VRIJE UNIVERSITEIT

Neuro-Evolution for Emergent Specialization in Collective Behavior Systems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op vrijdag 6 maart 2009 om 10.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Geoffrey Stuart Nitschke

Promotor:	Prof.dr. A.E. Eiben
Co-promotor:	Dr. M.C. Schut

Promotiecommissie:

Prof.dr. K. De Jong
Prof.dr. D. Floreano
Prof.dr. F. Groen
Prof.dr. W. Kowalczyk
Prof.dr. J. Treur

CONTENTS

1. INTRODUCTION

Specialization is observable in many complex adaptive systems[1] and is thought to be a fundamental mechanism in many complex adaptive systems in order to achieve optimal efficiency. In complex ecological communities, specializations have evolved over time as a means of diversifying the community in order to adapt to the environment [151]. Over the course of evolutionary time, specialization in biological communities has assumed both morphological [180] and behavioral forms [19]. For example, morphologically specialized castes have emerged in certain termite colonies [117], and honey bees dynamically adapt their foraging behavior for pollen, nectar, and water as a function of individual preference and colony demand [26]. The consequence of such specializations is that 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 a complex adaptive system.*

This thesis introduces the *Collective Neuro-Evolution* (CONE) method. This method is designed to derive sets of *Artificial Neural Network* (ANN) controllers[2] operating for the purpose of solving multi-agent tasks. Such tasks are herein referred to as *collective behavior tasks* given that the behaviors of multiple agents collectively interacting is required in order to derive solutions.

An example is the *Gathering and Collective Construction* (GACC) task [115]. This task requires a group of robots to search a given environment for objects of different types. Object types are defined by color. In this example the object types are: *red*, *blue*, and *green*. This particular task example requires that green objects be gathered first, blue objects be gathered second, and red objects be gathered last. The term *gathered* refers to the discovery and subsequent transportation of an object by robots to a home area in the environment. Green objects can be gathered by individual robots. However, blue objects require two robots in order to gather, and red objects require three robots in order to gather. Task performance is measured according to how fast the robots can gather all the objects in the correct sequence. One method of optimally accomplishing this task is for most robots to specialize to gathering the next object type in the sequence, where such an object is also the most scarce

[1] Examples of complex adaptive 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 [142].

[2] The terms *ANN* and *controller* are used interchangeably throughout the thesis. However, the term *ANN controller* refers to a simulated (software) or embodied (robotic) entity designed to operate in a given environment and perform a given task. Such an entity uses a neural computation process [70] in order to map its sensory inputs to motor outputs.

in the environment. Simultaneously, other robots would specialize to gathering object types that are next in the required sequence, where such objects are progressively less scarce. This approach assumes that the robots are aware of the required sequence of object types, and will thus wait until the required object types have been delivered before a newly gathered object is delivered. Hence, the GACC collective behavior task requires that at any given time, a particular portion of the robot group specialize to gathering each of the object types, where these portions collectively produce an optimal gathering behavior.

The CONE method is specifically designed to facilitate and utilize behavioral specialization[3] as a problem solving mechanism in collective behavior tasks. CONE increases collective behavior task performance or attains collective behavior solutions that could not otherwise be derived without specialization. CONE utilizes a set of ANN controllers as its problem solving substrate and artificial evolution [43] as the method for developing specific solutions. CONE is currently limited to the use of simple ANN controllers with a static number of input and output neurons and a single hidden layer. However, the relaxing of these limitations will be discussed in chapter 7.

1.1 Approach and Objectives

In this thesis, an *artificial collective behavior* system is a simulated (software based multi-agent) system, where components of the system interact in order to solve a collective behavior task. The design of such artificial collective behavior systems draws inspiration from nature. Artificial collective behavior systems often replicate desirable behaviors exhibited in biological collective behavior systems. Examples of biological collective behavior systems include complex ecological communities such as social insect colonies [151], [26], [19], biological neural networks [9], multi-cellular organisms [66], economies of a nation, companies and other business organizations [1], [106].

In fields of research such as *artificial life* [86], *multi-robot systems* [150], *neural computation* [70] and *evolutionary computation* [43], 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 specialization*. Specialization is either behavioral or morphological and for the purposes of designing collective behavior systems is either specified *a priori* or is an emergent property[4] of the system.

Given that sets of controllers are applied to solve a collective behavior task, it is feasible that emergent specialization increases the task performance of the controllers. Specifically, specialization that results (emerges) from the evolution (design) of an individual controller, or emerges from the interactions of multiple controllers may be used as part of a collective behavior problem solving

[3] *Behavioral specialization* as opposed to *morphological specialization* [113].

[4] The term *emergent property* is used interchangeably with *emergent behavior*. Both refer to any computation that achieves global affects, formally or stochastically, via working within a bounded number of neighbors and without the use of global visibility [47].

process. In the study of collective behavior systems within a broad range of research fields [21, 88, 149, 137], emergent specialization is typically not used as a problem solving mechanism, but rather emerges as an ancillary result of the system accomplishing its given task. With a few notable exceptions such as Zhang, Martinoli and Antonsson [202], and Bugajska and Schultz [22], the role of emergent morphological specialization in collective behavior problem solving has not been widely studied. This is consequent of the engineering challenges and inherent complexities of dynamically creating morphologically specialized robots and computer hardware components, that represent effective solutions to emerging challenges in a physical task environment [90, 130, 133, 178]. Given the complexity of issues that must be addressed in designing methods that evolve (or otherwise manufacture) collective behavior systems that use morphological specializations, this research focuses on collective behavior systems that derive emergent behavioral specialization as part of a controller design process.

Hence, emergent behavioral specialization is beneficial to, and often necessary for solving collective behavior tasks. That is, many collective behavior tasks benefit from, or require, sets of agents adopting complementary specialized problem solving behaviors in order to solve. In line with state of the art methods for ANN controller design [60, 49], this research uses *Neuro-Evolution* (NE) as a means of adapting a set of agents (ANN controllers) that are given the goal of solving a collective behavior task. CONE is a novel controller design method that addresses a gap in current state of the art controller design methods. Currently, specialization emerges as a side effect of controller design methods that solve collective behavior tasks. A controller design method that solves given collective behavior tasks via purposefully facilitating and using emergent behavioral specialization is currently lacking. CONE explicitly uses specialization that emerges as part of a controller design process in order to solve collective behavior tasks. Thus, the general research objective is defined as follows.

Define a controller design method that facilitates and uses emergent behavioral specialization such that any given collective behavior task is solved.

A cooperative co-evolution approach is employed in order to collectively derive, evaluate, and adapt sets of controllers for the purpose of solving collective behavior tasks. This cooperative co-evolution approach is formalized as the CONE method (chapter 3). Specific to CONE is the method's capability to dynamically identify the *degree of behavioral specialization* required by the given collective behavior task. An appropriate degree of behavioral specialization is then facilitated for each controller in a set of controllers that are working collectively. The purpose of such effectuated behavioral specialization is for the set of controllers to increase their collective behavior task performance, or to achieve a collective behavior solution that could not otherwise be achieved.

In this research a set of collective behavior tasks that benefit from a problem solving approach that uses behavioral specialization are identified. Such collective behavior tasks are those that are most effectively solved by each agent (in an agent group) adopting a complementary behavioral role. In such tasks

(chapters 4, 5, and 6), the interactions of specialized behaviors accomplishes collective behavior tasks with a near optimal performance.

Collective behavior case studies described demonstrate that CONE derives a degree of *behavioral specialization* that is appropriate for achieving a higher task performance, comparative to related controller design methods (chapter 2). Related controller design methods were selected given that such methods are applicable to collective behavior tasks and are appropriate for deriving both behaviorally specialized and non-specialized controllers.

1.2 Research Goals and Hypotheses

Given the need for a controller design method that explicitly facilitates and uses emergent specialization as a means of solving collective behavior tasks, the following research goals are formulated.

Research Goal 1: To demonstrate the viability of a controller design method that solves collective behavior tasks via effectuating and leveraging behavioral specialization as part of controller design and problem solving.

Research Goal 2: To evaluate a novel controller design method (CONE: *Collective Neuro-Evolution*), where CONE accomplishes research goal 1.

Given the second research goal, the following hypotheses are formulated.

Hypothesis 1: CONE facilitates emergent *behavioral specialization* in a set of ANN controllers, where such specialization contributes to solving (or increasing task performance in) a given collective behavior task.

Hypothesis 2: Difference metrics defined as part of the CONE architecture, that adaptively regulate genotype recombination between populations, encourage the evolution of a degree of behavioral specialization in ANN controllers appropriate for achieving a higher collective behavior task performance comparative to related NE methods.

Each hypothesis as well as the overall efficacy of CONE as a controller design method for collective behavior systems is comparatively tested with four related methods. These are, HomCNE: *Homogenous Conventional Neuro-Evolution*, HetCNE: *Heterogenous Conventional Neuro-Evolution* (section 2.3.1), CCGA: *Cooperative, Co-evolutionary Genetic Algorithm* (section 2.3.3), and Multi-Agent ESP: *Multi-Agent Enforced Sub-Populations* (section 2.3.3).

1.3 Contributions

The main contributions of this research are outlined in the following.

1. *Collective Neuro-Evolution:* The *Collective Neuro-Evolution* (CONE) controller design method. CONE derives sets of controllers for the purpose of solving collective behavior tasks.

2. *Emergent Specialization as a Problem Solver:* CONE dynamically and purposefully derives sets of behaviorally specialized controllers, where the interactions of these controllers potentially results in a near optimal collective behavior task performance. As an important part of the controller design process, the CONE method identifies if a given collective behavior task benefits from emergent behavioral specialization. CONE then derives controllers that exhibit a degree of behavioral specialization appropriate for accomplishing the given collective behavior task.

3. *Difference Metrics:* CONE includes *behavioral specialization* (section 3.2.2) and *genotype* (section 3.1.2) difference metrics. Genotype and behavioral difference metrics have been proposed in previous research such as that of Wineberg and Oppacher [190] and Balch [10]. However, these difference metrics identify and propagate beneficial specialized controller behaviors as part of the cooperative co-evolutionary process used by CONE. Specialized behaviors are then used as part of the problem solving process.

4. *Behavioral Specialization Metric:* A metric to measure behavioral specialization in controller behavior (section 3.2.1) is defined as a part of CONE. The specialization metric is applicable to controller behavior that is defined by distinct actions (as in the case of the multi-rover and GACC tasks) as well as behavior defined by a combination of actions (as in the case of the pursuit-evasion task).

5. *Empirical Evidence:* Three collective behavior case studies yield an abundance of empirical data that support the efficacy of CONE.

There are many collective behavior tasks that potentially benefit from applying CONE. These tasks include multi-robot systems that collectively transport cumbersome objects [85], [98], [167], [64] multi-robot systems that collectively construct structures in hazardous or uninhabitable environments, such as underwater habitats or orbiting space stations [183], [182], [181] and multi-robot systems that collectively survey hazardous or dangerous territories [148], [51], [150], [5]. Additional tasks include nanobot collective behavior systems designed for accomplishing manufacturing or medical tasks [28], and interactive multi-agent computer games designed to train or entertain human players [161].

Given the variety of tasks, in both simulated and physical task environments, that potentially benefit from behavioral specialization, the provision of CONE as an automated controller design method is deemed a valuable contribution to currently deficient controller design methods in collective behavior research.

1.4 Neuro-Evolution for Controller Design

This research supports the notion that NE is an appropriate approach for controller design methods that operate in continuous and partially observable collective behavior task environments. Also, NE is appropriate for purposefully deriving behavioral specialization within sets of controllers attempting to accomplish a collective behavior task. *Purposeful derivation* refers to controllers that become behaviorally specialized as a result of the CONE cooperative co-evolution process (genotype selection, evaluation, regulation and recombination mechanisms), and in response to task and environment constraints.

NE is the evolution of ANNs using evolutionary algorithms [194], and has been demonstrated as being beneficial in a disparate range of tasks such as robot arm control [102], computer processor unit design [56], and rocket control [59]. NE methods that co-evolve ANN functional units, such as neurons [103] or complete ANN controllers [136], have shown promising results when applied to non-Markovian tasks, tasks that are continuous, require memory, have high-dimensional state spaces, or tasks that are neither effectively addressed via pure evolutionary or neural computation methods [194].

NE methods have been successfully applied to a disparate range of collective behavior tasks that include multi-agent computer games [161], [160], [21], RoboCup soccer [184], pursuit-evasion games [197], [140], and cooperative transport [121] and coordinated movement [141], [13] in simulated multi-robot systems. Such research examples have demonstrated that cooperative co-evolution architectures are effective as a means of encouraging a set of specialized and complementary behaviors. A key advantage of applying NE to controller design in collective behavior systems is that details about how the task is to be solved does not need to be specified *a priori* by the system designer. An analytical model or otherwise formal specification of the task is not required. Rather, NE and a simulator are used to derive, evaluate and adapt collective behavior solutions for a given task. Such simulators evaluate ANN controllers using a fitness ranking of all controller behaviors derived as potential solutions.

