Evol. Intel.
DOI 10.1007/s12065-009-0034-z

RESEARCH PAPER

Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task

G. S. Nitschke · M. C. Schut · A. E. Eiben

Received: 4 July 2009 / Accepted: 6 November 2009

Abstract This article presents results from an evaluation of the collective neuro-evolution (CONE) controller design method. CONE solves collective behavior tasks, and increases task performance via facilitating emergent behavioral specialization. Emergent specialization is guided by genotype and behavioral specialization difference metrics that regulate genotype recombination. CONE is comparatively tested and evaluated with similar neuro-evolution methods 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. Results indicate that CONE is appropriate for deriving sets of specialized rover behaviors that complement each other such that a higher task performance, comparative to related controller design methods, is attained in the multi-rover task.

Keywords Neuro-evolution · Multi-rover · Collective behavior · Specialization

1 Introduction

Specialization is observable in many collective behavior systems and is thought to be a fundamental mechanism necessary to achieve optimal efficiency. Examples of collective behavior systems include social insect colonies, biological neural networks, traffic jams, economies of a nation, as well as industrial infrastructures such as energy and telecommunications networks [44]. In complex ecological communities, specializations have evolved over time as a means of diversifying the community in order to adapt to the environment [48]. Over the course of evolutionary time, specialization in biological communities has assumed both morphological [51] and behavioral forms [6].

For example, certain species of ants adapt their foraging behavior as a function of individual preference and colony demand, and have evolved specialized morphologies appropriate for different tasks such as foraging and nest construction [5]. Thus, labor is efficiently divided between specialized castes and individuals for the benefit of accomplishing group tasks. In such a sense, *specialization is an adaptive mechanism in collective behavior systems*. In fields of research such as *multi-robot systems* [47], it is highly desirable to reproduce the underlying mechanisms that result in replicating the success of biological collective behavior systems. One such underlying mechanism is *emergent behavioral specialization* [34]. *Emergent specialization* is that which emerges from the interaction of system components in response to a dynamic task that requires varying degrees, or types of specialization, in order to effectively accomplish. Such approaches are popular in collective behavior task domains where one does not know, a priori, the degree of specialization required to optimally solve the given task [30]. *Behavioral specialization* refers to agent behaviors that are advantageous for accomplishing specific types of tasks [34].

In the study of controller design methods that solve various collective behavior tasks [7, 27, 40, 46], emergent specialization is not typically used as a problem solving

G. S. Nitschke (✉)
Computational Intelligence Research Group,
Department of Computer Science, University of Pretoria,
Pretoria 0002, South Africa
e-mail: gnitschke@cs.up.ac.za

M. C. Schut · A. E. Eiben
Computational Intelligence Group,
Vrije Universiteit Amsterdam, De Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands

mechanism, but rather emerges as an ancillary result of the system accomplishing a collective behavior task. *Collective behavior task* refers a task that can only be solved if agents (behaviors) in the system cooperate. In this article the collective behavior task investigated is the *extended multi-rover task* (Sect. 4). Other collective behavior tasks that use or benefit from behavioral specialization include cooperative transport by multi-robot systems [36], pursuit-evasion tasks [57], RoboCup soccer [52], and multi-agent computer games [7].

This article tests collective neuro-evolution (CONE), which is a novel controller design method that addresses a gap in current controller design methods. CONE solves collective behavior tasks via purposefully facilitating emergent behavioral specialization [34]. The potential advantage of CONE is that it increases collective behavior task performance or attains collective behavior solutions that could not otherwise be attained without specialization.

1.1 Approach and objectives

In line with state of the art methods for controller design [15, 20], this research supports NE as an appropriate approach for controller design within continuous and partially observable collective behavior task environments. NE has been successfully applied to solve a disparate range of collective behavior tasks that include multi-agent computer games [7, 49, 50], RoboCup soccer [52], pursuit-evasion games [40, 57], and multi-robot tasks that include cooperative transport [35] and coordinated movement [4].

The extended multi-rover task (Sect. 4) investigated in this research elucidates that it is beneficial to use specialization that emerges during controller evolution, as part of the problem solving process. Multi-rover experiments indicate that CONE (comparative to related controller design methods) effectuates behavioral specialization in a set of controllers, allowing the derivation of collective behavior solutions that could not otherwise be derived. The multi-rover experiments also indicate that controllers adopting non-specialized behaviors produce a comparatively inferior collective behavior performance.

1.2 Research goals and hypotheses

In order to fulfill the following research goal and test the hypotheses, CONE is comparatively evaluated with two related NE controller design methods (*cooperative, co-evolutionary genetic algorithm* and *multi-agent enforced sub-populations*).

Research goal To demonstrate the efficacy of CONE as a controller design method that solves collective behavior tasks via effectuating behavioral specialization.

Hypothesis 1 CONE facilitates emergent *behavioral specialization* in a set of ANN controllers, where such specialization contributes to the evolution of collective behaviors that effectively solve the multi-rover task.

Hypothesis 2 Genotype and behavioral metrics in CONE adaptively regulate genotype recombination and encourage emergent behavioral specialization appropriate for achieving a higher task performance comparative to related controller design methods.

1.3 Contributions

The main contributions of this research are as follows.

1. *CONE:* CONE is an extension of the multi-agent ESP controller design method [57] that adapts a team of controllers in order to solve collective behavior tasks. Unlike multi-agent ESP, CONE includes adaptive mechanisms that autonomously regulate genotype recombination between multiple populations and controller size as a means of encouraging the emergence of behavioral specialization.
2. *Genotype and behavioral metrics:* CONE uses *genotype* and *behavioral* (Sect. 3.3) metrics. Such metrics have been proposed in previous research [3, 55]. However, the CONE metrics identify and propagate beneficial specialized controller behaviors as part of a cooperative co-evolutionary process.
3. *Dynamic controller size adaptation:* The number of hidden layer neurons in each ANN controller is adapted as a function of the fitness progress of all controllers. Dynamic controller size adaptation allows different controllers to evolve to sizes appropriate for solving different sub-tasks [24, 32].
4. *Emergent specialization as a problem solver:* Genotype and behavioral metrics, and dynamic adaptation of controller size allows CONE to purposefully evolve behaviorally specialized controllers. The interactions of these controllers results in a higher collective behavior task performance (comparative to that yielded by related methods). This assumes that the given collective behavior task benefits from behavioral specialization.

2 Approaches to specialization in collective behavior tasks

There has been a large amount of research on various methods for evolving specialization for solving collective behavior tasks. In addition to approaches delineated in the following, see Nitschke et al. [34] for a comprehensive review of various methods for evolving specialization for the benefit of solving collective behavior tasks.

2.1 Neuro-evolution

Neuro-evolution (NE) is the evolution of ANNs using evolutionary algorithms [56]. NE has been highlighted as being an appropriate method for controller design in collective behavior systems [15, 35, 41] as well as being effective for facilitating behavioral specialization [7, 40, 52].

Some prevalent research examples include Quinn et al. [43], who employed artificial evolution to adapt ANN controllers in a team of real robots. The team was given a coordinated movement task. Results indicated that complementary and specialized roles emerged that enabled the robots to move in a coordinated manner. Similarly, Bull and Holland [10] and Quinn [42] compared methods for the evolution of team behavior in a team of simulated robots. A method that specified the ANN controller of each robot as one genotype (*clonal*) was compared to a method that specified the controller of each robot as a different genotype (*aclonal*). These methods were tested on a coordinated movement task, and it was found that the *aclonal* approach evolved higher performance teams comparative to the clonal approach. Also, the highest performing teams evolved by the *aclonal* approach consisted of robots adopting complementary and behaviorally specialized roles.

Bryant and Miikkulainen [7] applied the *Enforced Sub-Populations* ESP [18] method to evolve behaviors of *barbarian* versus *legion* teams in a multi-agent computer game. A division of labor emerged during the evolution of legion teams. Some legion agents became specialized to pursuing barbarian agents, whilst others specialized to defending cities from barbarian agents. Whiteson et al. [53] apply the ESP method to evolve ANN controllers for specialized behavioral roles in a keep-away soccer task. Results indicated given a suitable task decomposition, NE (in this case ESP) was appropriate for evolving high performance collective (keep-away) behaviors. Potter and Meeden [40] applied NE for evolving ANN controllers in a simulated robot team. The authors demonstrated that emergent specialized behaviors were facilitated by increasing the number of skill sets necessary to solve the task. In their experiments, Potter and Meeden [40] added a predator robot which resulted in beneficial specialist behaviors emerging in herding robots. Baldassarre et al. [35] applied NE to evolve a collective object movement behavior in a simulated robot team. Specialized pushing versus pulling behaviors emerged enabling the team to cooperatively move an object when the robots were physically linked to each other.

