Collective Specialization for Evolutionary Design of a Multi-Robot System

A.E. Eiben, G.S. Nitschke, and M.C. Schut

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

Abstract. This research is positioned in the context of controller design for (simulated) multi-robot applications. Inspired by research in survey and exploration of unknown environments where a multi-robot system is to discover features of interest given strict time and energy constraints, we defined an abstract task domain with adaptable features of interest. Additionally, we parameterized the behavioral features of the robots, so that we could classify behavioral specialization in the space of these parameters. This allowed systematic experimentation over a range of task instances and types of specialization in order to investigate the advantage of specialization. These experiments also delivered a novel neuro-evolution approach to controller design, called the collective specialization method. Results elucidated that this method derived multi-robot system controllers that outperformed a high performance heuristic and conventional neuro-evolution method.

1 Introduction

Biological social systems have long been a source of inspiration to engineers. In particular, research in multi-robot and artificial life collective behavior systems, has often attempted to replicate the success of social insect societies at decomposing the labor of a group into composite specialized and complementary roles so as to accomplish collectively, global goals that could not otherwise be accomplished by individual insects. Mechanisms and design principles that facilitate emergent behavioral specialization have been studied in biological [7], artificial life [1], and multi-robot systems [10] research. However, collective behavior design methods for harnessing and utilizing emergent specialization for the benefit for problem solving are lacking in current swarm engineering approaches.

This paper describes a comparative study testing *Neuro-Evolution* (NE) and heuristic methods with respect to the role of specialization in solving a collective behavior task[1]. NE is an approach that combines techniques native to both neural networks and evolutionary computation research. Both of these techniques

[1] Terms used herein are defined as follows: *task*: what has to be done, *activity*: what is being done, *role*: the task assigned to a specific individual within a set of responsibilities given to a group of individuals, *caste*: a group of individuals specialized in the same role [10].

have historically been successful in addressing single agent control problems [6] and have recently had some success for controller design in collective behavior research [2]. The advantages of applying NE to collective behavior research have been illustrated in a variety of applications including multi-agent computer games [2], RoboCup [17], and robot controller design [1]. Such applications have highlighted that NE is most appropriately applied to complex problems that are neither effectively addressed via pure evolutionary computation methods or neural processing approaches.

It has been suggested that autonomous robotic explorers whose behavioral or morphological design (or both) is biologically inspired could be feasible and cost-effective in future planetary exploration [16], as well as providing an alternative to traditional, labor-intensive, tele-robotic operations [18]. The challenge addressed in this paper concerned developing controllers for a group of simulated Unmanned Autonomous Vehicles (UAV's) given a *search and find task* constrained by limited resources.

Environmental factors such as resource distribution greatly influence social organization in biological [7], and artificial social systems [10], given specific types of tasks such as collective foraging. In this paper, heuristic methods highlighted that specialization was beneficial in a search and find task, given specific types of resource distribution in the environment. For this task, we defined what we termed the *Collective NE* (CONE) method that was successful in deriving a *caste*[1] that outperformed a heuristic and conventional NE method.

1.1 Research Goal

The research goal was to demonstrate a NE method capable of deriving specialization for increasing task performance in environments where specialization was beneficial.

1.2 Specialization

Specialization was defined at the *agent* level (1 aerial explorer) and the *group* level (n aerial explorers). An agent was specialized if *more than 50%* of its lifetime was dedicated to one role. We defined a caste where *more than 50%* of the group members assumed one role for *more than 50%* of their respective lifetimes.

1.3 First Hypothesis

There exist particular types of task environments where specialization increases task performance.

To support our first hypothesis, the value of specialization in particular types of task environments was demonstrated using a heuristic method that tested the task performance of pre-defined *castes*.

1.4 Second Hypothesis

That the collective specialization NE method is appropriate for deriving specialized groups (that is: *castes*) with high task performance.

To support our second hypothesis of specialization being a requisite for increased task performance, the performance of the collective specialization method (where we supposed that emergent specialization would be observed to benefit performance), was compared to that of a conventional NE method.