CONE extends previous prevalent cooperative co-evolutionary methods that adapt sets of ANN controllers [102], [136], [56], where such methods are naturally amenable to solving collective behavior tasks. CONE regulates collective behavior dynamics in order to engineer specialization[5] for a set of ANN controllers that are given a collective behavior task. Previous methods have been applied to derive ANN controllers for the purpose of solving (typically single agent) tasks, where controllers have minimal *a priori* knowledge of how to optimally solve the task. However, current state of the art controller design methods (when applied to collective behavior tasks), do not utilize emergent specialization as a problem solving mechanism. Hence, when current controller design methods are applied to solve collective behavior tasks that benefit from or require specialization, an optimal task performance (or in some cases: any collective behavior solution) will not be possible.

[5] For a comprehensive review of the role of emergent specialization in collective behavior systems refer to Nitschke *et al.* [113].

1.5 Solving Collective Behavior Tasks

Three collective behavior tasks were selected to test the efficacy of CONE as a controller design method. Each task requires a different form of behavioral specialization in order to solve, and each task is incrementally complex. Each task benefits from controllers that adopt complementary behavioral specializations as part of a collective behavior problem solving process, and each is implemented in a continuous simulation environment. Also, controllers operate with incomplete sensory information, and minimal *a priori* information as to how to solve the task. The selected tasks are delineated in the following.

- *Pursuit-Evasion:* This task requires that a team of simulated pursuer robots derive a *prey-capture behavior*. A prey-capture behavior is a collective behavior formed by multiple pursuer robots, where such a behavior immobilizes a simulated evader robot. The goal of the controller design method is to evolve a set of individual specialized pursuer behaviors, where the interaction of these behaviors results in a prey-capture behavior. The pursuit evasion task evolves between two and six robot controllers.

- *Multi-Rover:* This task requires that a team of simulated autonomous vehicles (rovers) maximize the value of *features of interest* detected over the course of their lifetimes in an unexplored simulation environment. In the multi-rover task, the behavioral roles required to solve the task are *pre-defined*. Each rover controller produces several motor outputs, where each output corresponds to a distinct behavioral role. The goal of the controller design method is to evolve controllers specialized to different behaviors, such that the interactions of these behaviors results in an effective collective behavior. The multi-rover task evolves 20 rover controllers.

- *Gathering and Collective Construction (GACC):* This task requires that a team of simulated robots gather a set of *atomic objects* within an unexplored simulation environment. The robots must deliver these atomic objects in a *predefined order* to a home area. A specific delivery order is required so that *complex objects* are constructed from sets of gathered atomic objects. The behavioral roles required to solve the task are *pre-defined*, meaning that each robot controller produces several motor outputs, where each output corresponds to a distinct behavioral role. The goal of the controller design method is to derive a set of behaviorally specialized controllers, where the interactions of these controllers produces an effective collective behavior. The gathering and collective construction task evolves 30 robot controllers.

These collective behavior tasks were not selected as realistic multi-robot simulations that yield controller design solutions that are transferable to physical multi-robot systems. Rather, the pursuit-evasion, multi-rover, and GACC tasks provide a set of metaphorical examples that demonstrate the efficacy of CONE as a collective behavior controller design method that is able to leverage emergent behavioral specialization as a problem solving mechanism.

2. RELATED WORK

This chapter presents a survey of controller design methods that use behavioral specialization in order to solve collective behavior tasks. This literature review draws attention to behavioral specialization that emerges as a result of collective behavior system dynamics, where such emergent specialization is used as a problem solver. Section 2.1 overviews different types of specialization, collective behavior tasks that require specialization, and methods used to define and measure specialization in biological and artificial collective behavior systems. Section 2.2 delineates several prevalent research examples that model specialization in artificial collective behavior systems. Section 2.3 identifies and describes *Neuro-Evolution* (NE) and cooperative co-evolution as being appropriate for designing controllers that use behavioral specialization in order to solve collective behavior tasks. Section 2.4 draws conclusions from the literature review.

2.1 Specialization

In complex ecological communities, specializations have evolved over time as a means of diversifying the community in order to adapt to the environment [151]. Over the course of evolutionary time, specialization in biological communities has assumed both morphological [180] and behavioral forms [19]. For example, the morphologically specialized castes that have emerged in certain termite colonies [117], and honey bees that dynamically adapt their foraging behavior for pollen, nectar, and water as a function of individual preference and colony demand [26]. The consequence of such specializations is that labor is efficiently divided between specialized castes and individuals for the benefit of accomplishing group tasks. In such a sense, specialization can be viewed as an adaptive mechanism in a complex adaptive system[1].

Many artificial collective behavior systems have used design principles which draw their inspiration from examples of specialization in nature. Such examples include complex ecological communities such as social insect colonies [117], [180], [151], [26], [19], [18], biological neural networks [9], multi-cellular organisms [66],

[1] The benefits of specialization in terms of division of labor have been demonstrated in a broad range of disciplines. For example, in *The Wealth of Nations* [155], Adam Smith described economic specialization in terms of division of labor. Stating that in industrialism, division of labor represents a qualitative increase in productivity, and regarded its emergence as the result of a dynamic engine of economic progress. Smith viewed specialization by workers as leading to greater skill and greater productivity for given tasks, which could not be achieved by non-specialized workers attempting to accomplish those same tasks.

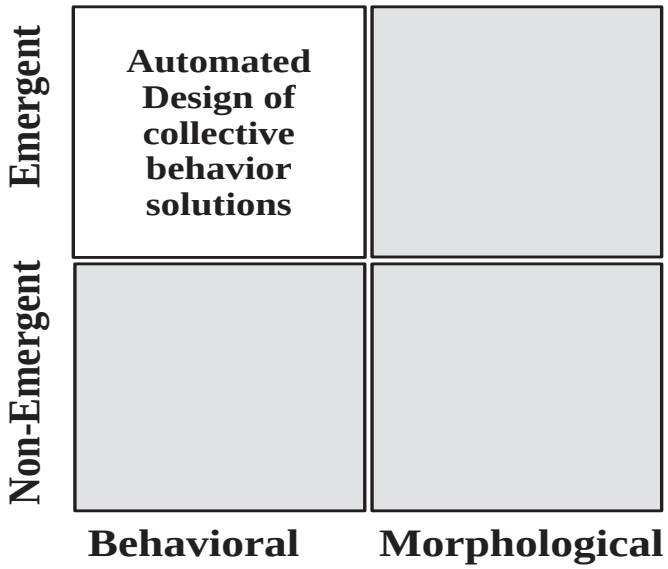

Fig. 2.1: Types of Specialization in Artificial Collective Behavior Systems. The top left-hand side quadrant defines the scope of this thesis. Specifically, adaptive systems that use heterogenous or homogenous design approaches with the aim of deriving emergent behavioral specialization for solving collective behavior tasks.

economies of a nation, companies, and other business organizations [142], [1], [106]. Biologically inspired design principles are especially prevalent in multi-robot [140] swarm intelligence [17] and artificial life systems [108] where it is highly desirable to replicate the success of biological collective behavior systems.

2.1.1 Types of Specialization

Specialization in collective behavior systems has been studied from many different perspectives that include simulated multi-robot systems that use behavioral specialization to enable cooperative transport [122], pursuit-evasion tasks where pursuer agents use complementary behavioral specializations in order to form collective pursuit behaviors [67], RoboCup soccer simulations where soccer agents use different behaviors in order to form effective team behaviors [184], and multi-agent computer games where agents use various behaviors in order to derive collective behavior game playing strategies [21]. Within collective behavior literature, specialization is either studied as an emergent property of the system, or is explicitly pre-programmed into the systems components. With notable exceptions such as [52], 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.

Emergent versus Non-Emergent Specialization

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 accomplish. Such approaches are useful in collective behavior task domains where one does not know, *a priori*, the degree of specialization required to optimally solve the given task [161], [176], [55], [140], [104]. Non-emergent specialization is that which is explicitly prespecified to be a part of the design of system components and global behavior of a system. Such approaches are either static, or use learning algorithms so as to ascertain which type of behavioral specialization, selected from a given set, is most appropriate for solving a given task. Such approaches are useful for solving collective behavior tasks that require specialization, where the degree of specialization required can be sufficiently described *a priori* [7], [11], [12].

Morphological versus Behavioral Specialization

It is possible to further categorize specialization into two distinct classes: *morphological* [96], [202] and *behavioral* [88], [18].

The term *morphological specialization* is applicable to situated and embodied agents, operating in simulated or physical task environments, with embodiment (sensors and actuators) structured so as to yield an advantage at task accomplishment [178], [177], [179]. Examples of morphological specialization include the evolution of optimal arrangements of sensors and actuators in the design of simulated automobiles [96], [202], evolution of agent morphologies and controllers for various forms of motion in simulated environments [153], evolution of physical electric circuits for control [174], and evolving robot morphology for accomplishing different forms of physical motion [90].

Behavioral specialization is applicable to agent behaviors that are advantageous for accomplishing specific types of tasks [11], [12], [125], [124]. An example is the use of machine learning methods that activate certain behaviors with a particular frequency as a response to dynamically arising tasks [55].

2.1.2 Collective Behavior Tasks Requiring Specialization

In the design of collective behavior systems, it remains an open research question as to which tasks are most appropriately solved using specialization. However, there is some agreement amongst researchers that if the task can be naturally decomposed into a set of complementary sub-tasks then specialization is often beneficial for increasing collective task performance [6], [7], [11], [12]. The following list enumerates several collective behavior task domains. Each of these task domains mandates some degree of collective behavior, where specialization is beneficial for improving task performance.

- *Collective Gathering.* Collective gathering is a task domain characterized by the social insect metaphor. Collective gathering tasks seek to emulate

the success and efficiency of social insects in gathering resources. Collective gathering tasks have been studied in the context of both physical multi-robot systems [83], [97], and simulated multi-robot systems [75], [89], as well as artificial life simulations [131], [37].

- *Collective Construction.* Collective construction is a task domain characterized by the social insect metaphor. Collective construction tasks have mainly been studied in the context of artificial life simulations [104], [105], [170]. Collective construction is typically viewed as an extension of the collective gathering task, in that it requires the agents to construct a particular structure with gathered resources. Specialization is typically required for building structures from different types of component resources.

- *Collective Resource Distribution and Allocation.* Collective resource distribution and allocation is research inspired by social insect societies, where the goal is to solve optimization tasks that require an optimal distribution or allocation of resources between agents [19], [18], [172], [27]. Such research studies emergent specialization using behavioral response threshold methods in simulations of agent groups that are implemented within the context of mathematical frameworks.

- *Collective Communication.* Collective communication is a research field that investigates the evolution of language in artificial collective behavior systems. This research employs biologically inspired algorithms designed for the purpose of facilitating a common lexicon in a group of agents. Artificial language evolution has been investigated in artificial life systems [78], [94], [77] as well as in physical multi-robot systems [50], [34], [164]. In the context of language evolution, specialization adopts the form of different agents contributing specialized elements to the lexicon which are specific to their environmental situation.

- *Multi-Agent Computer Games.* Recently, there has been particular research interest in the creation of adaptive interactive multi-agent games, including first-person shooter games [32], [160], as well as strategy games [21], [143], [193] using artificial evolution and learning as design methods for agent behavior. However, the study of specialized game playing behaviors, in teams of agents, has received relatively little research attention. Specialization is beneficial since it is often necessary for teams of agents to formulate collective behavior solutions in order to effectively challenge a human player, where an increasingly complex agent performance is expected as game time progresses.

- *RoboCup Soccer.* A distinct relation to multi-agent game research is RoboCup [79]. RoboCup is a research field dedicated to the design and development of multi-robot systems for the purpose of playing a robotic form of soccer. It is widely recognized as a specific test bed for machine learning algorithms, and engineering challenges [116]. The very nature of the RoboCup game demands the existence of several types of behavioral

specialization, in the form of different player roles. Such behaviors must be complementary and able to interact in such a way so as to produce a desired global behavior. That is, a team strategy that wins the game in a competitive scenario. Certain research has focused on machine learning, evolutionary computation, and NE methods that derive task accomplishing collective behaviors within groups of two or three soccer agents. However, specialized behaviors of individual soccer agents was either specified *a priori* or was derived in simplistic game scenarios [74], [91], [99], [166], [184]. Each of these research examples has been critiqued elsewhere [111].