2.2 Cooperative co-evolution

Cooperative co-evolution models [54] are suited for facilitating behavioral specialization when applied to solve collective behavior tasks. Such approaches segregate the solution space into multiple *species*, where individuals within a species constitute candidate partial solutions to a collective behavior solution. Cooperative co-evolution models use cooperation between species as well as competition between individuals of a species.

Holland and Reitman [23] did early work in cooperative co-evolution using Classifier Systems. A population of rules was evolved by assigning a fitness to each rule based on the success of interaction with other rules. Also, Husbands and Mill [25] used multiple genotype populations in order to evolve partial solutions to a manufacturing scheduling optimization task. These partial solutions were then combined in order to form a complete solution to the given task. However, in this research, emergent specialization did not play a role in problem solving. Potter and De Jong [37] developed a general cooperative co-evolution model that uses evolutionary algorithms. One particular instantiation of this generalized model was the CCGA. ESP co-evolves neurons for the purpose of evolving ANN controllers suited for a given task. In both CCGA and ESP, the genotype population is decomposed into a set of sub-populations. Each generation of the CCGA and ESP evolutionary process, the fittest genotype is selected from each sub-population. These genotypes are then decoded into partial solutions. Collectively, these partial solutions then represent a complete solution to a given task. This complete solution is evaluated on the given task, and a fitness is assigned back to each of the constituent genotypes (partial solutions). Thus, individuals are evaluated based upon how well they cooperate with other individuals for the purpose of accomplishing a given task. In CCGA, genotypes are general enough to encode various partial solutions, such as ANN controllers [37], and rules [38] (see also Bull et al. [9]). ESP is a specific case of CCGA, where each genotype is encoded as a string of values that represent input and output connection weights of a hidden layer neuron. Hence, combined, the fittest genotype selected from each sub-population represents the fittest ANN evolved for a given task.

Multi-agent ESP is the application of ESP to collective behavior tasks. Multi-agent ESP creates n populations for deriving n ANN controllers. Each population consists of u sub-populations, where individual controllers are constructed as in ESP. This process is repeated n times for n controllers, which are then collectively evaluated in a task environment. Multi-agent ESP is further described in related work [57].

Moriarty [28] applied a cooperative co-evolution approach in order to co-evolve a population of blueprints for ANNs and a population of neuron connection weights with which to construct ANNs. The population of ANN blueprints were evaluated based on how well their corresponding neurons solved a given task. In the other population, neurons received fitness based upon the number of successful blueprints they participated in. Drezewski [11] describes a multi-agent system that uses a cooperative co-evolutionary process with two species. The system solves multi-objective optimization tasks via locating the *pareto* frontier as a result of co-evolutionary interactions between the species.

2.3 State-of-the-art

In summary, although numerous studies have investigated methods to solve collective behavior tasks. Within such studies, behavioral specialization emerges as an ancillary result of task accomplishment. With notable exceptions such as [16], there are few examples of research that successfully specifies, a priori, what exactly the behavior of system components should be, in order to produce a specifically desired, yet emergent collective behavior. That is, a controller design method that purposefully facilitates and uses *emergent specialization* in order to solve collective behavior tasks is currently lacking.

3 Methods: collective neuro-evolution (CONE)

CONE is a controller design method that uses cooperative co-evolution in order to adapt a group of agent controllers. Each controller is an artificial neural network (ANN). Given n genotype populations (for evolving n controllers), CONE evolves one controller from each population, where the controllers must cooperate in order to solve collective behavior tasks. The controllers are collectively evaluated in a task environment according to how well they solve the given collective behavior task. Each controller is a feed-forward ANN with one hidden layer that is fully connected to the input and output layers. Each hidden layer neuron of each controller is encoded as one genotype. CONE evolves the connection weights of these neurons and then combines them into complete controllers.

The motivation for CONE operating at the neuron level is two-fold. First, NE methods that evolve neurons (or more generally functional units) have been demonstrated as exhibiting superior solutions for various controller design tasks [19, 21]. Second, cooperative co-evolutionary methods that operate at the neuron level avoid the competing conventions problem [45] and population diversity is maintained thus reducing the chance of premature convergence (relatively high for NE methods that evolve complete ANNs) [29].

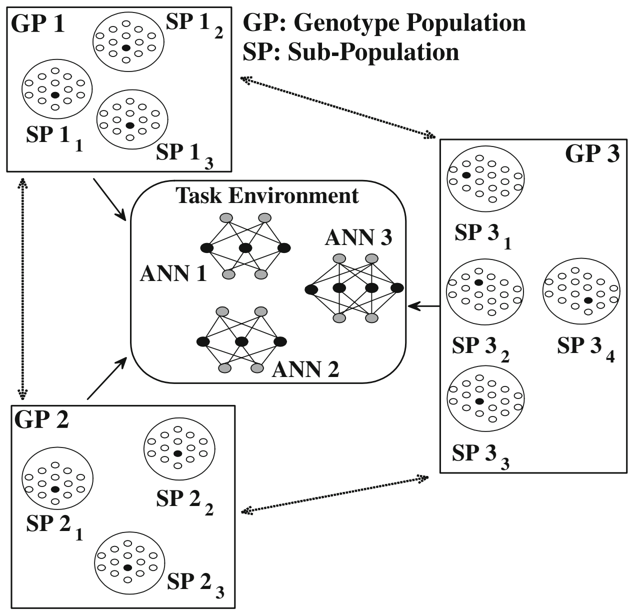

Fig. 1 *CONE example*. ANN controllers are derived from three populations and evaluated in a collective behavior task. *Doubleended arrows* indicate self regulating recombination occurring between populations

An example of CONE using three controllers (and thus three genotype populations) is presented in Fig. 1. Unlike related methods, which include SANE [28], CCGA [37], ESP [18], and multi-agent ESP [57], CONE implements genotype and behavioral specialization difference metrics to regulate genotype recombination *between* and *within* populations. Based upon genotype similarities and the success of behavioral specializations exhibited by controllers, these metrics control recombination and direct evolution. CONE is an extension of multi-agent ESP [57] that includes novel contributions that use emergent behavioral specialization as a problem solving tool.

1. *Genotype difference metric* (*GDM*): A heuristic that adaptively regulates recombination of similar and beneficial genotypes in different populations. For measuring genotype similarity, a *Genetic Similarity Threshold* (GST) is defined. Genotypes a and b are considered similar if the average weight difference [55] between a and $b < GST$. Given that similar genotypes in different populations may encode very different functionalities, recombining similar genotypes may produce genotypes (neurons) that do not work in a controller. The specialization difference metric addresses this problem.
2. *Specialization difference metric* (*SDM*): A heuristic that regulates genotype recombination based on behavioral similarities exhibited by controllers. The SDM ensures that specialized behavior exhibited by two controllers is sufficiently similar, before two

populations that encode the two controllers, can be recombined. If the SDM calculates the behavior of two controllers to be sufficiently similar, then the GDM is applied in order to recombine similar genotypes within the two populations[1].

3. *Controller size adaptation:* A heuristic that adapts the number of hidden layer neurons in each controller over the course of cooperative co-evolution. Controller size adaptation supports the facilitation of behavioral specialization by CONE, via allowing different controllers to evolve to different sizes. That is, controllers of varying sizes and complexity are often appropriate for solving sub-tasks of varying complexities [32].

For succinctness, this section presents a description of the CONE architecture (Sect. 3.1), CONE's novel contributions (Sects. 3.2 to 3.3), and an overview of the CONE process (Sect. 3.5). For a comprehensive description of CONE refer to Nitschke [32].

3.1 Representation: multi-population structure

As with related NE methods [18, 37], CONE segregates the genotype space into n populations for the purpose of developing n ANN controllers. CONE mandates that ANN_i ($1 \leq i \leq n$) is derived from genotype population $i(P_i)$, where P_i contains u_i sub-populations. ANN_i is derived from P_i via selecting one genotype from each of the u_i sub-populations and decoding these genotypes into hidden layer neurons. Figure 1 illustrates an example of the CONE representation. Three populations for deriving three controllers are illustrated. 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. ANN_i consists of w input neurons, and v output neurons, fully connected to u_i hidden layer neurons. The number of input and output neurons remains fixed and the number of hidden layer neurons can be adapted. The CONE process is driven by mechanisms of cooperation and competition within and between sub-populations and populations. There is competition between genotypes of a sub-population, given that the genotypes compete for a place as a neuron in the hidden layer of a fittest controller. There is cooperation between sub-populations, in that a fittest genotype is selected from each sub-population in order to participate in forming a controller. There is cooperation between controllers given that n controllers must cooperate in order to accomplish a collective behavior task.

[1] The SDM and GDM regulate recombination between populations such that the chances of genotype recombination producing deleterious offspring is minimal.