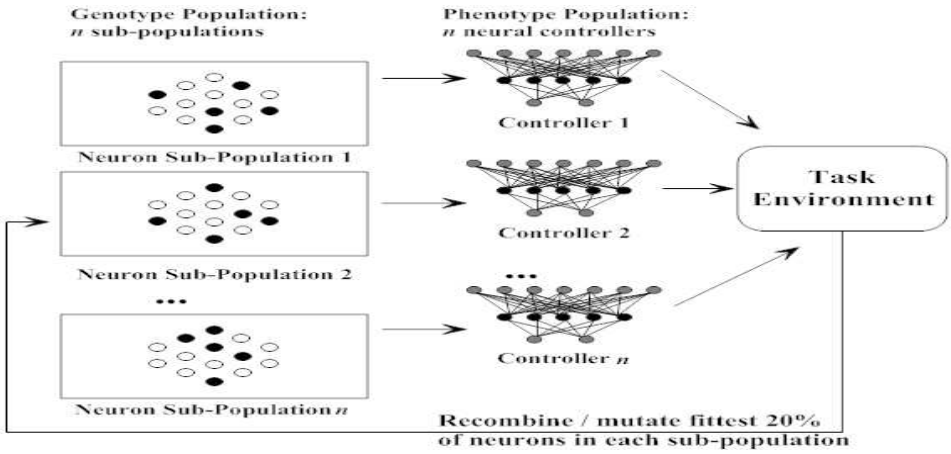

Fig. 1. CONE: Collective Neuro-Evolution. See section 2 for details.

2 CONE: Collective Neuro-Evolution

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

1. n agents (neural controllers) were constructed via selecting p genotypes (neurons) from each sub-population of genotypes. These p neurons then became the hidden layer of each of the n controllers, which were subsequently placed in the task environment. The group of controllers was thus heterogeneous, given that each was constructed via selecting a set of p hidden layer neurons from each of the n sub-populations. Evolutionary operators were not applied between the n sub-populations.
2. The n controllers were tested together in the task environment for a *lifetime* of q epochs, where an epoch was a test scenario lasting for w iterations of simulation time. Each epoch tested different task dependent agent and environment conditions, such as agent starting positions and locations of resources in the environment. For each of the q epochs (where $q \geq m$ genotypes

in a sub-population), each genotype in a given sub-population was selected and tested in combination with p-1 other neurons (thus forming a controller) randomly selected from the same sub-population.

3. Thus p neurons from each of the n sub-populations would concurrently be evaluated in the task environments and assigned a fitness. Testing of neurons within each sub-population would continue until all neurons had been tested at least once.
4. At the end of an agents lifetime (q epochs) a fitness value was assigned to each set of p neurons that participated in each of the controllers. The assigned fitness of each set of p neurons was calculated as the average of fitness values attained over all epochs of an agents lifetime.
5. For each sub-population, recombination and mutation of the fittest 20% of genotypes then occurred, where the fittest 20% were arranged into pairs of genotypes, and each pair produced 5 child genotypes, so as to propagate the next generation of each sub-population.
6. A single genotype was randomly selected from the fittest 20% of the newly recombined genotypes within each of the n sub-populations. These n selected genotypes were then decoded into controllers, placed in the task environment, and executed as the next generation. This process was then repeated for r (table 1) generations.

2.1 Online versus Offline adaptation in the CONE method

Most NE methods were originally designed to run offline, meaning that all genotypes in a population were successively tested and evaluated, and after the whole population had been tested, evolutionary operators were applied in order to create the next population. The fittest genotypes of any given population could then be selected as those best suited to solving the given task. Recently there has been some success in applying NE methods for online adaptation in certain collective behavior tasks such as multi-agent computer games [14].

As with the other NE methods applied to collective behavior tasks, such as NEAT [15] and rtNEAT [14], the evolutionary cycle of selection and replacement operated continually as controllers interacted with their task environment, effectuating the emergence of new controllers in response to dynamic challenges in the task environment. Dissimilar to other online NE methods, the CONE method derived a new controller (phenotype) from each of n sub-populations of genotypes at the turn of each generation of artificial evolution, where a separate evolutionary process of selection and replacement operated within each of the n sub-populations. Hence, as the n controllers worked to accomplish their task, each sub-population was progressively (and not necessarily synchronously) updated, such that the *ith* controller, where $i \varepsilon n$, was in turn updated (decoded) from the *fittest* genotype from the *ith* sub-population.