- *Pursuit-Evasion.* Pursuit-evasion is a collective behavior task that is commonly used within artificial life research to test both non-adaptive (typically game theoretic) and adaptive (typically learning and evolution) methods for agent controller design. The task requires that multiple pursuer agents derive a collective behavior for the capture of one or more evading agents [68], [38]. The investigation of emergent specialization remains a relatively unexplored area of research in the pursuit-evasion domain [129], collective herding variations [16], [140], as well as more traditional predator prey systems [107].

- *Moving in Formation and Cooperative Transportation.* Certain collective behavior research endeavors, mainly in the fields of simulated [13], [122], [141] and physical [150], [7] multi-robot systems, have aimed to model and reproduce various forms of social phenomena that are observable in biological systems [144], [201]. Coordinated movement and cooperative transport is sometimes studied within the context of a gathering task, and has been studied separately in both physical and simulated environments. Cooperative transport is inspired by biological prey retrieval models, which present many examples of the value of specialization, such as the pushing versus pulling behaviors exhibited in stigmatic coordination that allows several ants to transport a large prey [84].

2.1.3 Measuring Specialization

The scope of this research is concerned with methods that facilitate, measure and use emergent behavioral specialization as a problem solving mechanism in collective behavior systems. Hence, the following descriptions are limited to overviews of collective behavioral specialization metrics.

In collective behavior systems research, specialization is often closely associated with, and sometimes synonymous with, heterogeneity in collective behavior systems [10], [140]. Heterogeneity can be hardwired or plastic, and may assume either behavioral [21], [184], [117] or morphological [149], [202], [128] forms. Plastic heterogeneity is when an agent group adapts its degree of heterogeneity as a function of environment and task constraints, where as, hardwired heterogeneity is when the degree of heterogeneity in the group remains static [88]. Given this, several metrics for behavioral specialization that are based on pre-

defined heterogeneity and emergent heterogeneity have been proposed. These behavioral specialization measures assume the following.

- In heterogenous collective behavior systems, a given number of behavioral roles are predefined. Different agents perform different actions (behaviors), and the task explicitly requires a set of complementary roles in order to produce a collective behavior solution. An example of such a collective behavior system is RoboCup soccer [80].
- In homogenous collective behavior systems, a given number of behavioral roles are predefined. All agents are able to perform the same set of actions, and the task does not explicitly require a set of complementary behaviors in order to derive a collective behavior solution. An example of such a collective behavior system is collective gathering [10].

Each of the behavioral specialization measures delineated in the following descriptions, are defined according to the goals and perspectives of the researcher conducting the study. Hence, one can readily find task and environment examples for when a given specialization measure is not appropriate for effectively or correctly capturing the nature of behavioral specialization.

Examples of Group Based Specialization Metrics

Example 1: Li, Martinoli and Abu-Mostafa [89]. The authors define specialization as meaning more than diversity in a group of agents. The authors argue that when behavioral diversity is obtained via an adaptive process such as learning or evolution, noise or an inherent bias in the process can lead to behavioral heterogeneity. However, individual agents in an agent group become specialized when heterogeneity is utilized in order to increase the performance of collective behavior solutions derived by the group. That is, specialization is the part of diversity that is required in order to increase task performance. Hence, the authors define a specialization metric that is applicable to measuring a degree of specialization in a group's collective behavior, and measures the part of diversity that enhances group performance. The specialization metric defined in [89] measures the relationship between collective behavior performance and the degree of behavioral diversity in the agent group.

Example 2: Waibel, Floreano, Magnenat and Keller [176]. The authors use artificial evolution within an agent based simulation in order to investigate the relationship between genotypes and individual behavior and the resulting dynamics of task specialization and colony productivity. In their agent based model, each agent is able to perform one of five tasks at any simulation iteration. An agents propensity to execute a given task is controlled by a stimulus (and corresponding agent behavioral threshold) associated with the given task. That is, only a few workers with low thresholds will perform a task when the task stimulus is very low. As the stimulus level increases, the thresholds of more individuals are exceeded and those workers begin performing the task. Comparative

methods for the artificial evolution of behavioral thresholds are implemented as a means of studying the relationship between genetically determined thresholds, individual behavior, task specialization and overall agent group performance. Although the authors do not explicitly define a behavioral specialization metric, their experimental analysis indicates that agents that evolve a lifetime dedication to one of the five tasks are considered specialized. Hence, specialization in this research operates at the population level, since complements of different specialists are selected in response to task and environment constraints.

Example 3: Solow and Szmerekovsky [157]. The authors examine measures for functional (behavioral) specialization in the context of dynamic linear and non-linear models. Such models describe groups of individuals that interact in order to maximize a utility function. The authors define functional specialization as being related to the amount of time each individual devotes to each activity in a given task environment. The authors subsequently state that for linear problems with one optimal solution functional specialization is indicated if each individual spends 100% of its time on a single activity. For non-linear models with multiple optimal solutions, the authors state that individuals are considered specialized if subsets of individuals dedicate all of their time to solving a set of activities, such that all optimal solutions are achieved.

Examples of Individual Based Specialization Metrics.

The first of the following specialization metrics is taken from biological literature, and the subsequent measures have been used in artificial life, multi-robot systems, and collective behavior research. The first metric is included here since it is a prevalent example of how specialization is measured in biological collective behavior systems. Also, such specialization measures have provided inspiration for the derivation of specialization metrics applicable to artificial collective behavior systems (for example, simulated multi-agent systems in artificial life environments, and physical multi-robot systems in real world environments).

Example 1: O'Donnell and Jeanne [127]. The authors study patterns of individual forager specialization in response to nest damage in three colonies of the tropical social wasp *Polybia occidentalis*. The authors define specialization of an individual as the entropy of the proportions of its activity over a given time period. Specifically, the *Shannon-Weiner information variable H* [29] is used in order to indicate the degree to which a given individual is specialized to gathering a given resource. An individual is considered specialized if it has a low entropy. Low entropy means that an individual focuses on less activities and devotes most of its work time ($> 50\%$) to gathering one particular resource.

Example 2: Gautrais, Theraulaz, Deneubourg and Anderson [55]. The authors extend the specialization metric of O'Donnell and Jeanne [127]. The authors stated that the proportions of activity as described by an entropy measure do not distinguish between an individual that spends half its time on one

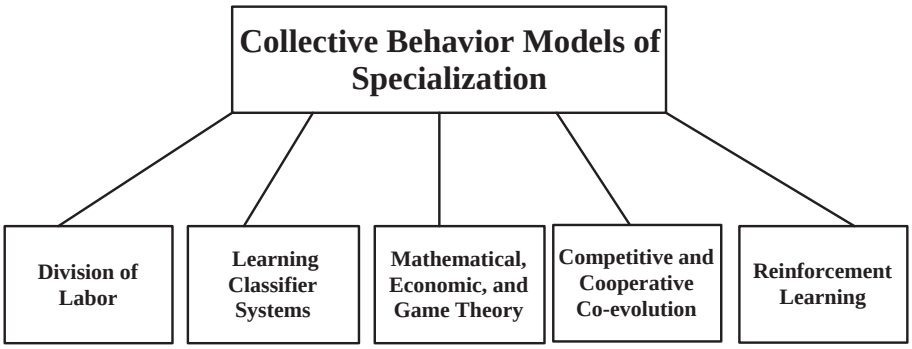

Fig. 2.2: Collective Behavior Models of Specialization. These approaches are appropriate for facilitating specialization in order to solve collective behavior tasks.

task and then switches to another task for the remainder of its lifetime, versus an individual that repeatedly alternates between two tasks. The specialization metric defined by Gautrais and Theraulaz [55] thus uses the frequency with which an agent changes between different tasks. The authors constructed their simulation such that agents would yield a specialization value between zero and one. A value equal to zero indicated that an agent switched between each task with the highest possible frequency and was thus considered a *generalist*. A value equal to one indicated that an agent switched between each task with the lowest possible frequency and thus was considered a *full specialist*. However, the authors did not define a threshold value that indicates when an agent stops being a generalist and becomes a specialist.

Example 3: Murciano, Millan, and Zamora [104, 105]. In this study, the authors applied *Reinforcement Learning* methods to a group of homogeneous agents operating in a discrete simulation environment. A collective gathering task mandated the utilization of specialized behavior, derived at the individual agent level. These individual specializations facilitated the emergence of collective behaviors that achieved a near optimal task performance. The learning task for the agent group was to find, for a given environment, an optimal distribution of specialized agent affinities. An agent was considered specialized if the probability that it collects an object of a given type is close to one, and the probabilities that it collects objects of other types is close to zero. A distribution of affinities was considered optimal for a given environment if this set of affinities resulted in a collective behavior that succeeded in collecting all objects in the environment. However, in their definition of a behavioral specialization metric, the authors did not define a threshold for when an agent is specialized versus when it is not specialized. Instead, agent specialization was defined according to the experimenter's interpretation of agent affinities for collecting objects.

2.2 Collective Behavior Models of Specialization

There is some agreement amongst researchers as to the models of specialization that are appropriate for solving particular collective behavior tasks. The models illustrated in figure 2.2 do not constitute an exhaustive set, but rather several examples that have recently received particular research attention. A particular focus is given to section 2.2.4 given that cooperative co-evolution is an integral part of the CONE method (chapter 3).

2.2.1 Reinforcement Learning Models

Reinforcement Learning (RL) is a problem where an agent must learn behavior via trial and error interactions with a dynamic environment [76]. RL methods have been successfully applied to non-linear and noisy task environments, where the task is complex and there is no *a priori* knowledge as to what constitutes an optimal solution. To solve such tasks, RL methods use statistical and dynamic programming techniques in order to estimate the utility of taking actions given different environmental states. Certain RL models provide periodic feedback signals to agent groups attempting to accomplish a collective behavior task [168]. A reinforcement signal is either local or global. Local reinforcement signals are calculated by, and given to a single agent, or a *caste* [83], upon task accomplishment. Global reinforcement signals are calculated by and given to the entire agent group at the end of a RL trial [89]. The main advantage of RL approaches is that agents are able to effectively operate in complex and noisy environments, with incomplete information.

However, approaches that utilize only a global reinforcement signal, do not typically effectuate specialization in the group, even if task performance could be increased with specialized agents [88]. Approaches that use local reinforcement signals are appropriate for deriving specialized agents, however such approaches suffer from the credit assignment problem [63] which potentially leads to sub-optimal collective behavior solutions. Furthermore, RL models have been highlighted in several studies [25] as performing poorly in multi-agent (collective behavior) tasks, due to a combinatorial explosion of the state space, and problems with scalability of RL algorithms [168]. In practice, RL methods do not scale up to large state spaces and do not perform well in non-Markov tasks where the state of the environment is not fully observable. This is especially the case in complex collective behavior tasks.

2.2.2 Division of Labor Models

The use of behavioral threshold and division of labor models, designed with the goal of optimizing a global task performance, have been investigated within the context of ant-based [36] and resource allocation [18] models. Such models typically utilize feedback signals given to agents of the same caste, in order to encourage emergent specialization for a specific task. Many variations of these models exist, including those that use evolutionary algorithms [169], [176], and

RL methods [104], [105] in order to derive behavioral threshold values.

Such response threshold models represent a very simple, yet powerful, self-regulating feedback system that assigns the appropriate number of agents to different tasks. Studies of such biologically inspired formalizations of behavioral specialization are worthy of future research attention given their applicability to a broad range of optimization tasks including dynamic scheduling and resource allocation. The models of Bonabeau *et al.* [171], Campos *et al.* [27], Gautrais *et al.* [55], and Bonabeau *et al.* [18] eloquently demonstrate how behavioral specialization emerges as a result of self-regulating task assignment and accomplishment, for which there exists a large amount of corroborating biological literature and empirical evidence [31], [30], [145], [36], [173], [126].

Division of labor models are appealing as their evolutionary dynamics and emergent properties can be described with a mathematical representation and the results of such models are thus typically amenable to a mathematical analysis [192]. However, such models are also limited to task domains that can be completely represented via the mechanics of a mathematical model. This limits the contributions of such models to optimization tasks that can be formally represented, or to supporting empirical results in biological literature.

2.2.3 Mathematical, Economic and Game Theory

Linear, non-linear and dynamic models based in mathematical, economic and game theory [8], [157] have many applications for resource assignment problems in business. For example, there are a set of mathematical models for solving task-assignment problems in business. The maximum matching algorithm developed by Edmonds [42], was designed to determine the maximum number of people that can be assigned to tasks in such a way that no person is assigned to more than one task. Thus, it is assumed that each person specializes in performing at most one task. A generalization of this problem is the resource assignment problem [95]. Such models are advantageous in that they have practical applications and their results are amenable to a formal analysis. However, they are limited by their abstract nature, and assume that the task domain can be mathematically or otherwise formally represented.