3.2 Behavioral specialization

An integral part of CONE is defining behavioral specialization exhibited by controllers, and measuring specialization similarities. The *degree of behavioral specialization* (S) exhibited by an individual controller is defined by the frequency with which the controller switches between executing distinct motor outputs (actions) during its lifetime. Controllers select from executing one of at least two different actions. Given this, the behavioral specialization metric used is an extension of that defined by Gautrais et al. [17].

This specialization metric was selected since it is applicable to the behaviors of individual controllers, accounts for the partitioning of a controller's work effort among different actions, and is simple enough to extend for the purposes of working within CONE. The metric is general enough to define controllers as being specialized when they regularly switch between different actions, and an approximately equal portion of the controller's lifetime is devoted to each action, but where there is a preference for executing a particular action. The metric is also general enough to be applicable to controllers that have distinct motor outputs, but a variable range for each output. For example, the metric was applied in a multi-robot experiment, where each robot had two motor outputs controlling left and right wheel speeds. Each motor output was executed with a value indicating its wheel speed [31]. For a given controller, S is calculated as the frequency with which a controller switches between each of its v actions (Eq. 1). In Eq. 1, A is the number of times the controller switches between different actions, and N is the total number of possible action switches. Equation 1 assumes that each controller has v distinct motor outputs (actions).

$$S = \frac{A}{N} \tag{1}$$

A value of S close to zero indicates a high degree of specialization, where a controller specializes to primarily perform one action, and switches between this and the other $v - 1$ actions with a low frequency. An S value close to one indicates a low degree of specialization, where a controller switches between some or all of its v actions with a high frequency. In the case of a perfect specialist ($S = 0$), a controller executes the same action for the duration of its lifetime ($A = 0$). An example of a non-specialist ($S = 0.5$) is where a controller spends half of its lifetime switching between different actions. For example, if $A = 5$, $N = 10$, $v = 2$ then the controller would switch between each of its actions every second iteration. Controllers are labeled as either *specialized* or *not specialized* according to the following rule.

- If $S \geq \mathrm{ST}$ then a controller is labeled as *non-specialized*.
- If $S < \mathrm{ST}$ then a controller is labeled as *specialized*.

Where, ST is the behavioral specialization threshold. Throughout this paper we use ST = 0.5. If a controller is defined as being specialized, then it is labeled as being *specialized to action x*, where *x* is the action that is most executed over the course of the controller's lifetime. Otherwise, a controller is labeled as being *non-specialized*. Also, if a set of controllers are defined as being specialized, then controllers are grouped according to their specialization label. A controller group specialized to executing the same action is called a *caste* [26]. For measuring the similarity between the specialized behaviors of two controllers ANN_i and ANN_j a specialization distance (*SD*) is defined (Eq. 2).

$$SD(ANN_i, ANN_j) = |S(ANN_i) - S(ANN_j)| \tag{2}$$

ANN_i and ANN_j have similar behavioral specializations if *SD* is less than a specialization similarity threshold (SST), where, SST ∈ {0, ..., 1} (Eq. 3).

$$SD(ANN_i, ANN_j) < SST \tag{3}$$

3.3 Adaptation of algorithmic parameters

This section describes the mechanisms that adapt the *Genetic Similarity Threshold* (GST) and *Specialization Similarity Threshold* (SST). The purpose of the GST and SST is to autonomously regulate recombination between different populations as a function of the fitness progress of the *n* controllers. GST and SST values are adapted by 0.01, which was determined from exploratory experiments run prior to multi-rover experiments (Sect. 4). This did not change the GST and SST by too much per regulation, thus missing useful values, and was not too small, thus inhibiting the discovery of effective GST and SST values.

Specialization for regulating recombination: This heuristic regulates the SST as a function of behavioral specialization and fitness progress.

1. If the average *degree of specialization* (*S*) measured for at least one of the fittest *n* controllers has increased over the previous *V* generations, and average fitness (for the fittest *n* controllers) stagnates or is decreasing over this same period, then decrement the SST value. That is, if the fittest controllers have an average *S* that is too high for improving team fitness, then recombination between populations is restricted.
2. If the average *S* of at least one of the fittest *n* controllers has decreased over the last *V* generations, and average fitness stagnates or is decreasing over this same period, then increment the SST value. That is, if the fittest controllers have an average *S* that is too low to improve team fitness, then allow for more recombination between populations.

Recombination for regulating recombination: This heuristic regulates the GST value as a function of recombinations and fitness progress.

1. If recombinations between populations have increased over the previous $V + W$ generations, and fitness has stagnated or decreased, then decrement the GST value.
2. If recombinations between populations have decreased or stagnated, and fitness has stagnated or decreased over the previous $V + W$ generations, then increment the GST value.

3.4 Adapting controller size

Controller size (the number of hidden layer neurons and sub-populations from which each hidden layer neuron is derived), is dynamically adapted as a function of collective behavior performance. If the fitness of at least one of the *n* fittest controllers has not progressed in $V + W + Y$ generations, then the size of the stagnating controllers is adapted. Specifically, the number of sub-populations in the population from which the stagnating controller is derived is adapted. This differs from related NE methods that evolve *n* controllers with fixed sensory input and motor output topologies in collective behavior tasks [7, 40, 57]. In such related work there was no dynamic adaptation of controller size in collective behavior tasks. Controller size adaptation in CONE allows for the derivation of controller sizes that effectively complement each other for the purpose of collective behavior task accomplishment. In collective behavior tasks comprised of sub-tasks of varying degrees of complexity and difficulty, controllers of different sizes work more effectively at solving complementary sub-tasks, and thus cooperating. A lesion mechanism [28] is applied in order to determine if the size of a stagnating controller should be increased or decreased. When an additional sub-population is created, new genotypes are created by randomly initializing each gene to a value within a range stipulated by the task. Multi-rover experiments (Sect. 4) with controller size adaptation *turned off* yielded an average task performance decrease of ~25% for all experiments.

3.5 Overview of the collective neuro-evolution (CONE) process

1. *Initialization. n* populations are initialized. Population $P_i (i \in \{1, ..., n\}$ contains u_i sub-populations.

Sub-population P_{ij} contains m genotypes. P_{ij} contains genotypes encoding neurons assigned to position j in the hidden layer of ANN_i (ANN_i is derived from P_i).

2. *Evaluate all genotypes*. Systematically select each genotype g in each sub-population of each population, and evaluate g in the context of a complete controller. This controller (containing g) is evaluated together with $n - 1$ other controllers. Other controllers are constructed via randomly selecting a neuron from each sub-population of each of the other populations. The evaluation results in a fitness being assigned to g.
3. *Evaluate elite controllers*. For each population, systematically construct a fittest controller from a genotype (randomly selected from the *elite portion*[2]) of each sub-population of the population. Controller fitness is determined by its *utility*. A controller's utility is determined by a simple sum of the fitness value the genotypes corresponding to each of its hidden layer neurons. Groups of the fittest n controllers are evaluated together in task simulations until all genotypes in the elite portion of each population have been assigned a fitness. For each genotype, this fitness overwrites the previously calculated fitness.
4. *Parent selection*. If the two fittest controllers ANN_i and ANN_j constructed from the elite portions of P_i and P_j are calculated as having *similar behavioral specializations* (Sect. 3.2) then these populations become candidates for recombination. For P_i and P_j to be recombined, both ANN_i and ANN_j must have the same specialization label (Sect. 3.2). That is, both ANN_i and ANN_j must be behaviorally specialized to the same action. Between P_i and P_j each pair of sub-populations is tested for *genetic similarity*. Genetically similar sub-populations are recombined, and mutation applied. Recombination occurs *within sub-populations* that *not* genetically similar to other sub-populations. Similarly, recombination occurs *within* populations that are *not behaviorally similar*.
5. *Recombination*. For recombination that occurs *between* a pair of sub-populations, the elite portion of genotypes in each sub-population is ranked by fitness. Genotypes with the same fitness rank are recombined. For recombination *within* a sub-population, each genotype in the sub-population's elite portion is systematically selected and recombined (using one-point crossover [12]) with a randomly selected elite portion genotype.
6. *Mutation*. After recombination, *burst mutation* with a *Cauchy* distribution [18] is applied to each gene of each genotype with a predefined degree of probability.
7. *Parameter adaptation*. If fitness of one of the fittest controllers has not progressed in:

   (a) V generations: Adapt the *Genetic Similarity Threshold* (GST).
   (b) $V + W$ generations: Adapt the *Specialization Similarity Threshold* (SST).
   (c) $V + W + Y$ generations: Adapt the number of sub-populations (controller size) of each controller with stagnating fitness (Sect. 3.4). V, W, Y are constants.

8. *Stop condition*. Reiterate steps [2, 7] until a desired collective behavior task performance is achieved, or the evolutionary process has run for X generations.