3 Conventional Neuro-Evolution

The conventional NE method was adapted from that used for previous evolutionary robotics experiments [12], and as illustrated in figure 2 used only a single population of genotypes. After, randomly initializing a population of m genotypes, the conventional NE process operated as follows.

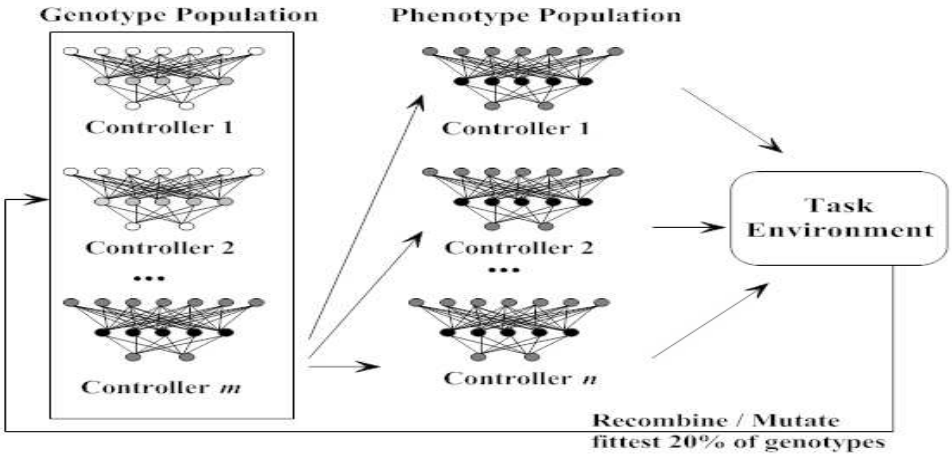

Fig. 2. Conventional neuro-evolution. See section 3 for details.

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

8. This process was repeated for the r generations that the conventional NE method was executed for (table 1).

4 Genotypes

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

4.1 Recombination of genotypes: Crossover and Mutation

Each child genotype was produced using single point crossover [4], and *Burst* mutation with a *Cauchy* distribution [8]. As illustrated in table 1 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.

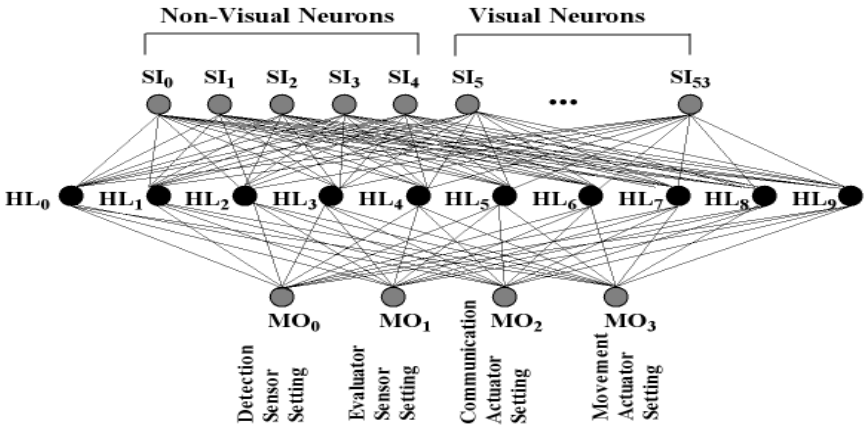

Fig. 3. Adaptive topology neural controller. See section 7.2 for explanation.

4.2 Fitness Calculation

At the end of each generation (section 2) a fitness value was assigned to each of the n controllers, where each of the neurons participating in each controller was assigned an equal portion of the fitness value. These individual neuron fitness values were then assigned back to the sub-population corresponding to each of the controllers. Although this fitness estimation method, known as *fitness*

sharing [3] was convenient for deriving the contribution of each neuron to a controller, it was problematic in that it potentially prevented the selection of the best neurons across successive generations. However, this was offset by the advantage that there was no disparity between controller fitness and the fitness of individual neurons.

5 Phenotypes: Constructing controllers from neurons

An agent phenotype (feed-forward neural controller) was constructed from a set of 10 genotypes (hidden layer neurons). Given that the CONE method operated at the neuron (not the controller [11]) 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 [15], 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.