2.2.4 Competitive and Cooperative Co-Evolution

In nature there is a balance of cooperation versus competition for resources between and within different species as an essential part of the bid for survival[2]. The fitness of a given individual changes over time since it is coupled to the fitness of other individuals of the same and different species inhabiting the same environment. To explain the coexistence of species, Gause [54] proposed that species cannot coexist if they occupy the same niche. The term *niche* refers to a species' requirements for survival and reproduction, such as sufficient resources and habitat conditions. Gause [54] reasoned that if two species had identical

[2] For a literature review pertaining to the topic of cooperation versus competition, specifically emergent cooperation in collective behavior systems, refer to Nitschke [111].

niches, they would attempt to live in the exact same area and would compete for the exact same resources. If this happened, the more competitive species would always exclude its competitors from that area. Therefore, different species must specialize to different niches in order to coexist and increase each individuals chance of survival in its environment. This phenomenon of co-adaptation within and between species is referred to as *co-evolution* [53] and has manifest itself in the form of increasingly complex competitive and cooperative behaviors [134], as well as advantageous morphological (phenotype) changes [191] over the course of evolutionary time[3]. Natural co-evolution has provided a key source of inspiration for the derivation of co-evolutionary algorithms [187]. In co-evolutionary algorithms, artificial evolution and task solving performance is improved via decomposing a given task into a set of composite sub-tasks that are solved by a composite set of artificial species. These species either *compete* or *cooperate* with each other in order to solve the given task.

Competitive Co-evolution Models

Most research conducted in the context of modeling co-evolution has focused upon competitive co-evolution systems. Competitive co-evolution systems implement an *arms race* as a means of producing increasingly effective behaviors that consistently improve via competition [146], [4], [123], [153], [46]. Such competition is typically for limited resources and occurs between multiple species, for example, competition between predators and prey. In the research of Nolfi and Floreano [123] one predator and one prey species are co-evolved, where the predator controller best able to capture the prey robot is selected from the predator species, and the prey controller best able to evade the predator robot is selected from the prey species. The species compete against each other in order to produce predator and prey controllers that effectively attain the highest fitness. Competition for an increasingly high fitness results in increasingly complex behaviors. Such competitive co-evolutionary models have demonstrated that competitive interaction between species aids in maintaining genetic diversity and often results in superior task performance comparative to non co-evolutionary approaches. In the context of competitive co-evolution applied to solving collective behavior tasks, there are relatively few research examples. D'Ambrosio and Stanley [33] applied the HyperNEAT method to solve a multiple predator agent, single prey pursuit-evasion task. Similarly, Nitschke

[3] Numerous examples of emergent phenotype specializations that result from co-evolution have been observed in nature. For example, Grant [62] found that different finch species living on the Galapagos Islands in the Pacific Ocean can coexist if they have traits that allow them to specialize to particular resources. For example, the *Geospiza fuliginosa* and *Geospiza fortis* finch species have the differing morphological traits of varying beak sizes. Beak size is a critical trait since it determines the size of a seed that a finch can eat. Individuals with small beaks eat small seeds, and individuals with large beaks can eat large seeds. *Geospiza fuliginosa* specializes upon smaller seeds because it has more individuals with small beaks. Conversely, *Geospiza fortis* specializes upon larger seeds because it has more individuals with large beaks. Thus, the *ecological niche* [135] occupied by each species differs slightly because the morphological trait of beak size allows them to specialize to a particular seed size.

[109] applied a co-evolutionary NE method in order to evolve increasingly complex collective prey-capture behaviors, as well as predator-evasion behaviors. Luke *et al.* [91] applied genetic programming to co-evolve soccer playing agents that were eventually able to win a game of RoboCup soccer.

Cooperative Co-evolution Models

Another type of co-evolution model is that which harnesses and utilizes cooperation between species as well as competition between individuals of a species. In such models, individuals within a species constitute candidate partial solutions to a complete solution for a given task. Individuals within the same species compete with each other for the role of the fittest individual, where the fittest individual is a candidate partial solution. Individuals selected from each species are then co-evolved in a common task environment, where they collectively form complete solutions. Individuals from each species that worked well together (as complete solutions) are then selected for recombination. The fittest complete solutions are those that best solve the given task.

Potter and De Jong [137] developed a general cooperative co-evolution model implemented within the context of evolutionary algorithms. One particular instantiation of this generalized model was the *Cooperative Co-evolutionary Genetic Algorithm* (CCGA). CCGA has applied co-evolving genetic algorithms to solve function optimization [138] and robot control tasks [139]. CCGA has since been extended and applied to various machine learning tasks [45], [186]. One prevalent instantiation of Potter's generalized cooperative co-evolutionary model is *Enforced Sub-Populations* (ESP) [56], which co-evolves neurons for the purpose of constructing effective *Artificial Neural Network* (ANN) controllers.

Moriarty [102] 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 (as a complete ANN) solved a given task. In the other population, neurons received fitness based upon the number of successful blueprints they participated in.

Dreżewski [40], [41] describes a multi-agent system that implements a cooperative co-evolutionary process that uses two species. The system effectively solves multi-objective optimization tasks via agents locating the *pareto* frontier as a result of co-evolutionary interactions between the two species.

As is the case with competitive co-evolution, there has been relatively little research which applies cooperative co-evolution in order to solve collective behavior tasks. Prevalent examples include the application of ESP to evolving multi-agent game playing strategies [21], the derivation of prey-capture behavior in a team of simulated robots [140], [196], [15], [14], the derivation of multi-agent surveillance behaviors [114], and the evolution of collective gathering and construction behaviors in teams of simulated robots [115].

The advantages of cooperative co-evolution models include versatility and applicability to a broad range of complex, continuous, and noisy tasks. Also, the use of multiple genotype populations provides a natural representation for

many collective behavior tasks, and often effectuates emergent specialized phenotypes. Cooperative co-evolution models differ from related models that use multiple interacting populations, as a means of exploring and exploiting niches in the solution space. A prevalent example is the *island model* [185], [154], which has been used primarily to prevent convergence to poor solutions. Island models do not support interactions between phenotypes in a common task environment. Such interactions are integral to cooperative co-evolution models, and are required for the purposes of co-adaptation of individuals endemic to different genotype populations, as well as the formation of competitive and cooperative relationships in both the genotype and phenotype space.

2.2.5 Learning Classifier Systems

Learning Classifier Systems (LCSs) use adaptive mechanisms based on genetic algorithms and reinforcement learning in order to derive a population of stimulus-response rules (classifiers) that appropriately represent a given problem space [152]. Individual rules in the population work together to form complete solutions to a given task. A micro-economy model is employed to manage the interactions of rules, such that the rules place bids for the purpose of becoming active. Credit is assigned to rules using the bucket brigade algorithm [72], which passes a value back along a rule activation chain to rules that aid in solving the task. The complex dynamics of the micro-economy model results in emergent problem decomposition and the preservation of rule diversity. LCSs were initially created to solve problems such as the *Animat problem* [188]. Many derivatives and variations of LCSs have been proposed since the inception of LCSs [71]. These include, Michigan style LCSs [73], Pittsburgh style LCSs [156], Extended Classifier Systems (XCS), Anticipation based LCSs (ALCSs) [165], and Zeroth-level Classifier Systems (ZCS) [189]. The adaptive mechanisms employed by LCSs use processes of generalization and specialization in order to produce a population of classifiers that yield effective solutions to given tasks. That is, an LCS evolves a population of classifiers, such that an effective balance between sufficiently general and sufficiently specialized classifiers is derived. Generalized classifiers are those that are applicable to many task environment conditions. Specialized classifiers are those that are applicable to specific task environment conditions.

The results of several studies such as Bull *et al.* [24], Potter *et al.* [139], Bull [23], and Hercog and Fogarty [69] have elucidated that LCSs are inherently appropriate for solving collective behavior tasks [23]. Though not explicitly stated, the results of such research indicates that the nature of LCSs provide the potential for effectuating behavioral specialization as a means of attaining collective behavior solutions. Bull [23] applied a ZCS [189] as an adaptive mechanism in a multi-agent trading simulation. Results indicated that the approach improved the efficiency of individual traders and thus the efficiency of the artificial market. Bull *et al.* [24] describe the use of Pittsburgh style [156] classifier systems for the control of a quadrupedal robot, where each leg is represented and controlled by a different classifier. Potter *et al.* [139], used a Pittsburgh

style LCS implemented in the context of a *Cooperative Co-evolutionary Genetic Algorithm* (CCGA). The CCGA was used to co-evolve two simulated robots for a food gathering task, where each robot was controlled by a classifier system and represented as a separate species. Results indicated faster convergence to an optimal solution using the CCGA, comparative to non co-evolutionary approaches. Hercog and Fogarty [69] presented a multi-agent system that learned, using an XCS, to solve the *El Farol* bar problem. Emergent behavior was observed as part of the XCS learning and multi-agent system dynamics. Specifically, a vacillating agent emerged which did not attain a reward for itself, but enabled other agents to increase their task performance.

2.3 Neuro-Evolution (NE)

NE is the artificial evolution of *Artificial Neural Networks* (ANNs) [70] using evolutionary algorithms [43]. NE operates via searching through a space of ANN behaviors for a behavior that adequately solves a given task. NE employs a phylogenetic[4] instead of a ontogenetic[5] adaptive process. A population of solutions (ANNs) are modified via the repeated recombination and mutation of a set of genotypes representing the fittest ANNs. This artificial evolution process continues until a sufficiently fit ANN (solution to the given task) is found. The evolutionary algorithm replaces ontogenetic ANN learning algorithms (for example, supervised learning) and facilitates adaptation via searching a space of solutions (encoded ANNs) directly. NE combines and utilizes the advantages of both ANNs and evolutionary algorithms. That is, NE combines ANN advantages of generalization, function approximation, and temporal capabilities, with the efficient parallel search capabilities of evolutionary algorithms. Given these advantages, NE overcomes problems inherit in ANN supervised learning processes, given that the weight training process is replaced with an unsupervised artificial evolution process. In such a process, an optimal set of weights is defined by a fitness function, and the training task is defined by the environment in which artificial evolution occurs [194]. Furthermore, given that NE is capable of searching for an ANN behavior, NE methods are beneficial in a disparate range of complex control problems with continuous and high-dimensional state spaces, as well as complex tasks that require memory [58]. NE has been demonstrated as an effective alternative to reinforcement learning methods when applied to control tasks such as pole balancing and robot arm control [103]. Also, NE is most appropriately applied to tasks that are neither effectively addressed via pure evolutionary or neural computation methods [194].

2.3.1 Conventional Neuro-Evolution Methods

Given the multitude and diverse range of NE methods designed to evolve various ANN properties such as connection weights, network architecture, and learn-

[4] Phylogenetic: the development or evolution of a particular group of organisms [39].
[5] Ontogenetic: the development of an individual organism from embryo to adult [39].

ing rules [60], [194], there is consequently no canonical *Conventional Neuro-Evolution* (CNE) method. Figure 2.3 illustrates an example CNE method which uses one population of N genotypes. From this population, each genotype is systematically selected, decoded into an ANN, evaluated in the task environment and assigned a fitness. During the ANN's *lifetime*, it receives sensory input from its environment, which is propagated though the network so as to produce a motor output that affects a particular change in the environment. At the end of its lifetime (or period of evaluation) the ANN is assigned a fitness according to its performance with respect to accomplishing the task. This fitness is then assigned back to the corresponding genotype. The evaluation and assignment of fitness to all ANNs (genotypes) typically constitutes one generation in the evolutionary process of CNE. At the completion of each generation, the fittest genotypes are recombined so as to propagate a new population of genotypes.

Figure 2.4 illustrates an example CNE method that uses one genotype population operating in a collective behavior task (a task that requires more than one ANN in order to accomplish). In this case the genotype space consists of N genotypes, where M genotypes are selected from the genotype space and decoded into M ANNs. One genotype encodes all the parameters of an ANN, and one ANN represents one phenotype. The M ANN controllers are evaluated together in the task environment. This evaluation is then passed back in the form of fitness values to the genotypes that encoded the ANNs. Hence, the ANNs that perform well in the task have their corresponding genotypes recombined so as a new *fitter* set of M ANNs are generated and placed in the task environment at the next generation of the CNE algorithm.

Either a *homogenous* or *heterogenous* approach can be used in order to construct a group of ANNs given the selection of M genotypes from the population. In order to construct a homogenous group, one of the fittest genotypes is selected from the population, and copied M times, and subsequently decoded into a group of M ANN clones. Alternatively, in order to construct a heterogenous group, M of the fittest genotypes are selected from the population, and subsequently decoded into a group of M different ANNs.