4 The extended multi-rover task

This multi-rover task requires a team of simulated autonomous vehicles (rovers), to detect features of interest (*red rocks*) with a maximal total value over the course of the team's lifetime. The term *red rock* is adapted from Young et al. [58] and refers to discrete high-value features of interest on an unexplored terrain. A red rock is *detected* when it is within range of more than one rover's *red rock detection* sensors. In this research, detecting a red rock represents is a metaphor for retrieving, collecting, or acquiring red rock value. Accordingly, a rover's red rock sensors are an abstraction of actuators necessary for red rock retrieval. The multi-rover task is a collective behavior task, meaning that a red rock is detected if at least two rovers can sense it. Furthermore, we distinguish five types of red rocks (A, ..., E), and three different resolution settings for red rock detection sensors (*low-res*, *med-res*, *hi-res*). For each type of red rock, we define a specific combination of sensor settings required to detect it. These are shown in Table 1, together with the value of each red rock type. This multi-rover task extends that described by Agogino [1], and differs in a number of respects.

1. This is a collective behavior task, and not a distributed artificial intelligence task, given that rovers are required to cooperate in order to accomplish the task (detect red rocks).
2. Rovers use detection sensors with variable settings (resolutions), where as in previous work [1], rovers operate with detection sensors always being active with a fixed setting.

[2] The elite portion is a fittest portion of a given genotype population. For the experiments described in this article, it was set to 50%. Random selection from the elite portion was opted for given previous research results that have demonstrated that evaluating only the current fittest individuals from each species is not necessarily the best approach [8, 39].

Table 1 Collective behavior for red rock detection

Red rock type	Red rock value	Rovers required
A	1	2 Low-res detectors
E	1	2 Detectors (using same settings)
B	1	1 Low-res, 1 med-res detector
C	1	1 Med-res, 1 hi-res detector
D	1	2 Hi-res detectors

Rover's need to use given detection sensor settings in order to cooperatively detect given red rock types

3. In this task, red rock detection requires different rovers to adopt different sensor resolutions. This encourages behavioral specialization for task accomplishment.

In line with the work of Young et al. [58] red rocks are arranged in a canal like structure, thus forming *canal environments*. In Young et al. [58], it was the goal of autonomous vehicles to maximize detection of red rocks via following these canal like structures. This was also the motivation for the use of red rock canals within these experiments, however, the goal of maximizing red rock detection has been made more complex via introducing collective behavior requirements. Canals defined by red rocks represent positions that are impassable to rovers. Hence, to move through the environment, rovers must navigate around these canals. Further details of and illustrations of the environments used in this study are described in Sect. 5.1 as part of the specification of the experimental setup.

4.1 Rovers: detecting other rovers and red rocks

Each rover is equipped with two types of sensors: those to detect other rovers and those to detect red rocks. The rover detection sensors fulfill two functions. First, they prevent collisions between rovers. Second, they provide each rover with an indication of red rock detection sensor settings being used by other rovers. Each rover is equipped with eight rover detection sensors, S-0 through S-7, each of which covering one quadrant in a 360 degree sensory field of view (FOV). The rover detection sensors are constantly active, maintaining a default value of zero, when not sensing anything, and have a fixed range and cost (Table 3). When another rover enters the range of a rover detection sensor, then that sensor is assumes a new value equal to the red rock detection sensor setting being used by the closest rover divided by the squared distance to this rover. To this end, we encode the symbolic values lo-res, med-res, and hi-res as integers 1, 2, and 3, respectively.

The red rock detection sensors (that are, in fact, abstractions of actuators) are for detecting red rocks as described above. Each rover is equipped with eight red rock detection sensors, S-8 through S-15 each of which covering one quadrant in a 360 degree sensory FOV. Red rock detection sensors have a fixed range and cost (Table 3). When red rocks come within range of sensor q, that sensor returns a value inversely proportional to the value of, divided by the squared distance to, the closest red rock.

4.2 Rovers: artificial neural network (ANN) controller

Each rover uses a recurrent ANN controller [13], that fully connects 22 sensory input neurons to six hidden layer neurons and five motor output neurons (Fig.2). Sensory input neurons [SI-0, SI-7] accept input from each of the eight rover detection sensors. Neurons [SI-8, SI-15] accept input from each of the eight red rock detection sensors. Neurons [SI-16, SI-21] accept the previous activation value of each hidden layer neuron. Furthermore, the five motor output neurons [MO-0, MO-4], are fully connected to the hidden layer neurons. The hidden and output layer neurons are sigmoidal units [22].

At each simulation time step, a rover executes one of four actions (Fig. 2). The motor output with the highest value is the action executed (MO-3 or MO-4 indicate movement). A rover's heading is determined by normalizing and scaling the vectors dx and dy by the maximum distance a rover can traverse in one simulation iteration [2].

1. MO-0: Activate all red rock detection sensors with *low-res setting*.
2. MO-1: Activate all red rock detection sensors with *med-res setting*.
3. MO-2: Activate all red rock detection sensors with *hi-res setting*.
4. MO-3, MO-4: Move in a direction calculated from MO-3 (dx) and MO-4 (dy).

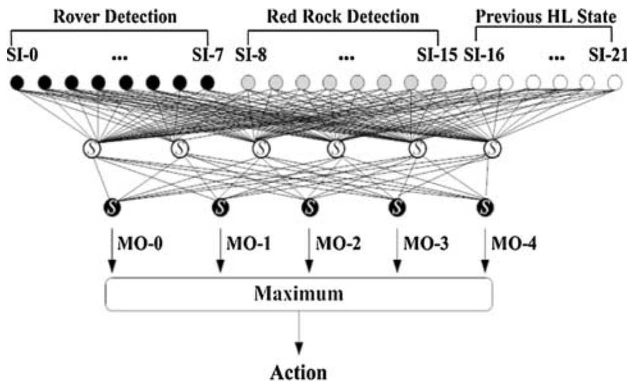

Fig. 2 *Rover ANN controller*. For clarity not all sensory input neurons are presented

4.3 Rovers: heuristic controller

Heuristic controllers are used in a lesion study conducted as part of the post-experiment analysis (Sect. 7). Four different heuristic controllers implement hard-wired *specialized* and *non-specialized* behaviors (Table 2). Each controller is defined by a set of probabilistic preferences for selection of one action at each simulation iteration. These actions are the same as those used by the ANN controller. The action selection preference values were selected given that they produced a specialized or non-specialized behaviors, where specialization is defined as switching between different actions with a low frequency, and non-specialization is defined as switching between different actions with a high frequency (Sect. 4.4).

4.4 Specialization in the multi-rover task

In the original rover task [2], each rover could only move at each simulation iteration, and red rock detection sensors were constantly active. In this task, each rover can select between multiple actions, and red rock detection requires at least two rovers to use complementary detection sensor settings. Hence, this task encourages different rovers to specialize to different actions for the purpose of collective behavior task accomplishment. Related research [33] indicated that the extended multi-rover task benefits from behavioral specialization given rover sensor and energy constraints. These constraints prevent an effective heuristic based systematic search of a large environment by the rovers. Specialization is measured with respect to the behavior exhibited by individual rovers. The specialization metric (Sect. 3.2) calculates a degree of specialization (S) for a given rover. Specialized rover's are labeled as follows. Label assignment is done after each rover's lifetime, since the portion of a rover's lifetime spent executing each action must be known.

- *Low-res detector:* A rover that activates its red rock detection sensors at *low-res* for a period of simulation time greater than any other action.
- *Med-res detector:* A rover that activates its red rock detection sensors at *med-res* for a period of simulation time greater than any other action.

Table 2 Rover heuristic controllers

Controller type	Low-res detection (%)	Med-res detection (%)	Hi-res detection (%)	Move (%)
Low-res detector	70	0	0	30
Med-res detector	0	70	0	30
Hi-res detector	0	0	70	30
Non-specialized	25	25	25	25

Exhibit a specialized (*low-res*, *med-res*, *hi-res* detectors) or non-specialized behavior. Probabilistic preferences are used for action selection

- *Hi-res detector:* A rover that activates its red rock detection sensors at *hi-res* for a period of simulation time greater than any other action.
- *Mover:* A rover that moves for a period of simulation time greater than any other action.

5 Extended multi-rover task: experimental design

Rovers operate in a continuous simulation environment which is characterized by a two dimensional plane. One rover can occupy any given x, y position in the environment. Movement is calculated in terms of real valued vectors. To calculate the distance between this rover and other rovers and red rocks, the squared Euclidean norm, bounded by a minimum observation distance [2] is used. Furthermore, the environment is defined by the following.

1. *Red rock distribution*. Red rock types: [A, B, C, D, E] are distributed in 10 different environments (Sect. 5.1). Environments are classification as *simple* if they contain only one type of red rock, and *complex* if they contain multiple types of red rock.
2. *Rovers*. Teams of 20 rovers initialized in random positions in each environment.