Neuro-Evolution Parameter Settings	
Runs per NE method	20
Generations	500
Epochs	50
Iterations / Epoch	1000
Mutation probability	0.05
Mutation range	[-1.0, +1.0]
Weight range	[-10.0, +10.0]
Initial Weights	Random
Crossover	single point
Hidden neurons	10
Phenotypes	100 Controllers
Genotype length	18 (14 + 4) weights
Genotypes	100 (Conventional NE) / 10000 (CONE)

Table 1. Neuro-evolution parameter settings.

5.1 Dynamic topologies

Many NE methods have used a process known as *complexification* which changes the topology of a neural controller, and thus its corresponding genotype length, as part of an evolutionary process [9]. However, only a few of such methods have been successfully applied to collective behavior tasks [14].

The CONE method also uses complexification to dynamically change genotype lengths as part of the evolutionary process. Where as, [14] used *synapsis* to recombine genotypes of different lengths, this was not necessary in CONE, as all genotypes within a given sub-population were kept the same length, and recombination of genotypes from different sub-populations did not occur. Also, the CONE method only changed the number of sensory input neurons, and the number of hidden layer and motor output neurons were kept static.

6 Collective Survey Task

Inspired by UAV survey and exploration missions of unknown environments [16], a group of 101 simulated UAV's (100 *explorer* agents and 1 *lander* agent) were given the task of maximizing the number of features of interest (herein termed: *red rocks*) discovered within a *survey area* of an environment given limited sensor and actuator capabilities, battery power and mission time. Red rock locations and distributions were initially unknown to the agents. The lander had no active role in the discovery of red rocks. Its role was to act as a base station that received red rock data (locations and value of features of interest) communicated to it, and to recharge explorer agents that successfully accomplished their task.

6.1 Environment

The simulation environment[2] was represented as a discrete three dimensional environment of 200 x 200 x 200 voxels.

Adaptive Input Layer Topology of Neural Controller			
Detection Sensor Setting	Genotype Length	Visual Neurons	Total Input Neurons
3 ($0.75 < \text{MOV1} \leq 1.0$)	**58**	**49** (FOV=49); PODS=0.99	**54**
2 ($0.5 < \text{MOV1} \leq 0.75$)	**34**	**25** (FOV=25); PODS=0.89	**30**
1 ($0.25 < \text{MOV1} \leq 0.5$)	**18**	**9** (FOV=9); PODS=0.79	**14**
0 ($0.0 < \text{MOV1} \leq 0.25$)	**10**	**1** (FOV=1); PODS=0.69	**6**

Table 2. Adaptive controller topology. MOV: Motor Output Value 1 (MO_0 in figure 3). FOV: Field of View. PODS: Probability Of Detection Success. See section 7.2 for explanation.

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

6.2 Red Rocks (Features of Interest).

As described by the red rock discovery algorithm (section 8.2), when an agent had discovered a red rock using its *detection sensor* the red rock location was moved to, using the *movement actuator*, red rock value (evaluation data) was then ascertained, using the *evaluation sensor*. Evaluation data was communicated from the agent to the lander, using the *communication actuator* and the red rock value communicated back to the agent from the lander. Red rocks either had a value (1) or not (0). A red rock with value was marked as *evaluated*, so it would not again be subject to evaluation. Thus the number of resources in the environment diminished with successful task accomplishment. Agents that had evaluated red rocks with value would be marked as being eligible for an energy reward, and also immediately receive a fitness reward (section 4.2) equal to the value of the last red rock evaluated.

Red rock value served two purposes. First, it provided a performance measure for the agent group. Second, it was translated into energy and fitness rewards. When an agents energy depleted to below 500 units, it would return to the lander and the total red rock value that the agent had gathered thus far would be translated directly into an energy reward.

6.3 Red Rock Distribution.

A simulation consisted of 40000 red rocks distributed over the base of the environment. That is, a red rock could be placed at each possible x_i, y_i, z_j, where $0 \leq i < 200$, j=0. We described red rock distribution (degree of structure) in the environment using a two dimensional *Gaussian mixture model* [13]. The mixture model was specified with 4 centroids, where the radius of each determined the spatial distribution of red rock around each. 10 radii were tested, such that red rock distributions generated ranged from a uniform (such an environment was termed as having a *low degree of structure*) through to a clustered distribution (such an environment was termed as having a *high degree of structure*). We labeled these environment types from *0* (low degree of structure) through to *9* (high degree of structure). All 10 environment types were tested using the heuristic (section 8), CONE (section 2) and conventional neuro-evolution methods (section 3).