There are several disadvantages of CNE approaches. First, as is the case with evolutionary algorithms [43], as the evolutionary process converges upon a given ANN solution, diversity in the genotype population is often decreased to the point where sub-optimal solutions are yielded and fitness progress of ANN solutions stagnates. Also, given that ANNs are necessarily encoded as genotypes, where genotypes are recombined and mutated as part of the evolutionary process, the competing conventions problem must be resolved [163], [194]. There are often different and incompatible encodings for the same phenotype (ANN behavior) solution. Hence, in order for genotype recombination operators, and genotype to phenotype encoding to function properly, a mechanism that matches only compatible genotypes for recombination and resolves differences in encodings for a given ANN, is required. Also, as the complexity or scale of a given task increases, the number of parameters that need to be optimized simultaneously increases exponentially.

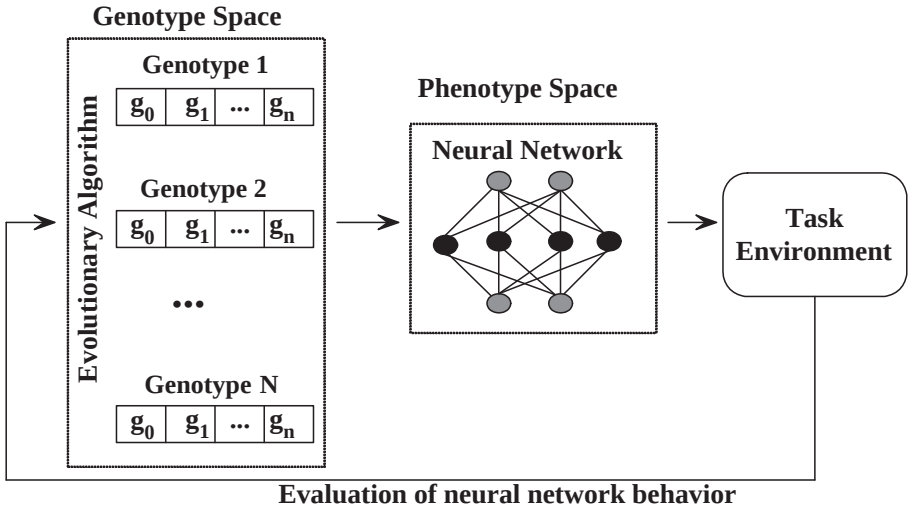

Fig. 2.3: Conventional Neuro-Evolution (CNE). An example CNE method using a single genotype population. In each genotype, $[g_0, g_n]$ denotes the n constituent genes.

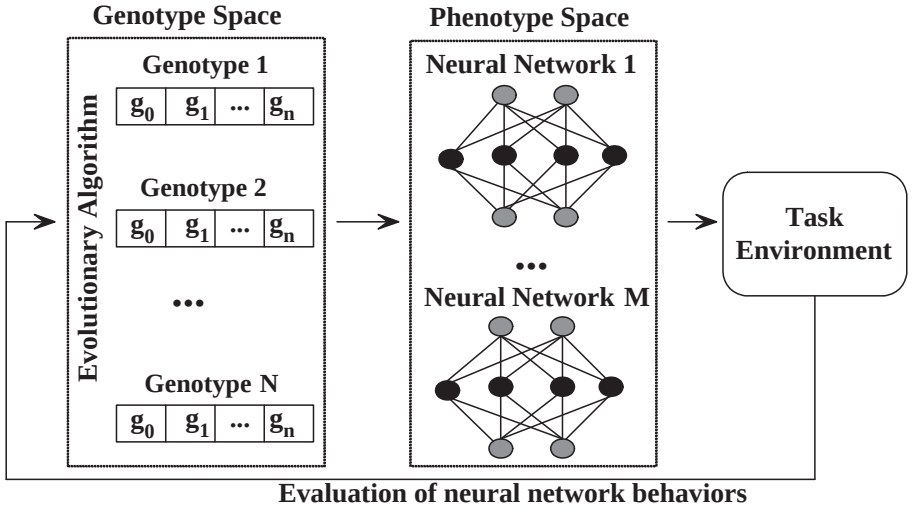

Fig. 2.4: Conventional Neuro-Evolution (CNE) in a Collective Behavior System. An example of CNE operating in a collective behavior task, using a single genotype population. Genotype selection may form either a homogenous or heterogenous group of ANNs. A homogenous group is derived via cloning one selected genotype in order to derive a set of identical ANNs. A heterogenous group is derived via selecting and decoding a number of genotypes corresponding to the number of ANNs in the group. In each genotype, $[g_0, g_n]$ denotes the n constituent genes.

2.3.2 Encoding Schemes for Neuro-Evolution

The key difference between various NE methods is how ANNs are encoded as genotypes. The scheme used to encode an ANN as a genotype that can be manipulated by an evolutionary algorithm and the subsequent decoding of the genotype into a phenotype (ANN behavior) is either *direct* or *indirect*.

In direct encoding, ANN parameters are represented explicitly in the genotype as binary or real numbers that are mapped directly to the phenotype [57], [102]. An indirect encoding scheme operates at an abstract level. An example of an indirect encoding scheme is the evolution of a set of rules that indirectly encode an ANN's parameters. Such parameters include network topology, learning rules, as well as connection weights [194]. Direct encoding schemes have the advantage of simplicity in encoding and subsequent decoding schemes. The disadvantage of direct encoding schemes is that they must be designed for particular types of task environments. If the task environment is complex and high dimensional then large genotype populations are requisite for convergence to optimal or near optimal solutions. Indirect encoding schemes have the advantage that the size of the genotype space can be adapted as a function of the complexity of the task environment. If large genotypes are not required for solving a task, then they will not be specified as such. The capability of indirect encoding to adapt the structure of the genotype space allows for sufficient variability in the phenotype space, meaning that a greater spectrum of solutions is possible.

The disadvantage of indirect encoding schemes is that the rules specifying the growth and development of genotype encodings must be sufficiently general and powerful so as to allow for the development of genotype encodings appropriate for a given task environment. The method used to encode ANN parameters as a genotype significantly influences the dimensions of the solution space as well as the types of phenotypes that may be decoded and evaluated in the task environment. ANN parameters that are usually encoded as genotypes include the connection weights between neurons, the number of hidden layer neurons, network topology, and the learning rate of an ontogenetic learning algorithm.

2.3.3 Neuro-Evolution and Cooperative Co-Evolution

Prevalent examples of cooperative co-evolution systems implemented within the context of NE include *Symbiotic Adaptive Neuro-Evolution* (SANE) [102], CCGA [136], and the ESP [56] method. SANE and ESP (and one particular case study using CCGA [136]) combine cooperative co-evolution with NE so that the search for optimal ANN behaviors is enhanced via a cooperative co-evolution process. In the context of evolving ANN controllers, figure 2.5 presents the general scheme for cooperative co-evolution systems. In methods such as CCGA, evolution operates at the level of a complete ANN, where as in methods such as SANE [102], and ESP (section 2.3.3), the evolutionary process operates at the neuron level. In the case of the latter type of method, a phenotype is the function performed by a single hidden layer neuron. This is presented as the bottom (neuron) layer in figure 2.5. In the case of CCGA (section 2.3.3) the

phenotype is the behavior exhibited by a complete ANN. This is presented as the middle (agent) layer in figure 2.5. It is also conceivable that an evolutionary process may operate at the level of agent collectives. That is, where an individual genotype represents at least two ANNs. This is presented as the top (collective behavior) layer presented in figure 2.5. This layer does not illustrate a single genotype representing n ANNs.

ESP: Enforced Sub-Populations

This section describes a NE method known as *Enforced Sub-Populations* [56]. In ESP, genotypes are represented as individual neurons instead of complete ANNs. Each genotype is encoded as a string of values that represent input and output connection weights of a hidden layer neuron. ESP segregates the genotype space into u sub-populations, where each sub-population evolves each hidden layer neuron in an ANN. ESP is applicable to the evolution of any type of ANN that consists of at least one hidden layer [20]. Figure 2.6 illustrates one generation of the ESP evolutionary process. The hidden layer of an ANN is constructed via selecting one of the fittest neurons from each of the u sub-populations. The ANN is then evaluated in a task environment.

Multi-Agent ESP [197] 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 ANNs are constructed as in ESP. This process is repeated n times for n ANNs, which are collectively evaluated in a task environment.

Disadvantages of ESP / Multi-Agent ESP

The main disadvantage of ESP is that genotypes within different sub-populations evolve specializations at the expense of sub-population diversity. This problem is especially prevalent in tasks that necessitate the use of ANN controllers that exhibit different specialized behaviors. ESP tends to evolve ANN behaviors suited for one particular niche in the problem space [56]. In continuous, noisy and non-linear tasks such as rocket control [59], optimal task performance is only achievable via co-adaptation of optimally performing specialized neurons. However, the convergence to sub-optimal neurons is highly probable in sub-populations which lack a sufficient level of genotype diversity [56]. This in turn results in the derivation of ANN controllers which do not take full advantage of specialization and achieve a sub-optimal task performance.

However, the application of Multi-Agent ESP by Yong and Miikkulainen [197] demonstrated that even though genotype diversity was lost as an ANN controller converged to one specialized behavioral role, cooperation between multiple ANN controllers (populations) emerged as a result of different ANN controllers converging to complementary behavioral roles. Derivation of these behavioral roles was driven by a cooperative co-evolution scheme that rewarded effective collective behavior. However, such collective behavior emerges without any direction of evolution based on the success of behavioral interactions in the

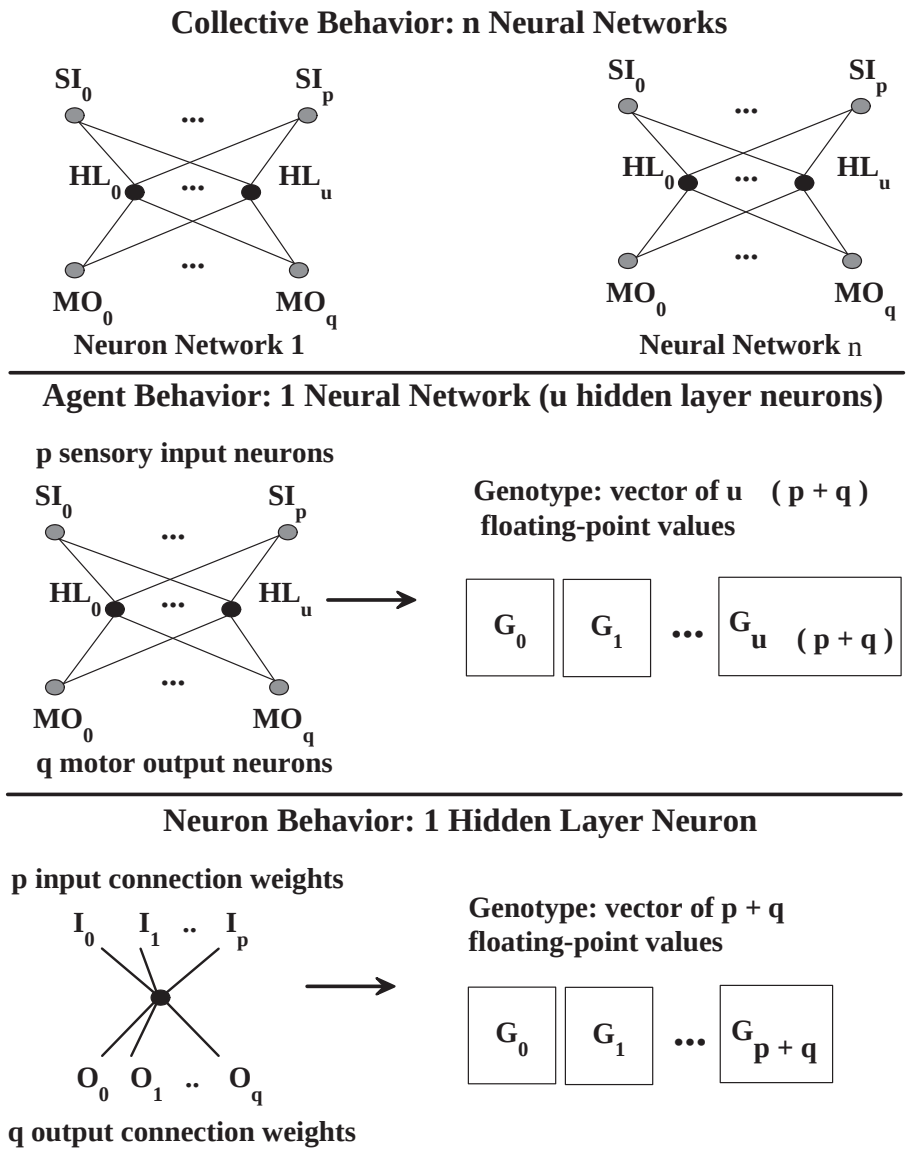

Fig. 2.5: Cooperative Co-evolution Models. A general scheme where evolution operates at the neuron (bottom layer), at the agent (middle layer), or at the collective behavior level (top layer).