5.1 Canal environments: red rock distribution

The ten canal environments used in this study are illustrated in Figs. 3, 4, and 5. Red rocks are distributed such that a red rock can placed at each possible x, y position, within the confines of a given canal structure. The total red rock value and the number of red rocks equals 5,000 for each environment. These canals were selected from exploratory experiments that found such environments to contain sufficient complexity in order that the least effective teams attained a near zero task performance, and the most effective teams achieved a relatively high performance. Experiments tested 20 rovers in the simulation environment and measured the impact of a *controller design method* and *environment* upon *red rock value detected* by a rover team. The experimental objective was to determine which controller design method maximizes the red rock value detected by a rover team. Furthermore, the contribution of emergent behavioral specialization to rover team task performance is investigated.

- *Controller design methods*: Each rover's ANN controller is adapted with either: CCGA [37], multi-agent ESP [57], CONE (Sect. 3).
- *Environment*: For each method, two sets of environments (labeled: *simple* and *complex*) are tested. Each set contains ten red rock distributions (Sect. 5.1).

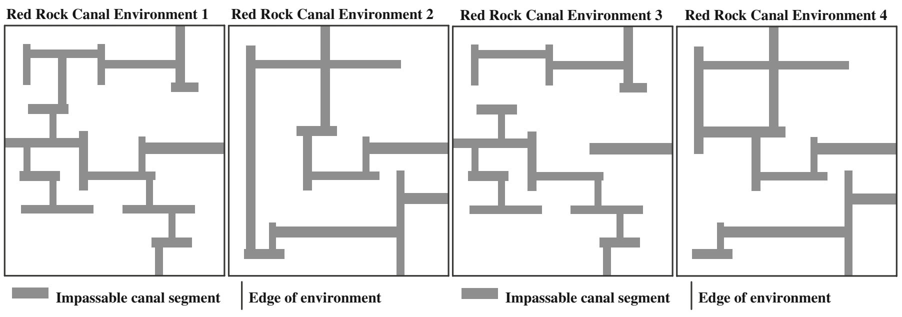
Red Rock Canal Environment 1
Red Rock Canal Environment 2
Red Rock Canal Environment 3
Red Rock Canal Environment 4
Impassable canal segment
Edge of environment
Impassable canal segment
Edge of environment

Fig. 3 Canal environments [1–4]

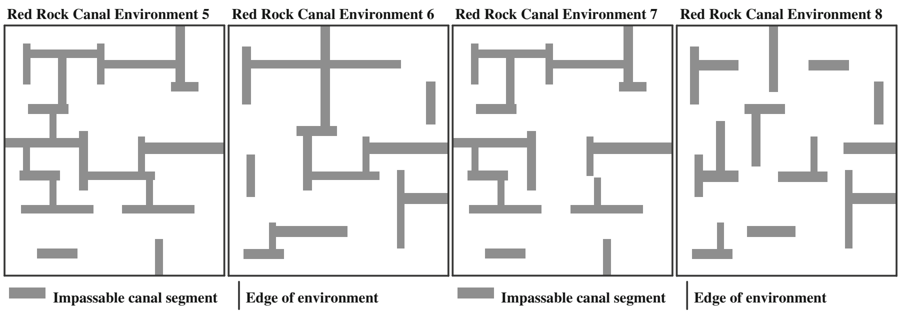
Red Rock Canal Environment 5
Red Rock Canal Environment 6
Red Rock Canal Environment 7
Red Rock Canal Environment 8
Impassable canal segment
Edge of environment
Impassable canal segment
Edge of environment

Fig. 4 Canal environments [5–8]

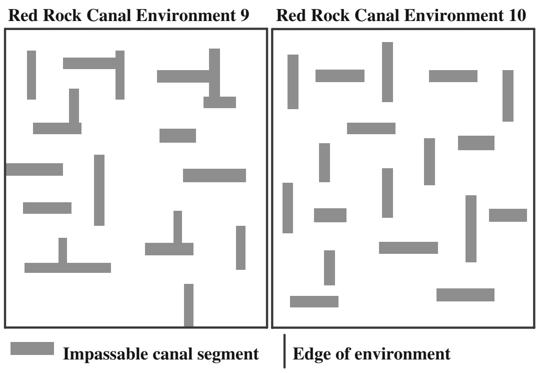

Fig. 5 Canal environments [9, 10]

Rover Team Fitness Evaluation. To evaluate team performance, a global fitness function, G, is defined as a the total *red rock value detected* by a team over the course of its lifetime. This fitness calculation is also used as each rover's private fitness function (g_η). The goal of a team is to maximize G. However, rovers do not maximize G directly. Instead each rover η attempts to maximize g_η. G does not guide evolution, but rather provides a measure of team task performance. It is g_η that guides rover controller evolution.

Experimental Setup and Parameters. Table 3 presents the simulation and NE parameter settings. These settings were derived in a set of exploratory experiments, which indicated that minor changes to these parameter values produced similar results. Changing the number of iterations per epoch, to lower values decreased experiment running time, but did not provide rovers with lifetimes that were long enough to widely explore the environment. Lowering the number of generations and epochs also decreased experiment running time, but did not provide the artificial evolution process with sufficient time to derive rover teams that were as effective as those reported upon in Sect. 5. More than 20 runs per experiment were not used due to time constraints and the time consuming nature of the experiments. Less than 20 runs were not used since, since a reasonable sample size of results was required for each method in order to draw sound statistically based conclusions.

Each experiment consists of 500 generations. Each generation corresponds to the *lifetime* of each rover in the team. Each rover lifetime lasts for 10 epochs, where each epoch consists of 2500 simulation iterations. Each epoch is a task scenario that tests different rover starting positions, and red rock locations (within a given distribution) in the simulation environment. Rover task performance (red rock value detected) is calculated as an average taken over all epochs of a rover's lifetime. The highest task performance is then selected for each rover in each experiment. An average team task performance is calculated over 20 runs.

6 Multi-rover task results

This section presents results from the task performance comparison of CONE, CCGA and multi-agent ESP evolved teams. In the following, *task performance* refers to the red rock value detected by a rover team, and *caste* refers to a set of rovers that are specialized to the same

Table 3 Simulation and neuro-evolution parameter settings for the multi-rover experiments

Simulation and neuro-evolution parameters	
Rover movement range	0.01
Red rock/rover detection sensor range	0.05
Initial rover positions	Random
Environment width/height	1.0
Total red rocks (value) in environment	5,000
Red rock distribution	10 Canal environments (Sect. 5.1)
Generations	500
Epochs	10
Iterations per epoch (rover lifetime)	2500
Mutation probability (per gene)	0.05
Mutation type	Burst (Cauchy distribution)
Mutation range	[−1.0, +1.0]
Fitness stagnation Y	15 Generations (CONE/multi-agent ESP)
Fitness stagnation V/W	10 Generations (CONE)
Genotype/specialization threshold (GST/SST)	[−1.0, 1.0] (CONE)
Genotype/specialization distance (GD/SD)	[0.0, 1.0] (CONE)
Genotype population elite portion	50%
Weight (gene) range	[−10.0, +10.0]
Crossover	Single point
ANN sensory input neurons	22
ANN hidden layer neurons	6 (Adapted with CONE only)
ANN motor output neurons	5
Genotype	1 neuron (multi-agent ESP, CONE), 1 ANN (CCGA)
Total genotypes	10,000
Genotype populations/ number of rovers	20
Genotype length	31 (CONE, multi-agent ESP), 310 (CCGA)
Genotypes per population	500

behavior [26]. One experiment consists of evolving a rover team using a given NE methods in a given environment, and consists of an *evolution* and a *testing* phase.

- *Evolution phase:* Rover team controllers are evolved for 500 generations (Table 3).
- *Testing phase:* The fittest team (n controllers) are selected and run in the same environment for one rover lifetime. During this phase the evolved connection weights of each rover's controller remains static. Task performance results presented are averages calculated over 20 runs of the fittest team in each test environment.

In order to compare the task performances yielded by different rover teams a statistical comparison is conducted between two given sets of task performance data. The following procedure is followed for a statistical comparison between any given two data sets.

- The Kolmogorov–Smirnov test [14] is applied to each of the data sets in order to check if the data sets conform to normal distributions.
- To determine if there is a statistically significant difference between performance results of any two teams, an independent t-test [14] is applied. The threshold for statistical significance is 0.05. The null hypothesis is that data sets do not significantly differ.

6.1 Experiments: environments appropriate for behavioral specialization

These experiments compare the task performance of rover teams in *simple* and *complex* environments. *Simple* environments contain a distribution of only one red rock type. *Complex* environments contain a distribution of multiple red rock types. Experiment results first illustrate that *complex* environments are appropriate for encouraging behavioral specialization during the NE of rover controllers, where as, the simple environments do not encourage emergent behavioral specialization. There are 10 simple and 10 complex environments, where each is a different canal environment (Sect. 5.1). Each *simple environment* contains only distributions of *type E red rocks* (Table 4). *Type E red rocks* are detectable by sensors operating at any setting (Table 1). Each *complex environment* contains a distribution of *type [A, B, C, D] red rocks* (Sect. 5.1). Two statistical comparisons of performance results yielded by teams evolved by CCGA, multi-agent ESP, and CONE in *simple* (the reader is referred to nitschke [32]) and *complex* environments, were conducted.

