eta,t}}{min \delta(L_i, L_{\eta,t})} \tag{9}$$

Where, g_η is equivalent to the global fitness function when there is only one rover.

6. EXPERIMENTS

Experiments apply ESP and CONE for evolving ANN controllers in teams of 20 rovers in ten test environments (section 5.3). Each experiment executes a method for 20 simulation runs. Each simulation consists of 500 generations. A generation corresponds to the *lifetime* of each rover in the team. A rover lifetime lasts for 50 epochs, where each epoch consists of 2500 simulation iterations. An epoch represents a task scenario that tests different rover starting positions and orientations in the simulation environment. An *elite portion* is the fittest 20% of a genotype population. Results in figure 3 are averages calculated over 20 runs.

6.1 ESP

Six sub-populations are initialized, where each sub-population contains 2000 genotypes. A homogenous rover team is created via randomly selecting one genotype from the elite portion of each sub-population. These six genotypes are then decoded into neurons which form the hidden layer of an ANN rover controller. The ANN is copied 20 times in order to form a team of rover clones.

Fittest ESP Evolved Team Compositions						
N-S	**Movers**	**SRD**	**MRD**	**LRD**	**S**	**Fitness**
29%	16%	12%	17%	26%	0.45	4303
Fittest Multi-ESP Evolved Team Compositions						
N-S	**Movers**	**SRD**	**MRD**	**LRD**	**S**	**Fitness**
3%	24%	41%	8%	24%	0.76	4670
Fittest CONE Evolved Team Compositions						
N-S	**Movers**	**SRD**	**MRD**	**LRD**	**S**	**Fitness**
8%	17%	27%	20%	28%	0.91	5565

Table 3: Fittest ESP, Multi-ESP and CONE evolved teams. *Fitness*: Red rock value *detected*. *S*: Maximum degree of specialization measured for this team. *N-S*: Non-Specialized. *SRD*: Short Range Detectors. *MRD*: Medium Range Detectors. *LRD*: Long Range Detectors.

Lesioned Sub-Group	CONE	Multi-ESP	ESP
Movers	31.90%	32.87%	35.95%
Short Range Detectors	40.34%	45.98%	55.81%
Medium Range Detectors	35.75%	40.12%	62.02%
Long Range Detectors	32.56%	39.10%	53.92%

Table 4: Rover team performance when specialized sub-groups are systematically lesioned.

6.2 Multi-ESP/CONE

One genotype population is initialized for each of the 20 rovers, where each population contains six sub-populations, and each sub-population contains 100 genotypes. A heterogenous rover team is created via deriving one ANN controller from each genotype population. That is, for each of the 20 populations, one genotype is randomly selected from the elite portion of each sub-population. These six genotypes are then decoded into hidden layer neurons which form the hidden layer of an ANN rover controller.

6.3 Evolved Team Compositions

Table 3 presents the team compositions of individually specialized rovers derived by the highest performing ESP, Multi-ESP and CONE evolved teams. Specifically, these teams consist of *specialized* sub-groups, where sub-groups are defined according to the number of individual specializations of a given type (section 1). That is, either: *Short-Range Detectors*, *Medium-Range Detectors*, *Long-Range Detectors*, *Movers*, or non-specialized.

6.4 Method Comparisons

In order to draw conclusions from this comparative study, a set of statistical tests are performed in order to gauge respective task performance differences between rover teams evolved with the ESP, Multi-ESP, and CONE methods.

6.4.1 Statistical Tests

The Kolmogorov-Smirnov test [6] confirms that the data sets representing results of Multi-ESP, ESP and CONE evolved teams (figure 3) for all environment types, conform to normal distributions. That is, P = [0.64, 0.51, 0.85], [0.9, 0.99, 0.58], [0.54, 0.89, 0.84], [0.97, 0.77, 0.86], [0.73, 0.37, 0.75], [0.61, 0.99, 0.39], [0.61, 1.0, 0.98], [0.84, 0.95, 0.98], [0.43, 0.91, 0.80], [0.84, 0.57, 0.86], respectively. In order to determine if there is a statistical significance of difference between data presented in figure 3, an independent t-test [6] is applied. A statistical significance of 0.05 is selected, and the null hypothesis is stated as the data sets not significantly differing. For each environment type, the t-test is applied to results of ESP and CONE, ESP and Multi-ESP, and finally to Multi-ESP and CONE evolved teams. For the ESP and CONE comparison, P = [0.97, 0.077, 0.18, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001], respectively. For the ESP and Multi-ESP comparison, P = [0.58, 0.064, 0.45, 0.0001, 0.0007, 0.0001, 0.0001, 0.0021, 0.0001, 0.0083], respectively. For the Multi-ESP and CONE comparison, P = [0.51, 0.77, 0.92, 0.0012, 0.0001, 0.0002, 0.0028, 0.0001, 0.0001, 0.0001], respectively. This result indicates that, except for environment types [1, 3] there is a significant difference between ESP, Multi-ESP, and CONE evolved teams. This result supports the hypothesis that CONE, on average, evolves teams with a higher level of task performance comparative to related methods (section 1.2).