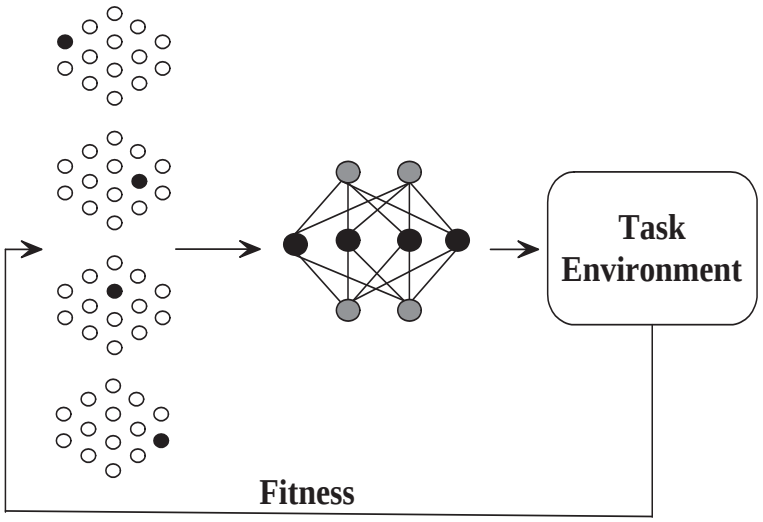

Fig. 2.6: Enforced Sub-Populations (ESP). The genotype (neuron) population is segregated into sub-populations. An ANN is formed via selecting a fittest neuron from each sub-population. Figure adapted from Gomez [57].

task environment. Hence, the optimality of evolved behavioral specializations and thus the composite collective behavior, remains unclear. Also, there was no sharing of potentially useful genetic information between populations in Multi-Agent ESP (and between sub-populations in ESP). This lack of information sharing means that genotype solutions from one population cannot contribute to genotype solutions in another population. Furthermore, redundant solutions need to be independently discovered within each population, resulting in a slow or otherwise inefficient search [56].

CCGA: Cooperative Co-evolutionary Genetic Algorithms

The *Cooperative Co-evolving Genetic Algorithm* (CCGA) [136] uses genetic algorithms to co-evolve multiple partial solutions that interact in order to form complete solutions to a given task. CCGA is represented by several composite genotype populations. Each of these populations is termed a *species*, and each species is genetically isolated meaning that genotypes can only be recombined with other genotypes within the same species. At each generation of the evolutionary process, genotypes from each species are instantiated as partial solutions, which are evaluated in a common task environment. Genotypes (selected from different species) are evaluated based upon how well their corresponding phenotypes (partial solutions) interact in a given task environment for the purpose of finding a complete solution. Figure 2.7 illustrates the co-evolution model used by CCGA. Although this particular illustration shows only two species, the model may comprise n species [138].

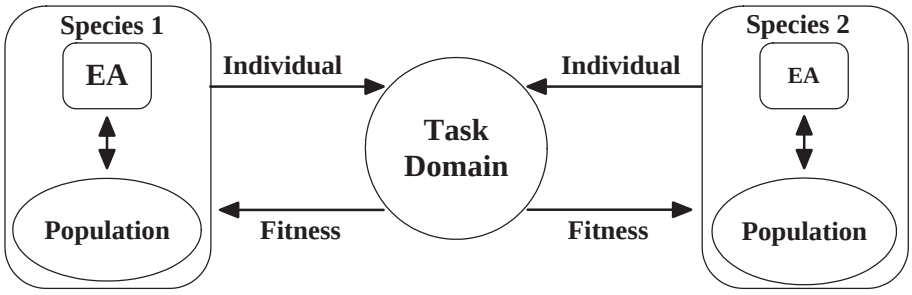

Fig. 2.7: Cooperative Co-evolutionary Genetic Algorithm (CCGA). CCGA allows for n species to be co-evolved. Figure adapted from Potter [143].

Disadvantages of CCGA

The recombination of genotypes solely within the same species removes the possibility for sharing potentially beneficial genetic information between species. This also increases the likelihood that genotypes in a given species will converge to a sub-optimal, yet specialized, part of the genotype space. For example, different species in CCGA can represent the same specialization, as may be necessitated in certain tasks. However, different species cannot share information and therefore must find the same specializations independently. Also, to date, only genetic algorithms and evolutionary strategies have been used as the adaptive mechanisms within each species. Different evolutionary algorithms executing simultaneously in different species have not been tested [138].

2.4 Conclusions

This chapter over-viewed two important research for automated controller design. First, the chapter delineated several approaches for facilitating emergent behavioral specialization in collective behavior systems. Second, the chapter outlined NE as an effective approach for deriving robust controllers that operate with incomplete sensory information in continuous and noisy simulation environments. Controller design methods that combine cooperative co-evolution and NE were identified as being appropriate for facilitating emergent specialization in collective behavior systems. The heterogenous multi-population nature of cooperative co-evolution is a natural approach for encouraging the evolution of behavioral specialization in collective behavior systems. The main purpose of this chapter was to elucidate that controller design methods that use emergent behavioral specialization as a problem solver are currently lacking. The research topic of this thesis thus investigates an intersection between three research areas (NE, cooperative co-evolution, and behavioral specialization) in order to propose and derive a method that will close this gap. The following chapter describes *Collective Neuro-Evolution* (CONE) as a method that combines and extends approaches from these research areas.

3. COLLECTIVE NEURO-EVOLUTION METHOD

This chapter introduces the *Collective Neuro-Evolution* (CONE) method. CONE is designed to develop controllers for agent teams. The agent controllers are *Artificial Neural Networks* (ANNs) and the development method is artificial evolution, hence the name: CONE.

Given its application to a collective behavior task, CONE purposefully facilitates emergent specialization for the benefit of problem solving during controller evolution. Emergent specialization is used in order to increase task performance or to solve tasks that could not otherwise be solved without specialization. Such an approach is currently lacking in controller design methods for collective behavior tasks. *Emergent specialization* refers to behavioral specialization that results (emerges) from the evolution (development) of an individual controller, or emerges from the interactions of multiple controllers over the course of an evolutionary process. Given that emergent behavioral specialization is beneficial to, and often necessary for solving collective behavior tasks, CONE contributes to controller design research in fields such as multi-robot systems.

CONE evolves n populations of genotypes which represent n sets of neurons. These neurons are used to construct n ANN controllers that are collectively evaluated in a common task environment. All ANN controllers necessarily have a single hidden layer fully connected to input and output layers. An example of a CONE setup using three populations is presented in figure 3.1. Unlike related methods, which include SANE [102], CCGA [136], and ESP [56], CONE implements genotype and behavioral specialization difference metrics to regulate genotype recombination *between* and *within* populations. These difference metrics control recombination and direct evolution based upon genotype similarities and the success of behavioral specializations exhibited by ANN controllers.

The remainder of this chapter is organized as follows. First, section 3.1 describes the representation used by CONE, which includes the population architecture and genotypes. Second, section 3.2 describes the role of specialization in CONE, which includes a definition of behavioral specialization and a metric for calculating behavioral similarity between controllers. Third, section 3.3 describes the process used for evaluating and assigning fitness to ANN controllers and genotypes. Fourth, section 3.4 describes the method of parent selection, as well as the recombination and mutation operators. Fifth, section 3.5 describes mechanisms used to adapt algorithmic parameters, which includes the adaptation of genetic similarity and specialization similarity thresholds. Finally, section 3.7 presents an overview of the entire Neuro-Evolution (NE) process employed by CONE, and section 3.9 presents the chapter conclusions.

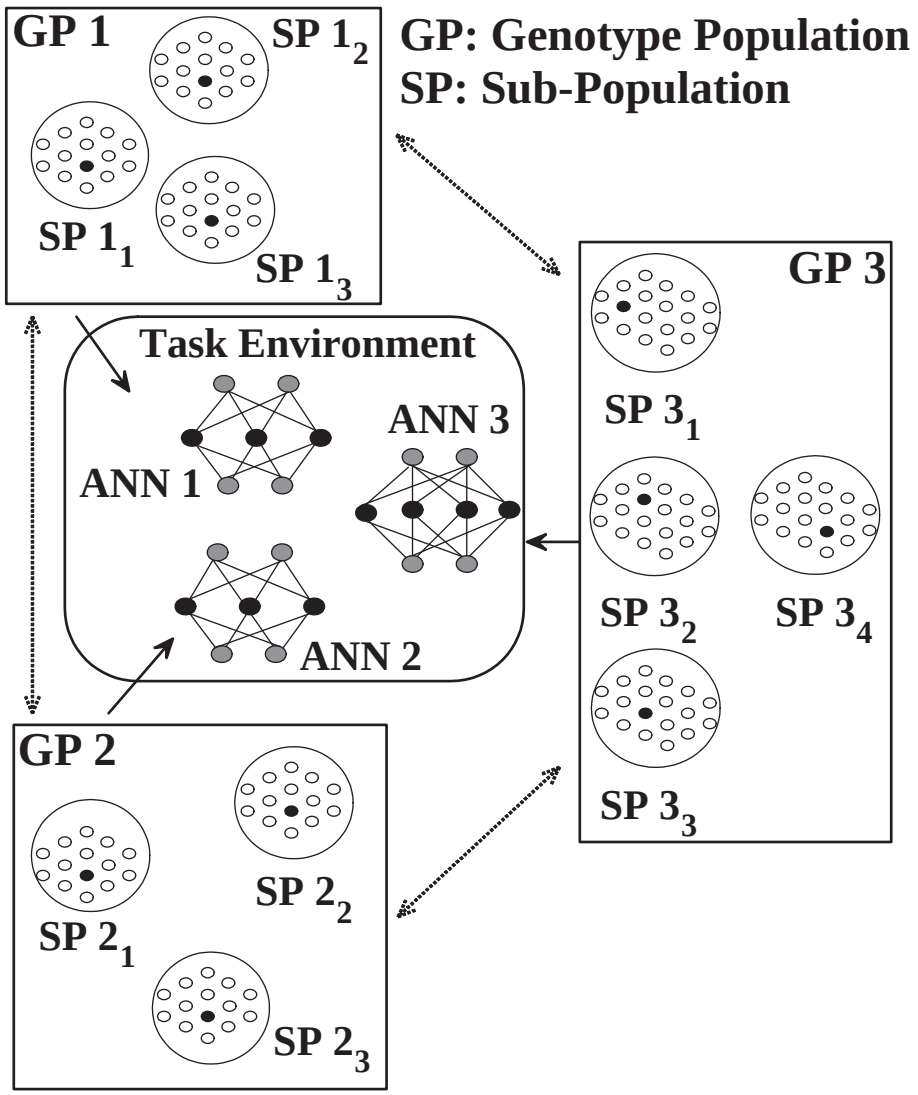

Fig. 3.1: Collective Neuro-Evolution (CONE) Example. Three ANN controllers are derived from three populations and evaluated in a collective behavior task. Double ended arrows indicate self regulating recombination occurring between populations.

3.1 Representation

3.1.1 Population Structure

As with related NE methods [138], [57], CONE segregates the genotype space into n populations for the purpose of developing n ANN controllers. ANN_i $(1 \leq i \leq n)$ 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 throughout the evolutionary process of CONE and the number of hidden layer neurons is adaptive. CONE mandates that ANN_i 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 3.1 illustrates an example of the population structure of CONE. Three populations for deriving three controllers in a collective behavior task are illustrated. Accordingly, ANN-1 and ANN-2 (derived from genotype populations 1 and 2, respectively) consist of three hidden layer neurons, whilst ANN-3 (derived from genotype population 3) consists of four hidden layer neurons.

The CONE evolutionary process is driven by mechanisms of cooperation and competition within and between sub-populations and populations. There is competition between the genotypes of a sub-population, given that the genotypes compete for a place as a fittest neuron in the hidden layer of a fittest controller. Also, there is cooperation between sub-populations, given that a fittest genotype is selected from each sub-population (within a given population) in order to participate in forming a controller. Furthermore, there is cooperation between the controllers derived from each population, given that n controllers must cooperate in order to accomplish a collective behavior task.

3.1.2 Genotypes

A genotype in CONE corresponds to a hidden layer neuron which is fully connected to all the input and output neurons of ANN_i. All ANN controllers have the same number of input and output neurons. Genotypes have a fixed length, where the number of genes in a genotype corresponds to the number of parameters (number of connection weights in the corresponding hidden layer neuron), plus a tag. The first gene is the genotype's tag which specifies which sub-population (which position in the hidden layer of ANN_i) the genotype is assigned to. The first gene is not subject to the evolutionary process. All genotypes in sub-population u_i have the same tag. All other genes represent input and output connection weight values. There is a one-to-one mapping between any genotype and its corresponding neuron.