1. A task performance comparison between teams evolved by CCGA, multi-agent ESP and CONE in the *simple* environments (results presented in nitschke [32]).
2. A task performance comparison between teams evolved by CCGA, multi-agent ESP and CONE (Fig. 6) in the *complex* environments.

Table 4 Distribution of red rock types per environment set

Environment set	Type-A red rocks	Type-B red rocks	Type-C red rocks	Type-D red rocks	Type-E red rocks	Red rock value
Complex	0	0	0	0	5,000	5,000
Simple	500	500	500	500	0	5,000

Each of the 10 simple and complex environments is defined by a different red rock distribution (canal structure)

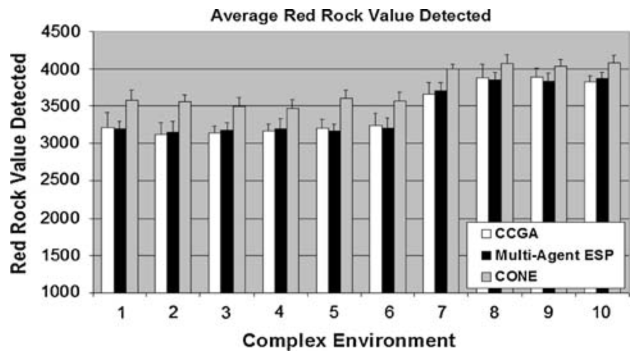

Fig. 6 *Average red rock value detected in the complex environments*. Comparative results yielded by CCGA, multi-agent ESP, and CONE evolved teams

6.1.1 *Results: teams evolved in simple and complex environments*

Figure 6 presents the average task performance for teams evolved by CCGA, multi-agent ESP and CONE in the *complex environments*. Tables 5, 6, and 7 present the behavioral compositions of the fittest teams evolved by CCGA, multi-agent ESP, and CONE in the complex environments. A statistical comparison of task performances results yielded by teams evolved in the *simple* and *complex* environment sets, indicate the following results.

1. Average task performance of teams evolved by CCGA, multi-agent ESP, and CONE in the *complex environments* is significantly higher than that in the *simple environments*.
2. A comparison of behavioral compositions of the fittest teams evolved by CCGA, multi-agent ESP, and CONE in *simple* and *complex* environments, indicates that complex environments are appropriate for evolving of teams comprised of multiple *castes*.

These results indicate that the complex environments encourage the derivation of teams composed of complementary behavioral specializations, where such specializations result in a higher task performance comparative to teams evolved in simple environments.

6.1.2 *Results: teams evolved within the complex environment set*

A statistical comparison of task performances yielded by CCGA, multi-agent ESP, and CONE in the *complex environments*, indicate the following results.

1. CCGA, multi-agent ESP, and CONE all evolved teams comprised of complementary castes (Tables 5, 6, and 7).

Table 5 Behavioral composition of fittest CCGA evolved teams

CM	Low-res detectors	Med-res detectors	Hi-res detectors	Mover	NS
1	4	6	9	0	1
2	2	8	8	0	2
3	2	5	8	0	5
4	1	6	7	0	6
5	0	3	11	0	6
6	0	3	10	0	7
7	0	2	9	0	9
8	0	3	10	0	7
9	0	0	10	0	10
10	0	0	11	0	9

For teams of 20 rovers. *Low-/Med-/Hi-res detectors* rovers specialized to low-res, med-res, and hi-res detection, respectively, *Mover* rovers specialized to moving, *NS* non-specialized rover, *CM* complex environment

Table 6 Behavioral composition of fittest multi-agent ESP evolved teams

CM	Low-res detectors	Med-res detectors	Hi-res detectors	Mover	NS
1	2	6	6	0	6
2	3	5	7	0	5
3	3	6	5	0	6
4	0	6	5	0	9
5	0	5	5	0	10
6	0	4	7	0	9
7	0	3	6	0	11
8	0	4	8	0	8
9	0	3	7	0	10
10	0	2	12	0	6

For teams of 20 rovers. *Low-/med-/hi-res detectors* rovers specialized to low-res, med-res, and hi-res detection, respectively, *Mover* rovers specialized to moving, *NS* non-specialized rovers, *CM* complex environment

2. For all 10 complex environments, CONE evolved teams yielded a significantly higher task performance, comparative to that of CCGA and multi-agent ESP evolved teams.

These results support the hypothesis that CONE facilitates behavioral specialization in teams, where such specialization results in a comparatively higher task performance.

7 Multi-rover task experimental analysis

A statistical comparison of the task performance results presented in Fig. 6 indicates that the average task

Table 7 Behavioral composition of fittest CONE evolved teams

CM	Low-res detectors	Med-res detectors	Hi-res detectors	Mover	NS
1	6	7	5	0	2
2	5	8	6	0	1
3	5	6	7	0	2
4	4	5	7	0	4
5	4	4	7	0	5
6	5	5	6	0	4
7	4	5	6	0	5
8	5	4	7	0	4
9	3	5	8	0	4
10	3	4	8	0	5

For teams of 20 rovers. *ow-/med-/hi-res detectors* rovers specialized to low-res, med-res, and hi-res detection, respectively, *Mover* rovers specialized to moving, *NS* non-specialized rover, *CM* complex environment

Table 8 Emergent controller sizes (hidden layer neurons) in castes

CM	Low-res detectors	Med-res detectors	Hi-res detectors	Mover	NS
1	7	8	5	6	11
2	6	7	6	6	9
3	6	5	6	7	10
4	6	6	7	5	9
5	7	7	6	8	8
6	6	7	7	6	8
7	6	7	6	5	10
8	6	6	7	7	12
9	7	5	5	5	11
10	7	7	7	6	9

For the fittest team in each environment. *Low-/med-/hi-res detectors* rovers specialized to low-res, med-res, and hi-res detection, respectively, *Mover* rovers specialized to moving, *NS* non-specialized rover, *CM* complex environment

performance of CONE evolved teams is significantly higher than that of CCGA, and multi-agent ESP evolved teams, for all complex environments. Table 7 presents the fittest team evolved by the CONE method for each environment as being comprised of one non-specialized and multiple specialized castes. Specialized castes are those sets of rovers specialized to *low-res*, *med-res*, and *hi-res* red rock detection behavior (Sect. 4.1). This result supports the research hypothesis that CONE is appropriate for deriving a level of behavioral specialization such that a higher collective behavior task performance, comparative to related methods, is achieved.

7.1 The role of castes

The emergence of *non-specialized* and *specialized* castes in the fittest teams evolved by CCGA (Table 5), multi-agent ESP (Table 6), and CONE (Table 7) in each of the complex environment, is attributed to the collective behavior requirement in the multi-rover task. That is, detection of a red rock's value depends upon the red rock's type and the settings of the rovers' red rock detection sensors (Sect. 4.1). So, in order for rovers to detect a near optimal value of red rocks, rover teams are required to adopt specializations to different red rock detection sensor settings. This in turn results in the emergence of low-res, med-res and hi-res detector as well as non-specialized castes. The emergent non-specialized caste (in each of the fittest teams) results from rovers that frequently switch between movement and activating sensors with low-res, med-res, and hi-res settings. It is theorized that the non-specialized caste complements the specialized castes for the purpose of detecting a high value of red rocks. That is, when coupled with a specialized rover, a non-specialized rover provides the necessary second detection sensor setting in order that a red rock within the FOV of both rovers is detected. This statement is supported by the caste lesion study.

7.2 Emergent controller sizes in castes

Table 8 presents, for the fittest team (evolved by CONE), the average emergent controller size (number of hidden layer neurons) for all specialized rovers. The specializations are: *Low-Res*, *Med-Res*, and *Hi-Res* Detector, and *Mover*. The average controller size for the non-specialized caste (in the fittest team) is also presented. The results show that the non-specialized rover caste, on average, consist of controllers that have evolved larger controller sizes, comparative to the specialized castes. Controllers in the *Low-Res Detector*, *Med-Res Detector*, *Hi-Res Detector*, and *Mover* castes are, on average, approximately the same size. These results indicate that specialized controllers require comparatively less internal state compared to that of non-specialized controllers. It is theorized that, for non-specialized controllers, a high frequency of switching between behaviors occurs, requiring a larger internal state than that of a controller which switches with a low frequency between behaviors.