Agent Type	**Detect**	**Evaluate**	**Move**	**Communicate**
Detector	0.6	0.1	0.2	0.1
Evaluator	0.1	0.6	0.2	0.1
Communicator	0.1	0.1	0.2	0.6

Table 3. Heuristic Method: Specialized agent types and their probabilistic preferences for action selection.

7 Agents

At simulation initialization, each aerial explorer was placed in a random voxel (x_i, y_i, z_i, where, $0 \leq i < 200$). A maximum of 4 aerial explorers could occupy a given voxel.

7.1 Morphology: Sensors and Actuators

Agent morphology was defined in terms of 1 *detection sensor*, 1 *evaluation sensor*, 1 *movement actuator*, and 1 *communication actuator*. This selection of sensors and actuators was based upon design proposals for rotorcraft that are to operate as autonomous aerial explorers [18]. Rotorcraft are considered advantageous given their vertical lift capabilities, allowing them to *detect* red rocks in flight using a directional visual sensor, perform some preliminary categorization of red rocks, *move* in order to *evaluate* selected red rocks using a physical contact sensor, and then *communicate* red rock data. Rotorcraft are also able to land to recharge at a base station.

7.2 Controllers: Adaptive Topology Neural Network

Figure 3 illustrates the feed forward neural controller used by the explorer agents. The controller connected all sensory input nodes to 10 hidden layer nodes, (HL_0..HL_9) to 4 motor output nodes (MO_0..MO_3). The number of non-visual, hidden-layer and output nodes remained static at 5, 10, and 4 respectively.

Sensory inputs: Non-visual Non-visual input nodes (SI_0..SI_4) took as input the 4 motor output (MO_0-MO_3) values, and the red rock evaluation value from the previous simulation iteration, respectively. These previous values were teaching inputs [12] which influenced the next motor outputs.

Sensory inputs: Visual Figure 3 also illustrates that the number of visual neurons in the sensory input layer was dynamic within the range SI_5 (one voxel viewable) and SI_{53} (49 voxels viewable). All values taken by sensory input nodes were normalized.

The number of sensory input neurons determined the accuracy of sensor readings for detecting (table 2) features of interest (red rocks) in the environment. In this case, more sensory input neurons indicated that more discrete locations in the environment (voxels) could be observed with the directional red rock detection sensor (table 6).

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

Genotype Representation A single genotype was encoded as a string in the interval of [10, 58] connection weights. The genotype to phenotype (hidden layer neuron) mapping scheme was a direct one-to-one mapping, where each

connection weight corresponded to a floating-point number in the interval [-10, +10]. A controller was constructed from a set of 10 genotypes (encoded hidden layer neurons). So as to simplify assembly of a neural controller, all genotypes within a sub-population (CONE method), or population (conventional NE method) of genotypes were kept the same length. Also, the given NE method was applied separately to the part of a genotype encoding input-hidden weights versus the part encoding hidden-output weights, so as not to recombine parts of a genotype responsible for distinctly different neural functions.

The genotype length was determined by the detection sensor (field of vision) setting (given by motor output MO_0). That is, the NE method also determined (indirectly via evolution of hidden-output connection weights) the field of vision most appropriate for a given controller (agent). As the value of MO_0 changed (given that it was the highest of all motor-output values at a given simulation time step), so to would the number of visual neurons in the sensory input layer, and the number of weights connecting visual neurons to hidden layer neurons.

As presented in table 2, the minimum genotype length was 10, and the maximum was 58. That is, 5 input-hidden weights connected the 5 non-visual neurons (SI_0..SI_4), between 1 and 49 input-hidden weights connected the same number of visual neurons (SI_5..SI_{53}), and 4 output-hidden weights connected the 4 motor output neurons (MO_0..MO_3) to the a given hidden layer neuron (HL_0..HL_9).

Action selection Action selection depended on whether the agent was using a heuristic or a NE method (comparative experiments were executed). In the case of a heuristic method, selection was according to a probabilistic preference, where as in the case of NE, selection was determined by the motor output node yielding the highest output value.

8 Heuristic Methods

For the heuristic method, we *hand-coded* specialization at the agent (table 3) and group (table 4) level, according to our definition of specialization (section 1.2). An agent was considered to be specialized if it dedicated more than 50% of its lifetime to one activity. A group was considered to be specialized if more than 50% of its agents were dedicated to one activity over the course of the groups lifetime.