Genotype a is defined as follows.

$$a =< a_0, a_1, \ldots, a_k > \tag{3.1}$$

Where, a_0 is the genotype's tag, and a_i $(1 \leq i \leq k)$ denotes a neuron connection weight, where each connection weight is a floating point value normalized

in order to be within the range: [-1.0, +1.0]. Figure 3.2 presents an illustration of genotype a and the corresponding hidden layer neuron.

$$b =< b_0, b_1, \ldots, b_k > \tag{3.2}$$

CONE also requires a measurement for the similarity between two genotypes a and b, where equation 3.1 defines a, and equation 3.2 defines b.

A *Genetic Distance* (GD) between a and b is defined by equation 3.3.

$$GD(a, b) = \frac{\sum_{i \in \{1,\ldots,N\}} |a_i - b_i|}{N} \tag{3.3}$$

Where, N is the length of a genotype.

For the purposes of measuring genotype similarity, a *Genetic Similarity Threshold* (GST) is defined, where GST is within the range: [-1.0, +1.0]. Genotypes a and b are considered to be similar if: $GD(a, b) < GST$.

3.2 Specialization

An integral part of CONE is defining behavioral specialization exhibited by each ANN controller, and measuring the similarity of specializations between multiple controllers. This section describes the calculation of a specialization distance between two controllers, and the method used to determine if two controllers are considered to have a similar behavioral specialization.

3.2.1 Degree of Specialization

CONE requires a behavioral specialization definition for ANN controllers. Controllers select from activating at least two different motor outputs, in order to perform at least two different actions. Hence, a specialization metric that accounts for the partitioning of a controller's work effort among different actions is required. Given this, an extension of the behavioral specialization metric for individual controllers defined by Gautrais *et al.* [55] is selected. The *degree of behavioral specialization* (S) exhibited by an individual controller is defined by the frequency with which it switches between executing distinct motor outputs (actions) during its lifetime. Extensions to the metric of Gautrais *et al.* [55] are described in the following.

1. The metric defined by Gautrais *et al.* [55] allowed for negative values, and agent simulations were concocted such that only positive specialization values were produced. This extension restricts the degree of specialization calculated for individual behavior to a value in the range: [0, 1].

2. A *Specialization Similarity Threshold*. This threshold is static and determines if a controller's behavior is *specialized* or *non-specialized*.

3. A *degree of behavioral specialization* can be measured for a controller team as well as for an individual controller.

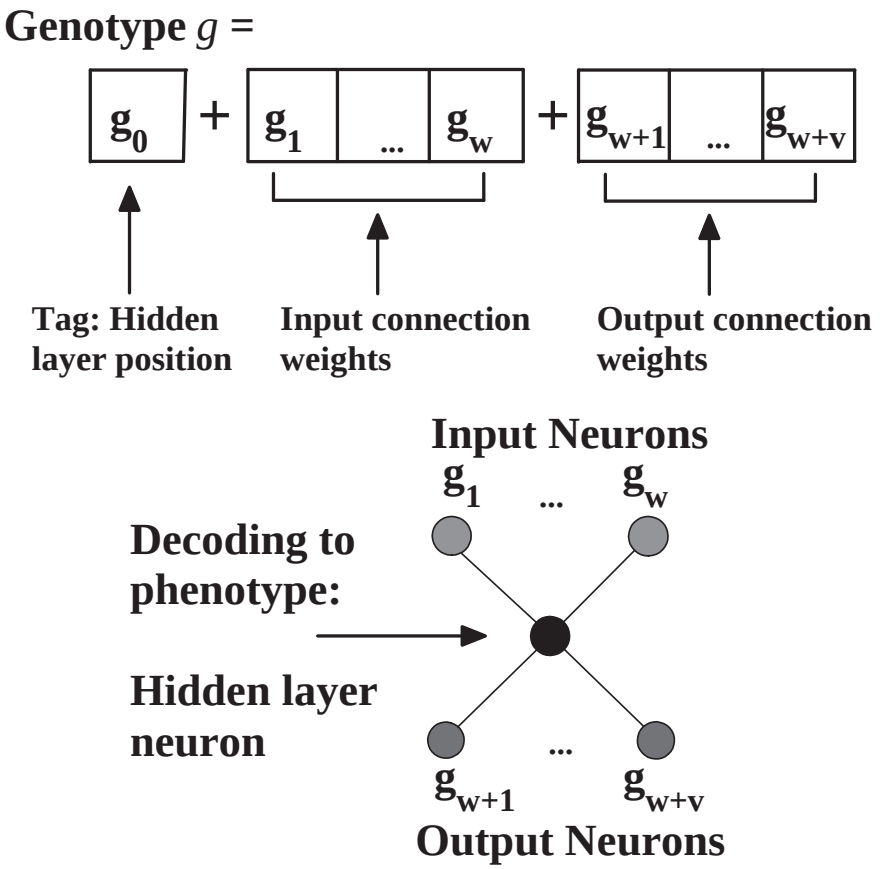

Fig. 3.2: **CONE Genotype**. A genotype is an encoded hidden layer neuron that participates in a given ANN controller. There is a direct (one-to-one) mapping between a genotype and the corresponding neuron. The genotype consists of w genes that correspond to the neuron's input connection weights, and v genes that correspond to the neuron's output connection weights. A tag (t) at the beginning of the genotype (the first gene) specifies the position in the controller's hidden layer, and hence the sub-population number, to which the genotype is assigned.

For any given controller, S is calculated as the frequency with which it switches between each of its v actions. The calculation of S is presented in equation 3.4, where A denotes the number of times the controller switches between different actions, and N denotes the total number of possible action switches.

$$S = \frac{A}{N} \tag{3.4}$$

The calculation of S assumes that any given controller has v distinct motor outputs, where one motor output activation denotes one action as being executed. 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. A value of S 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 the controller spends half of its lifetime switching between different actions.

3.2.2 Specialization Distance Metric

Given two controllers, ANN_i and ANN_j, a specialization distance (SD) is defined as follows (equation 3.5).

$$SD(ANN_i, ANN_j) = |S(ANN_i) - S(ANN_j)| \tag{3.5}$$

3.2.3 Degree of Specialization Threshold

In order to label an individual controller as *specialized* or *not specialized*, specific values need to be assigned to the *Specialization Similarity Threshold* (SST).

- If $S \geq 0.5$ then a controller is labeled as *non-specialized*.
- If $S < 0.5$ then a controller is labeled as *specialized*.

Given that a controller is defined as 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 simply labeled as *non-specialized*. If a set of controllers are defined as being specialized, then it is possible to group controllers according to their specialization to action x. A set of controllers specialized to executing the same action is defined as a *caste* [83].

3.2.4 Specialization Similarity Threshold

ANN_i and ANN_j are considered to have a similar behavioral specialization if both are measured as being within a *Specialization Similarity Threshold* (SST), where SST $\in \{0, \ldots, 1\}$. That is, if the condition in equation 3.6 is true.

$$SD(ANN_i, ANN_j) < SST \tag{3.6}$$

3.3 Evaluation

This section describes the method used to evaluate and assign a fitness to each genotype within each population. Given a genotype that is selected to be evaluated, the following evaluation process is followed. In the following process description different fitness types are referred to. The term *elite portion* refers to a fittest portion of genotypes in a population.

Type A Fitness: Attained in the first part of the evaluation process which evaluates all genotypes in all populations (section 3.3.1). Type A fitness values are applied to genotypes, individual controllers, and groups of controllers.

Type B Fitness: Also referred to as controller *utility*. Controller utility is used to rank and select a controller from the elite portion of each population. Type B fitness values are calculated as the sum of all fitness (type A) values of all genotypes used to construct the hidden layer of a controller.

Type C Fitness: Type C fitness values are attained in the second part of the evaluation process which evaluates an elite group of controllers selected from the elite portion of each population (section 3.3.2). Type C fitness applies to genotypes, individual controllers, and groups of controllers. In preparation for the parent selection process, type C fitness values overwrite the fitness (type A) values previously calculated for all genotypes.

Overview of the Evaluation Process

1. *Evaluate all Genotypes.* From a given population, construct a controller which includes in its hidden layer a genotype g that is to be evaluated.
2. For each of the other n-1 populations, construct n-1 other controllers via randomly selecting genotypes from each of the other n-1 populations.
3. Test the n controllers in a task to attain a fitness (type A) value.
4. Assign this fitness (type A) value to g.
5. Repeat steps 1 to 4 until all genotypes in all sub-populations of all populations have been assigned a fitness (type A) value.
6. *Evaluate Elite Controllers.* From each population construct a controller that maximizes fitness type B. Fitness type B is defined as the simple sum of the fitness (type A) of each genotype that comprises the hidden layer of this controller.
7. For each population test each controller derived from the elite portion with a controller constructed from the elite portion (randomly selected) of each other population.
8. Run these elite groups of controllers in task simulations in order to get a fitness for each group. This group fitness (type C) is passed down to each genotype in the hidden layer of each controller. This fitness (type C) then

overwrites the old fitness (type A) of each genotype in the elite portion of each population.

9. Repeat steps 6 to 8 until all genotypes in the elite portion of all populations have been assigned a fitness (type C) value.

Genotype and elite controller evaluation is elaborated upon in the following.

3.3.1 Evaluate all Genotypes

Given that P_i is a population of m genotypes ($\{g_0, \ldots, g_m\}$), u_i is the number of hidden layer neurons in ANN_i, n is the number of populations, and ANN_i is constructed from P_i, and genotype $g \in \{g_0, \ldots, g_m\}$, then execute the following.

For j = 1,.., u_i

For i = 1,.., n

Select next g from P_{ij}

For $k = 1, .., u_i$, where $k \neq j$

Select a random g_k from P_{ik}

End For

Construct ANN_i using g and all g_k genotypes as the hidden layer.

End For

For l = 1,.., n, where $l \neq i$

For o = 1,.., u_l

Select a random g_o from P_{lo}

End For

Construct ANN_l using all selected genotypes g_o as the hidden layer.

End For

Test the n controllers together in task simulation for a lifetime of q epochs. An epoch is a task trial that is executed for w simulation iterations. Each epoch tests different task and simulation environment dependent conditions. An average fitness (type A) value is assigned to the controller group at the end of the task simulation. This average fitness is calculated as the average of all fitness values attained for all epochs of a task simulation.

End For

The same fitness (type A) is assigned to a group of n controllers, each controller in the group, and the current genotype g_i (selected from P_i) being evaluated. This fitness assignment (f) is presented in equation 3.7.

$$f(nANNs) = f(ANN) = f(g_i) \tag{3.7}$$

3.3.2 Evaluate Elite Controllers

Given that all genotypes in all populations have been assigned a fitness (type A) value, then a fittest set of controllers can be derived according to their *utility*. The utility of ANN_i is calculated as presented in equation 3.8, given that ANN_i is constructed from P_i, u_i is the number of hidden layer neurons in ANN_i, $k \in \{u_0, \ldots, u_i\}$, and $f(g_k)$ is the fitness (type A) of neuron k.

$$utility(ANN_i) = \sum f(g_k) \tag{3.8}$$

Given this, an *elite portion* of the fittest controllers is created as follows.

For i = 1,..,n, where, n is the number of populations.

For l = 1,..,L, where, L is the number of genotypes in the elite portion of any given sub-population.

Construct ANN_i *from* P_i , where each of the u_i genotypes used to construct ANN_i is selected from the elite portion of each of the u_i sub-populations of P_i. Genotypes with an equal fitness (type A) rank (l) are selected from each of the sub-populations.

Construct the other n-1 controllers , where the each of the u_i genotypes used to construct ANN_j ($j \neq i$) is randomly selected from the elite portion of each of the u_j sub-populations of P_j.

Test the n controllers together in task simulation for a lifetime of q epochs. The task simulation will give an average fitness (type C) value which overwrites the fitness (type A) value of each genotype in the elite portion of each population. The average fitness (type C) is calculated as the average of all fitness values attained for all epochs of a task simulation.

End For

End For

The result of these task simulations is that a fitness (type C) value is assigned to each of the genotypes in the elite portion of each population. This fitness value is then used for parent selection and application of variation operators.

3.4 Selection and Variation

This section describes the methods used to select parent genotypes from the elite portion of each population, and then the variation operators that are used to recombine and mutate selected parents.

3.4.1 Elite Selection

The following process is known as *elite selection*, given that selection operates within the elite portion of each population. Given two controllers ANN_i and ANN_j, constructed from the elite portions of population i (P_i) and population j (P_j), respectively, a degree of specialization (S) measured for controllers ANN_i and ANN_j is denoted as $S(ANN_i)$ and $S(ANN_j)$, respectively. In the following selection process description, recombination operators are referred to. These recombination operators are described in section 3.4.3.