7.3 Rover caste lesion study

In order to investigate the role of emergent behavioral specialization in evolved rover teams, a *caste lesion study*, was conducted. The goal of this study was to ascertain the contribution of specialized and non-specialized castes to the task performance of the fittest teams. The lesion study evaluates the fittest teams evolved by CCGA, multi-agent ESP, and CONE via systematically removing specialized and non-specialized castes, replacing them with specialized or non-specialized heuristic controllers (Sect. 4.3), and then re-evaluating the task performance of the given team. Each lesioned team is executed in 20 new experimental runs for each complex environment. An average *red rock value detected* is then calculated over these 20 experimental runs for each environment. It is important to note that for each of the fittest teams evolved by each method, castes are removed and then re-evaluated in the environments in which they were evolved. This means that castes within a fittest team are often re-evaluated in a subset of the complex environment set.

- *Fittest teams evolved by CCGA in complex environments* (Table 5):

  [1, 4]: *Low-res* detector caste is replaced with low-res heuristic controllers.
  [1,8]: *Med-res* detector caste is replaced with med-res heuristic controllers.
  [1,10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers.
  [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers.

- *Fittest teams evolved by multi-agent ESP in complex environments* (Table 6):

  [1, 3]: *Low-res* detector caste is replaced with low-res heuristic controllers.
  [1, 10]: *Med-res* detector caste is replaced with med-res heuristic controllers.
  [1, 10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers.
  [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers.

- *Fittest teams evolved by CONE* (Table 7):

  [1, 10]: *Low-res* detector caste is replaced with low-res heuristic controllers.
  [1, 10]: *Med-res* detector caste is replaced with med-res heuristic controllers.
  [1, 10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers.
  [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers.

Lesion study results indicate that the task performance yielded by the fittest teams evolved by CCGA, multi-agent ESP and CONE depend upon the behavioral roles fulfilled by each of the castes as well as the interaction of these castes. Furthermore, the lesion study results indicate that the fittest CONE evolved teams (for each complex environment) are more reliant upon the constituent specialized and non-specialized castes, comparative to the fittest CCGA and multi-agent ESP evolved teams in the same environments. The contribution of individual castes to team task performance is most pronounced in the fittest CONE evolved teams. However, the performance of the fittest CCGA and multi-agent ESP evolved teams are more robust when castes are removed. That is, there is less of a reduction in team task performance when either the non-specialized, low-res, med-res, or hi-res detector castes are removed. This indicates that there is less interdependency between constituent castes in the fittest CCGA and multi-agent ESP teams. These teams do not rely as much as the fittest CONE evolved teams upon the interactions between different castes in order to achieve the task performances presented in Fig. 6. Performance results of teams re-executed with castes removed (replaced with heuristic controllers) are presented in nitschke [32].

7.4 Difference metrics (GDM and SDM) study

This study supports the efficacy of the GDM, and SDM for facilitating behavioral specialization, and increasing task performance in CONE evolved teams. To support this, CONE is re-executed with the following three variations of the original experimental setup (Sect. 5).

1. **CONE without GDM (CONE-1 in Fig**. 7): Teams are evolved using CONE without the GDM (Sect. 3), meaning that genotype recombination only occurs *within* sub-populations of a given population. This was also the case for multi-agent ESP. The SDM for inter-population recombination is active.
2. **CONE without SDM (CONE-2 in Fig**. 7): Teams are evolved using CONE without the SDM (Sect. 3). The GDM remains active.
3. **CONE without GDM and SDM (CONE-3 in Fig**. 7): Teams are evolved using CONE without both the GDM and SDM.

Figure 7 presents the *average red rock value* detected by teams evolved using CONE, without the GDM, SDM, and both the GDM and SDM, for all complex environments. These task performance results are averaged over 20 experimental runs for each variation of the CONE setup and each environment. For comparison, results previously attained by CONE (original experimental setup) evolved teams are also presented in Fig. 7. A statistical comparison of results presented in Fig. 7 indicates that, for all complex environments, there is a significant difference between the task performance of teams evolved by CONE, and that of teams evolved by CONE-1, CONE-2, and CONE-3. That is, the teams evolved by CONE yield a performance advantage over the CONE variants. This result supports the research hypothesis that both the GDM and SDM are

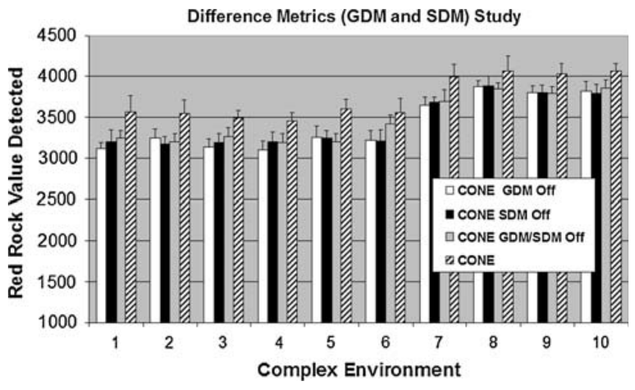

Fig. 7 *Red rock value detected by teams evolved with CONE variants in complex environments* The CONE variants (CONE-1, CONE-2, CONE-3) are described in Sect. 7.4

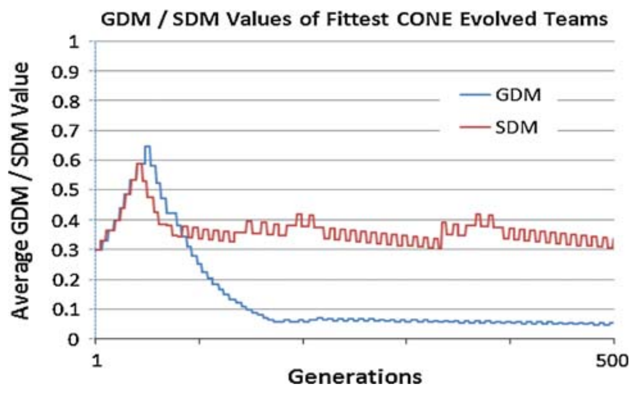

Fig. 8 *Average GDM and SDM values for the fittest CONE evolved teams*. Averages calculated via taking the fittest team from each (complex) environment

beneficial in terms of increasing task performance in CONE evolved teams. Without either the GDM or SDM, the CONE evolved teams lose their advantage of a significantly higher task performance.

Furthermore, the CONE variants, for all complex environments, did not evolve teams that were comprised of multiple complementary and interacting castes, as was the case for CONE (Table 7). Figure 8 presents the progression of average GDM and SDM values over the evolutionary runs of the fittest CONE evolved teams. The average is calculated from the progression of GDM and SDM values for the fittest CONE evolved team in each complex environment. Figure 8 indicates that in the initial stages of evolution (<100 generations), both the GDM and SDM adapt to allow for more recombination to occur between populations. The SDM, for the rest of each run, then adapts to a point where it fluctuates between a value of 0.3 and 0.4. It is theorized that this is an optimal range of values for *how similar* specialized behaviors of different rovers should be, in order for the propagation of beneficial specializations to occur. Similarly, the GDM is adapted such that after 200 generations it stabilizes to a value equal to ~ 0.05. This indicates that genotypes in different populations (of behaviorally similar rovers) need to be very similar in order to be recombined. Also, this provides an explanation for why either the GDM or SDM are ineffective alone. The GDM and SDM working in company provide a double check mechanism for determining if recombination should occur between populations. That is, having both the SDM and GDM as regulation mechanisms reduces the amount of recombination between populations and increases the likelihood that only similar and beneficial specialized behaviors are recombined and propagated.

8 Conclusions

This article investigated the application of the CONE controller design method for evolving collective behaviors in a team of simulated rovers. The multi-rover task stipulated that a team of rovers must maximize the value of features of interest (red rocks) detected in a simulated environment, where cooperative sensor usage was required. CONE was found to be successful for evolving controllers that specialized to complementary sensor settings necessary for an effective collective red rock detection behavior. The task performance of CONE evolved teams was compared with teams evolved by related controller design methods. An analysis of results elucidated that CONE was able to facilitate a degree of behavioral specialization in rover controllers necessary for teams to achieve a high task performance (comparative to teams evolved by related methods).