The heuristic method used probabilistic preferences to determine which action to execute at each simulation iteration. The degree of agent specialization was thus defined and labeled, via setting a probabilistic bias to one of the four possible actions.

Table 3 presents the specialized agent types tested in experiments using the heuristic method. The composition of specialized (caste) versus non-specialized groups is specified in tables 4 and 5 respectively. For these specifications, CP denotes the portion of communicators, EP denotes the portion of evaluators, and DP denotes the portion of detectors. CP = 1 - DP - EP. The letters (V-ZA) denote the non-specialized group types. A blank space denotes a non applicable combination of detectors, evaluators, and communicators.

		Portion of Evaluators in Group (EP)					
		0	1/5	2/5	3/5	4/5	1
Portion of Detectors in Group (DP)	0	A	B	C	D	E	F
	1/5	G	H	I	J	K	
	2/5	L	M	N	O		
	3/5	P	Q	R			
	4/5	S	T				
	1	U					

Table 4. Heuristic Method: Specialized group types. The portion of Communicators is calculated as 1 minus portion of Detectors (DP) minus the portion of Evaluators (EP).

		Portion of Evaluators in Group (EP)		
		0	1/3	1/2
Portion of Detectors in Group (DP)	0			W
	1/3		V	X
	1/2	Y	Z	ZA

Table 5. Heuristic method: Non-specialized group types. The portion of Communicators is calculated as 1 minus portion of Detectors (DP) minus the portion of Evaluators (EP).

8.1 Specialized and Non-Specialized Group Types

Specialized group types were defined by setting more than 50% of a group to be of one specialized agent type (table 3). Table 4 presents the specialized group types tested using the heuristic method. Note that not all the group types presented in table 4 are specialized according to this definition. Group types M and N do not have a greater than 50% majority of any one agent type in their group composition. Non-specialized group types were defined when no single agent type had a greater than 50% majority in the groups composition. Table 5 presents the non-specialized group types tested using the heuristic method.

8.2 Red Rock Discovery Algorithm

The red rock discovery algorithm described the activity of aerial explorers with respect to discovering and evaluating red rocks, regardless of the controller type.

```
Red Rock Discovery Algorithm()
{ Simulate for N iterations (agent lifetime)
 {
  IF red rock evaluation data in memory (not communicated) THEN
  {
   Communicate red rock evaluation data to lander;
   Get fitness reward = r, and energy reward = e;
   IF lander not within communication range
   THEN communicate red rock data to agents in communication range;
  }
  Select action:[Detect, Evaluate, Move, Communicate];
  IF red rock detected THEN
  {
   Move to closest red rock with value
   Evaluate red rock (store in memory as red rock evaluation data);
   Communicate red rock evaluation data to lander;
   Get fitness reward = r, and energy reward = e;
   IF lander not within communication range
   THEN communicate red rock data to agents in communication range;
  }
  IF current energy < minimum energy threshold Move back to lander
  to recharge e units;
 } }
```

Experimental Parameters	
Communication Range	100 voxels
Communication Type	broadcast
Initial Aerial Explorer / Lander Battery	1000 / 100000 units
Detection / Evaluation Sensor Cost	0.5
Movement / Communication Actuator Cost	0.5
Maximum Move / Iteration	3 voxels
Simulation Length	2500
Initial Agent Positions	Random
Reproduce (Apply NE operators)	After evaluation
Energy / Fitness reward per red rock	100 / 1
Number of red rocks per simulation	40000
Simulation runs per experiment	20

Table 6. Agent and environment simulation parameters.

Environment Types									
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P	U	R	R	O	R	P	Q	R	Q

Table 7. Highest performing group types and the environment type (0-9) they performed best in.

9 Experiments and Results

We designed two sets of experiments. The first experiment set applied and measured the performance of the heuristic method using specialized and non-specialized agent groups. The second experiment set applied a conventional NE and the CONE method and tested their task performance comparatively with the 27 configurations 19 *specialized* group types and 8 *non-specialized* group types) of the heuristic method. Each experiment set was tested for 10 degrees of structure in 10 test environments (section 6.3). For all experiments, the performance measure used was the Red Rock Value Gathered (RRVG). Averages and standard deviations were calculated over 20 runs, where a single run consisted of 2500 iterations and a given method. The method used was either heuristic, or NE. The experimental agent and environment simulation parameters are presented in table 6.