6.4.2 Lesion Study

A lesion study conducted at the ANN controller sub-group level supports the hypothesis that CONE evolves a degree of behavioral specialization (S in table 3) appropriate for achieving a comparatively higher task performance. Specifically, each specialized sub-group of rovers is removed from the fittest CONE evolved team and the team executed again over 20 experimental runs. The removed sub-group is replaced with another sub-group of rovers using heuristic controllers with *hard-wired* non-specialized behaviors [11]. This same sub-group lesioning procedure is repeated for each of the specialized sub-groups in the fittest ESP and Multi-ESP teams. Table 4 presents the re-evaluated performances of the fittest ESP, Multi-ESP and CONE evolved teams when each of the specialized sub-groups are systematically removed and replaced with non-specialized sub-groups. The values in table 4 are percentages of original performances, and indicate that, on average, the fittest CONE evolved team is more dependent upon the specialized sub-groups for accomplishing a high task performance, where as the fittest ESP and Multi-ESP teams are less dependent. The degree of specialization derived for the fittest CONE evolved team is theorized to facilitate the comparatively higher task performance. However, this is subject to further study.

6.4.3 Genotype Difference Metric Analysis

The CONE method is re-applied for evolving rover teams in a new set of multi-rover experiments. In this set, the GDM for regulating genotype recombination between populations was not active. In this case, genotype recombination only occurs *within* sub-populations of any given population. Average red rock detection results calculated over 20 experimental runs are compared with previous results yielded by ESP and Multi-ESP evolved teams. Table 5 presents results from the re-execution of teams evolved using CONE (without the GDM) for environment types [1, 10]. The third and fourth columns present the results previously attained by ESP and Multi-ESP evolved teams. These results are also

Average Evolved Rover Team Performance			
Environment Type	CONE: GDM Off	ESP	Multi-ESP
1	4403	4354	4489
2	4346	4441	4371
3	4192	3969	4215
4	4035	3409	4075
5	4129	2879	4150
6	3394	2738	3390
7	3476	2716	3459
8	3189	2584	3220
9	3284	2602	3297
10	3090	2514	3070

Table 5: Performance of CONE evolved rover teams re-executed *without* the *Genotype Difference Metric* (GDM).

presented in figure 3. A statistical comparison[2] of respective method results in table 5 elucidated that teams evolved by CONE without the GDM, yielded a task performance comparable to Multi-ESP evolved teams. However, these CONE evolved teams yield a higher task performance comparable to ESP evolved teams. These results indicate that the GDM for regulating genotype between populations is beneficial to the task performance of CONE evolved rover teams.

7. CONCLUSIONS

This research compared the task performance of simulated rover teams operating with ANN controllers and evolved with the ESP and CONE methods in a collective behavior search and find task. Results indicate that CONE derives a degree of emergent specialization that is appropriate for deriving a higher collective behavior task performance, comparative to ESP and Multi-ESP evolved teams.

8. 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] A. Eiben. Multiparent recombination. In T. B. D. Fogel and Z. Michalewicz, editors, *Evolutionary Computation 1: Basic Algorithms and Operators*, pages 289–307. Institute of Physics Publishing, Ottawa, Canada, 2000.

[4] A. Eiben and T. Baeck. An empirical investigation of multi-parent recombination operators in evolution strategies. *Evolutionary Computation.*, 5(3):347–365, 1997.

[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.

[2]T-test P values are not presented due to space constraints.

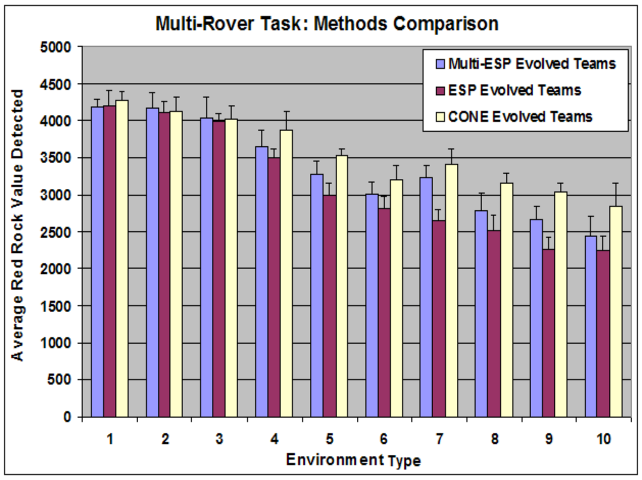

Figure 3: Average red rock values detected by ESP, Multi-ESP, and CONE evolved rover teams, for approximately uniform (environment type 1) through to clustered (environment type 10) red rock distributions.

[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] D. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22(1):11–33, 1996.

[10] 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.

[11] G. Nitschke, M. Schut, and A. Eiben. Emergent specialization in the extended multi-rover problem. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 100–106, Singapore, 2007. IEEE.

[12] 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.

[13] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation.*, 10(2):99–127, 2002.

[14] M. Wineberg and F. Oppacher. The underlying similarity of diversity measures used in evolutionary computation. In *Proceedings of the Fifth Genetic and Evolutionary Computation Conference*, pages 1493–1504, Berlin, 2003. Springer.

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