References

1. Agogino A (2003) Design and control of large collections of learning agents. Ph. D. Dissertation, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, USA
2. Agogino A, Tumer K (2004) Efficient evaluation functions for multi-rover systems. In: Proceedings of the genetic and evolutionary computation conference. Springer, New York, pp. 1–12
3. Balch T (1998) Behavioral diversity in learning robot teams. PhD Thesis, College of Computing, Georgia Institute of Technology, Altanta
4. Baldassarre G, Nolfi S, Parisi D (2003) Evolving mobile robots able to display collective behavior. Artif Life 9(1):255–267
5. Bonabeau E, Dorigo M, Theraulaz G (1998) Swarm intelligence: from natural to artificial systems. Oxford University Press, Oxford
6. Bonabeau E, Theraulaz G, Deneubourg J (1996) Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. Proc R Soc Lond B 263(1): 1565–1569

7. Bryant B, Miikkulainen R (2003) Neuro-evolution for adaptive teams. In: Proceedings of the congress on evolutionary computation. IEEE Press, Canberra, pp. 2194–2201
8. Bull L (1997) Evolutionary computing in multi-agent environments: partners. In: Proceedings of the seventh international conference on genetic algorithms. Morgan Kaufmann, East Lansing, pp. 370–377
9. Bull L, Fogarty T, Snaith M (1995) Evolution in multi-agent systems: evolving communicating classifier systems for gait in a quadrupedal robot. In: Proceedings of the sixth international conference on genetic algorithms. Morgan Kauffman, San Mateo, pp. 382–388
10. Bull L, Holland J (1997) Evolutionary computing in multi-agent environments: eusociality. In: Proceedings of the second annual conference on genetic programming. IEEE Press, San Francisco, pp. 347–352
11. Drezewski R (2003) A model of co-evolution in multi-agent system. In: Multi-agent systems and applications III. Springer, Berlin, pp. 314–323
12. Eiben A, Smith J (2003) Introduction to evolutionary computing. Springer, Berlin
13. Elman J (1990) Finding structure in time. Cogn Sci 14(1):179–211
14. Flannery B, Teukolsky S, Vetterling W (1986) Numerical recipes. Cambridge University Press, Cambridge
15. Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evol Intell 1(1):47–62
16. Funes P, Orme B, Bonabeau E (2003) Evolving emergent group behaviors for simple humans agents. In: Proceedings of the European conference on artificial life. Springer, Berlin, pp. 76–89
17. Gautrais J, Theraulaz G, Deneubourg J, Anderson C (2002) Emergent polyethism as a consequence of increased colony size in insect societies. J Theor Biol 215(1):363–373
18. Gomez F (2003) Robust non-linear control through neuroevolution. PhD thesis, Department of Computer Sciences, The University of Texas, Austin
19. Gomez F, Miikkulainen R (1999) Solving non-markovian control tasks with neuroevolution. In: Proceedings of the international joint conference on artificial intelligence. Morgan Kaufmann, Stockholm, Sweden, pp. 1356–1361
20. Gomez F, Schmidhuber J, Miikkulainen R (2006) Efficient non-linear control through neuroevolution. In: Machine learning: ECML 2006. Springer, Berlin, pp. 654–662
21. Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9(1):937–965
22. Hertz J, Krogh A, Palmer R (1991) Introduction to the theory of neural computation. Addison-Wesley, Redwood City
23. Holland J, Reitman J (1978) Cognitive systems based on adaptive algorithms. Pattern Direct Inference Syst 7(2):125–149
24. Hurst J, Bull L (2006) A neural learning classifier system with self-adaptive constructivism for mobile robot control. Artif Life 12(3):353–380
25. Husbands P, Mill F (1991) Simulated co-evolution as the mechanism for emergent planning and scheduling. In: Proceedings of the fourth conference on genetic algorithms. Morgan Kaufmann, Cambridge, pp. 264–270
26. Kreiger M, Billeter J (2000) The call of duty: self-organized task allocation in a population of up to twelve mobile robots. Rob Auton Syst 30(1):65–84
27. Li L, Martinoli A, Mostafa Y (2002) Emergent specialization in swarm systems. In: Lecture notes in computer science, vol 2412. Intelligent data engineering and automated learning. Springer, Berlin, pp. 261–266
28. Moriarty D (1997) Symbiotic evolution of neural networks in sequential decision tasks. PhD thesis, Department of Computer Sciences, The University of Texas, Austin
29. Moriarty D, Miikkulainen R (1996) Efficient reinforcement learning through symbiotic evolution. Mach Learn 22(1):11–33
30. Murciano A, Millan J (1997) Learning signaling behaviors and specialization in cooperative agents. Adapt Behav 5(1):5–28
31. Nitschke G (2007) Neuro-evolution methods for designing emergent specialization. In: Proceedings of the ninth European conference on artificial life. Springer, Lisbon, pp. 1120–1130
32. Nitschke G (2008) Neuro-evolution for emergent specialization in collective behavior systems. PhD thesis, Computer Science Department, Vrije Universiteit, Amsterdam
33. Nitschke G, Schut M, Eiben A (2006) Collective specialization for evolutionary design of a multi-robot system. In: Proceedings of the second international workshop on swarm robotics. Springer, Rome, pp. 189–206
34. Nitschke G, Schut M, Eiben A (2007) Emergent specialization in biologically inspired collective behavior systems. In: Intelligent complex adaptive systems. IGI, New York, pp. 100–140
35. Nolfi S, Baldassarre G, Parisi D (2003) Evolution of collective behaviour in a team of physically linked robots. In: Applications of evolutionary computing. Springer, Berlin, pp. 581–592
36. Nolfi S, Deneubourg J, Floreano D, Gambardella L, Mondada F, Dorigo M (2003) Swarm-bots: swarm of mobile robots able to self-assemble and self-organize. Ecrim News 53(1):25–26
37. Potter M, De Jong K (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29
38. Potter M, De Jong K, Grefenstette J (1995) A co-evolutionary approach to learning sequential decision rules. In: Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, San Franciso, pp. 366–372
39. Potter M, De Jong K (1994) A cooperative coevolutionary approach to function optimization. In: Proceedings of the third conference on parallel problem solving from nature (PPSN III). Springer, Jerusalem, pp. 249–257
40. Potter M, Meeden L, Schultz A (2001) Heterogeneity in the co-evolved behaviors of mobile robots: the emergence of specialists. In: Proceedings of the international joint conference on artificial intelligence. AAAI Press, Seattle, pp. 1337–1343
41. Quinn M (2000) Evolving cooperative homogeneous multi-robot teams. In: Proceedings of the international conference on intelligent robots and systems (IROS 2000). IEEE Press, Takamatsu, pp. 1798–1803
42. Quinn M (2001) A comparison of approaches to the evolution of homogeneous multi-robot teams. In: Proceedings of the congress evolutionary computation. IEEE Press, Seoul, pp. 128–135
43. Quinn M, Smith L, Mayley G, Husbands P (2009) Evolving teamwork and role-allocation with real robots. In: Proceedings of the 8th international conference on artificial life. MIT Press, Cambridge, pp. 302–311
44. Resnick M (1997) Turtles, termites, and traffic jams: explorations in massively parallel microworlds. MIT Press, Cambridge
45. Schaffer J, Whitley D, Eshelman L (1992) Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Whitley D, Schaffer J (eds) Proceedings of an international workshop on the combinations of genetic algorithms and neural networks (COGANN-92). IEEE Press, New York
46. Schultz A, Bugajska M (2000) Co-evolution of form and function in the design of autonomous agents: micro air vehicles project. In: Proceedings of the workshop on evolution of sensors in nature, hardware, and simulation, GECCO. AAAI Press, Chicago, pp. 154–166
47. Schultz C, Parker L (2002) Multi-robot systems: from swarms to intelligent automata. Kluwer, Washington DC
48. Seligmann H (1999) Resource partition history and evolutionary specialization of subunits in complex systems. Biosystems 51(1):31–39

49. Stanley K, Bryant B, Miikkulainen R (2005) Evolving neural network agents in the nero video game. In: Proceedings of the IEEE 2005 symposium on computational intelligence and games. IEEE Press, Piscataway, pp. 182–189
50. Stanley K, Bryant B, Miikkulainen R (2005) Real-time neuro-evolution in the nero video game. IEEE Trans Evol Comput 9(6):653–668
51. Wenseleers T, Ratnieks F, Billen J (2003) Caste fate conflict in swarm-founding social hymenoptera: an inclusive fitness analysis. Evol Biol 16(1):647–658
52. Whiteson S, Kohl N, Miikkulainen R, Stone P (2003) Evolving keep-away soccer players through task decomposition. In: Proceeding of the genetic and evolutionary computation conference. AAAI Press, Chicago, pp. 356–368
53. Whiteson S, Kohl N, Miikkulainen R, Stone P (2005) Evolving keepaway soccer players through task decomposition. Mach Learn 59(1):5–30
54. Wiegand R (2004) An analysis of cooperative coevolutionary algorithms. PhD. Thesis, George Mason University Press, George Mason University, Fairfax
55. Wineberg M, Oppacher F (2003) The underlying similarity of diversity measures used in evolutionary computation. In: Proceedings of the genetic and evolutionary computation conference. Springer, Berlin, pp. 1493–1504
56. Yao X (1993) Evolutionary artificial neural networks. J Neural Syst 4(3):203–222
57. Yong C, Miikkulainen R (2007) Coevolution of role-based cooperation in multi-agent systems. Technical Report AI07-338. Department of Computer Sciences, The University of Texas, Austin
58. Young L, Pisanich G, Ippolito C (2005) Aerial explorers. In: Proceeding of the 43rd AIAA aerospace sciences meeting and exhibit. AIAA Press, Reno, pp. 4–12