9.1 Heuristic Method Comparison: Specialized versus Non-Specialized Groups

Table 8 presents the performance results from the heuristic method applied with specialized and non-specialized groups. The values in parentheses are the corresponding standard deviations. Highlighted values are the highest values attained for both specialized and non-specialized groups. Table 7 illustrates the highest performing groups for each of the 10 test environments (each degree of structure). Only group types in the range are (H-U) are displayed, since these were the highest performing group types.

9.2 Neuro-Evolution Method Comparison

Table 9 presents the performance results for the conventional NE method (A) versus the CONE method (B) when applied to the 10 test environments. The

Specialized versus Non-Specialized Group Types				
			Group Types: Specialized	Group Types: Non-Specialized
		0	2846 (855)	2124 (388)
D	S	1	2923 (931)	2103 (669)
E	O T	2	3178 (345)	**2723 (628)**
G	F R	3	3197 (285)	2501 (744)
R	U	4	3313 (421)	2414 (594)
E	C	5	3218 (977)	2115 (597)
E	T	6	3507 (818)	2470 (644)
	U	7	**3636 (322)**	2538 (776)
	R	8	3412 (847)	2475 (647)
	E	9	3313 (618)	2397 (713)

Table 8. Average RRVG for specialized (A-L; O-U) versus non-specialized group types (M, N, V-ZA).

highlighted values are the RRVG attained for each method. The value in parentheses are the corresponding standard deviations. For the NE methods, we determined if a given agent in a given group assumed a particular role via measuring what portion of its lifetime was spent on each of the detection, evaluation, and communication activities. An agent that spent the majority (more than 50%) of its lifetime on the detection activity was termed a detector. Similarly, the terms evaluator and communicator were applied for agents that spent a majority of their lifetimes on evaluation and communication. Likewise, convergence to a caste, via measuring the portion of detectors, evaluators and communicators that comprised a group, for the majority of the groups lifetime.

10 Analysis and Discussion

In order to draw conclusions from this comparative study, we performed a set of statistic tests in order to gauge respective differences between heuristic and NE method results. First, we determined results from the specialized and non-specialized heuristic (table 8), CONE and conventional NE (table 9) methods to be normal distributions via applying the Kolmogorov-Smirnov test [5]. P values were P=0.72, P=0.99, P=1.0, and P=0.98, respectively. To determine the statistical significance of difference between each of these data sets we applied an independent t-test [5]. For each t-test we selected 0.05 as the threshold for statistical significance, and stated the null hypothesis as two data sets not significantly differing. The t-test was first applied to the comparative specialized and non-specialized heuristic method data sets. P=0.00003 was calculated, meaning the null hypothesis was rejected. This served to support our first hypothesis of specialization being advantageous in terms of increasing task performance in particular environment types. Second, we applied the t-test to the comparative NE method results. A P=0.00007 was calculated, meaning the null hypothesis was rejected. This partially supported our second hypothesis that the CONE method would yield a high task performance.

Finally, we applied the t-test to the specialized heuristic method (table 8) and the conventional NE method (table 9) results. A P=0.32 was calculated, meaning the null hypothesis was accepted and there was no significant difference between task performance results. This served to partially support our second hypothe-

Neuro-Evolution Method Comparison			
		Method A: Conventional NE	Method B: CONE
DESCRIPTOR	0	2874 (365)	4290 (265)
	1	3338 (341)	4229 (350)
	2	3218 (417)	4189 (421)
	3	3040 (430)	4394(304)
	4	**3988 (451)**	4511 (321)
	5	3956 (338)	4898 (444)
	6	3633 (314)	4658 (321)
	7	3466 (384)	**4907 (481)**
	8	3525 (253)	4602 (406)
	9	2971 (441)	4638 (375)

Table 9. Average RRVG for the conventional NE (A) versus the CONE (B) method.

Group Composition of Conventional Neuro-Evolution Method					
Detectors	Evaluators	Communicators	No Specialization	DoS	Average RRVG
0.37	0.36	0.15	0.12	4	3988
Group Composition of Collective Neuro-Evolution Method					
Detectors	Evaluators	Communicators	No Specialization	DoS	Average RRVG
0.52	0.25	0.22	0.01	7	4907

Table 10. Comparative group compositions of best performing groups using conventional NE and the CONE methods. DoS denotes Degree of Structure.

sis that specialization was beneficial for task performance, via illustrating that an adaptive method with no specialization (table 10) yielded no significant advantage in performance. In terms of supporting our second hypothesis, that the CONE method derived specialized groups yielding high task performance, it is necessary to compare table 7 and table 10. The heuristic method showed that the best performing specialized group type (*caste Q*) was operating within *environment type 7*. As presented in table 4 the group composition of *caste Q* was such that a (60%) majority was the agent type *detector*. The remainder of the group composition was split between *evaluators* and *communicators*. Table 10 presents the group composition derived by the CONE method in *environment type 7* consisted of a majority of *detector* agents (0.52) and minor portions of *evaluator* (0.25) and *communicator* (0.22) agent types. This group composition resembled the *caste Q* in terms of consisting of a majority of detectors and two minorities of evaluators and communicators. This was not the case for the conventional NE method. This method yielded on average a comparable performance to the heuristic method using specialized group types for all test environments. Additionally, the conventional NE method was unable to out-perform the CONE method for all test environments (table 9). Table 10 illustrates that the best performing conventional NE run (test environment 4) as not converging to a caste. In table 10 only the group composition for environment type 4 is presented, however, this held true for all environment types. It is theorized that the inferior performance of the conventional NE (comparative to the CONE) method was a lack of derived specialization.

11 Conclusions

This paper described a comparative study of neuro-evolution and heuristic methods designed to test the efficacy and benefits of utilizing specialization as a means of increasing performance in a search and find task given a range of test environments. Performance comparisons were made according to the total value of features of interest (termed *red rocks*) discovered by a simulated multi-robot system. In support of our first hypothesis, a heuristic method using pre-defined specialized multi-robot groups elucidated that specialization was beneficial in a range of test environments (defined by different resource distributions. In support of our second hypothesis, our neuro-evolution method yielded a higher performance in all test environments, comparative to a conventional neuro-evolution method. The best performing group using the collective neuro-evolution method converged to a specialized group composition (such that the majority of the agents assumed one role) that resembled the group composition of the highest performing specialized group tested with the heuristic method. The comparatively low performance of the conventional method was deemed to be consequent of the lack of specialization exhibited in group compositions derived.

References

1. G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behavior. *Artificial Life*, 9(1):255–267, 2003.
2. B. Bryant and R. Miikkulainen. Neuro-evolution for adaptive teams. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pages 2194–2201. IEEE Press, Canberra, Australia, 2003.
3. L. Bull and J. Holland. Evolutionary computing in multi-agent environments: Eusociality. In *Proceedings of the Second Annual Conference on Genetic Programming*, pages 347–352. IEEE Press, San Francisco, USA., 1997.
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. D. Floreano and J. Urzelai. Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13(1):431–443, 2000.
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. N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of Third European Conference on Artificial Life (ECAL-95)*, pages 704–720. Springer-Verlag, Granada, Spain, 1995.
10. M. Kreiger and J. Billeter. The call of duty: Self-organized task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30: 65–84, 2000.

11. S. Nolfi and D. Floreano. Learning and evolution. *Autonomous Robots*, 7(1): 89–113, 1999.
12. S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 1(5):75–98, 1997.
13. P. Paalanen, J. Kamarainen, J. Ilonen, and H. Kälviäinen. Feature representation and discrimination based on gaussian mixture model probability densities - practices and algorithms. *Pattern Recognition*, 39(7):1346–1358, 2006.
14. 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.
15. K. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100, 2004.
16. S. Thakoor. Bio-inspired engineering of exploration systems. *Journal of Space Mission Architecture*, 2(1):49–79, 2000.
17. S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone. Evolving keep-away soccer players through task decomposition. In *Proceeding of the Genetic and Evolutionary Computation Conference*, pages 356–368. AAAI Press, Chicago, 2003.
18. L. Young, E. Aiken, G. Briggs, V. Gulick, and R. Mancinelli. Rotorcraft as mars scouts. In *Proceeding of the IEEE Aerospace Conference*, pages 4–12. IEEE Press, Big Sky, USA, 2002.