IF SD(ANN_i, ANN_j) < SST THEN

Recombine (P_i, P_j)

ELSE *Recombine*(P_i) and then *Recombine*(P_j)

3.4.2 Sub-Population Selection

Given that two populations are to be recombined (using the *recombine*(P_i, P_j) operator), the selection of sub-populations in different populations, where genotypes within these sub-populations are to be recombined, occurs as follows.

1. Run task simulations with elite groups n of controllers. An elite controller ANN_i is one where each of the u_i genotypes corresponding to a hidden layer neuron is selected from the elite portions of each of the u_i sub-populations in P_i. These task simulations result in a fitness (type C) being assigned to each of the elite controllers.
2. In the elite portions of all sub-populations of all populations, rank the genotypes by their fitness (type C). These fitness values overwrite previous fitness (type A) values that were assigned to these genotypes.
3. Select the fittest controller from each elite group ANN_1,..,ANN_n, where a fittest controller is constructed via selecting the fittest genotype from each sub-population in each population.

Given an elite group of n controllers, execute the following procedure.

For each pair i, j $\in \{1, \ldots, n\}$, $i \neq j$

Calculate SD(ANN_i, ANN_j)

End For

Rank these SD values in a list of increasing order

For each pair (i, j) on this list, where $i \neq j$.

IF SD(ANN_i, ANN_j) < SST THEN
Recombine (P_i, P_j)
ELSE *Recombine*(P_i) and then *Recombine*(P_j)

End For

3.4.3 Recombination Operators

Two types of recombination operators are described in the following. Crossover refers to a one-point crossover [43] operation. *Elite*(P_i) refers to the elite portion of all genotypes in population i, where the genotypes are ranked (within each of the u_i sub-populations) of population P_i.

Recombine(Population P_i):

For all u_i sub-populations in P_i

Rank elite portions of all u_i sub-populations

Apply recombination within each sub-population via applying crossover to each g_i and another genotype g, where $g, g_i \in elite(\{g_0, \ldots, g_{mu}\})$, g is randomly selected, $g_i \neq g$, and mu is the number of genotypes in each sub-population.

End For

Survivor selection occurs in the form of recombination producing enough child genotypes in order to replace all genotypes in P_{ik}.

Recombine(Population P_i, Population P_j):

For all P_i, P_j, where, $i \in \{1, \ldots, u_i\}$, $j \in \{1, \ldots, u_j\}$

Rank $elite(P_i)$, $elite(P_j)$ by fitness type C

Pair genotypes according to identical fitness ranks

End For

IF GD(g_1, h_1) + .. + GD(g_L, h_L) < GST, where, g_l is selected from P_i, and h_l is selected from P_j, and $l \in \{1, \ldots, L\}$, and L is the size of the elite portions of P_i and P_j, THEN

For all g_1, h_1 ,.., g_L, h_L

crossover (g_l, h_l)

End For

Survivor selection occurs in the form of recombination producing enough child genotypes in order to replace all genotypes in P_i and P_j.

ELSE $recombine(P_i)$ and then $recombine(P_j)$

3.4.4 Mutation Operator

After recombination, mutation is applied to each gene of each genotype with a predefined degree of probability. *Burst* mutation with a *Cauchy* distribution [57] is used in order to ensure that most weight changes are small whilst allowing for larger changes to some weights.

3.5 Adaptation of Algorithmic Parameters

Recombination between populations is autonomously regulated as a function of the behavioral specialization exhibited by controllers, and recombinations. In both cases, recombination is regulated with respect to fitness progress in each population, where such regulated recombination always occurs between sub-populations situated within different populations. The mechanisms that regulate genotype recombination between n populations are intended to evolve n specialized genotype niches that derive n behaviorally specialized controllers, where these controllers behaviorally complement each other and interact in order to produce a collective behavior. The SST and GST values are adapted by 1.0%, calculated by exploratory experiments and found to be effective for experiments detailed in chapters 4, 5, and 6. This did not change the GD and SD values by too much per regulation, thus missing useful values, and was not too small so as to slow or inhibit the discovery of effective values.

- The SST value is adapted as a function of behavioral relatedness between the fittest n controllers and controller fitness progress (section 3.5.1).
- The GST value is adapted as a function of genotype relatedness between sub-populations situated within n populations and the fitness progress of each of these sub-populations (section 3.5.2).

3.5.1 Specialization for Regulating Recombination

The following heuristics are applied for regulating the SST value 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 (ranked by type C fitness) has increased over the last V generations, and average fitness (for the fittest n controllers) stagnates or is decreasing over this same period, then SST is decremented. 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 measured for at least one of the fittest n controllers has decreased over V generations, and average fitness stagnates or is decreasing (for the n controllers) over this same period, then increment SST. 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.

3.5.2 Recombinations for Regulating Recombination

The following heuristics are applied for regulating the GST value as a function of recombinations and fitness progress.

1. If the number of recombinations between populations has increased over the previous $V + W$ generations, and fitness has stagnated or decreased over this same period, then decrement the GST value.

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

3.6 Adapting Controller Size

As in ESP [57], CONE adapts the number of hidden layer neurons of controllers as a function of fitness progress in a collective behavior task. If the fitness of at least one of the n fittest controllers has not progressed in $V + W$ generations, then adapt the controller size of stagnating controllers (that is, the number of sub-populations in the population from which the stagnating controller is derived). This differs from related work which applies NE to the adaptation of n controllers with fixed sensory input and motor output topologies in collective behavior tasks [197], [21], [140]. In such related work there was no dynamic adaptation of controller size in collective behavior tasks.

Adaptation of hidden layer size in ANN cooperative co-evolution allows for the derivation of controller sizes that effectively complement each other in collective behavior task accomplishment [160]. That is, the number of hidden layer neurons required by a controller to effectively solve a given task greatly depends upon the complexity and nature of the task [158]. 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 in order to solve collective behavior tasks.

A lesion mechanism [102] is applied in order to regulate the adaptation of the number of sub-populations in genotype population i (P_i). This is the population from which a stagnating controller (ANN_i) is derived. The lesion mechanism evaluates the contribution of individual hidden layer neurons to overall controller behavior, and accordingly increases or decreases the number of sub-populations. That is, the current fittest controller derived from P_i is tested in u_i collective behavior task trials (one task trial for each hidden layer neuron in ANN_i). In each task trial ANN_i is tested without one of its u_i hidden layer neurons. In such task trials, ANN_i is evaluated together with the other (currently fittest) n-1 controllers. A minimum fitness is defined as a threshold which ANN_i (minus a specific hidden layer neuron) must surpass in a task trial. If the re-evaluated fitness of the lesioned ANN_i is not below this fitness threshold, then the missing neuron is deemed not to be a worthy contribution to ANN_i, and the neuron is removed. Accordingly, the sub-population from which the unworthy hidden layer neuron is decoded is also removed. Otherwise, if all neurons are found to be worthy contributors to ANN_i, then the number of hidden layer neurons of all controllers derived from population i is increased by one. That is, one more sub-population is created within P_i. Within a new sub-population, genotypes are created by randomly initializing each gene to a value within a given range.

3.7 CONE Process

An overview of the CONE evolutionary process is described in the following.

1. *Initialization.* n populations are initialized, where population P_i ($i \in \{1, \ldots, n\}$ contains u_i sub-populations. Each sub-population contains m genotypes. Sub-population P_{ij} contains genotypes corresponding to neurons assigned to position j in the hidden layer of ANN_i, where, ANN_i is derived from P_i, and $j \leq u_i$. Refer to section 3.1.1 for details on the population structure.

2. *Evaluate all Genotypes.* Systematically select each genotype g in each sub-population in each population, and evaluate g in the context of a complete controller. This controller (containing g) is evaluated together with n-1 other controllers, where these controllers do not contain g. The evaluation results in a fitness (type A) being assigned to g. Refer to section 3.3.1 for a complete description of how all genotypes are evaluated.

3. *Evaluate Elite Controllers.* For each population, systematically construct a fittest controller from the fittest genotypes (elite portion) in each sub-population of the population. The fitness of a controller is determined by its *utility* (also known as fitness type B). A controller's utility is determined by a simple sum of the fitness values of the genotypes corresponding to each hidden layer neuron. 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 (type C) value. This fitness (type C) value overwrites the previously calculated fitness (type A) for each of these genotypes. Refer to section 3.3.2 for a complete description of how elite controllers are evaluated.

4. *Parent Selection.* Given that the two fittest controllers ANN_i and ANN_j constructed from the elite portions of P_i and P_j are calculated as having a sufficiently *similar behavioral specializations* then these populations become candidates for recombination. Between P_i and P_j each pair of sub-populations is tested for *genetic similar*. Genetically similar sub-populations are recombined, and mutation applied. Recombination occurs *within sub-populations* that are measured as *not* being genetically similar to other sub-populations. Similarly, recombination occurs *within* populations that are measured as *not* being *behaviorally similar*. Section 3.4 presents a complete description of the parent selection process.

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 (type C) and genotypes with the same fitness rank are recombined. For recombination that occurs within a sub-population, each genotype in the elite portion of the sub-population is systematically selected and recombined with another genotype randomly selected from the elite

portion. One-point crossover [43] is used for recombination. Refer to section 3.4 for a complete description of the recombination operators.

6. *Mutation.* After recombination, *burst mutation* with a *Cauchy* distribution [57] is applied to each gene of each genotype with a predefined degree of probability. Section 3.4 describes the mutation operator.

7. *Adaptation of Algorithmic Parameters.* If the fitness of at least one of the fittest n controllers has not progressed in V (or $V + W$) generations, then adapt the the *Specialization Similarity Threshold* (SST) value or the *Genetic Similarity Threshold* (GST) value, respectively. Section 3.5 presents a complete description of the method used to adapt the GST and SST values. If fitness of at least one of the n controllers has not progressed in $V + W + Y$ generations, then the number of sub-populations (controller size) for each of the controllers with stagnating fitness, is adapted. Section 3.6 describes the method used to adapt controller size. V, W and Y are constant values which were determined using exploratory experiments for each of the collective behavior case study.

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

Parameters used by CONE are described in appendix E.

3.8 CONE and Related Methods

In the case studies detailed in chapters 4, 5, and 6, CONE is comparatively evaluated with related controller design methods. The CONE architecture stipulates that a hidden layer neuron is the *basic unit* (genotype) upon which artificial evolution operates. Each neuron is directly encoded as a vector of floating point values. Other NE methods that use direct encodings (for example, NEAT: *Neuro-Evolution of Augmenting Topologies* [158]) and indirect encodings (for example, HyperNEAT: *Hypercube-based Neuro-Evolution of Augmenting Topologies* [33], and Cellular Encoding [65]) in order to evolve ANN controller topology are not included in the methods comparison for the following reasons.

1. CONE uses direct encoding in order to evolve connection weights, and adapt the number of hidden layer neurons in each controller. However, the number of input and output neurons connected to the hidden layer, remains static for all controllers. This architecture was sufficiently complex for accomplishing the collective behavior tasks described in chapters 4, 5, and 6. Whilst, indirect encoding methods that adapt controller topology as a function of task requirements are conceivably appropriate for more complex tasks, the application of such methods would add unnecessary complexity to the methods comparison and potentially hinder elucidation of this dissertations research goals.

2. Methods such as NEAT and HyperNEAT use a competitive co-evolution process as a means of deriving species that represent a set of fittest ANN controllers. One focus of this research is the use of cooperative co-evolution as a process that derives controllers to be specialized to a set of complementary behaviors that interact in a beneficial manner.

3. With the exception of HyperNEAT (an extension of NEAT) the NEAT method has not been applied in order to solve collective behavior tasks.

D'Ambrosio and Stanley [33] apply HyperNEAT to a pursuit-evasion task as a means reducing search space dimensionality, deriving heterogeneity in an evolved team, and producing a shared behavior set that could be used by all predator agents. HyperNEAT evolves prey-capture behaviors that are encoded by a single genotype. This genotype represents a team of heterogeneous behavioral roles, where a highly fit genotype corresponds to a set of predator behaviors that cooperate and contribute to a high performance collective behavior. The implication of this is that beneficial predator behaviors do not need to be rediscovered by separate predators. D'Ambrosio and Stanley [33] state that HyperNEAT addresses a problem that is typically encountered by cooperative co-evolution methods that are applied to evolve collective behavior solutions. That is, agents may either readily specialize yet not share behaviors, or may share behaviors yet specialize poorly [187].

CONE also addresses the problem of sharing behaviors between controllers without the need for different controllers to independently derive the same behavior. The approach taken by CONE is to share beneficial genotypes between populations. Recombination of genotypes endemic to different populations is regulated by behavioral specialization and genotype difference metrics. For example, consider that a set of the fittest genotypes (spread across u subpopulations) in a given population derives a beneficial controller behavior. If this behavior is found to be sufficiently similar (defined by the specialization difference metric) to another controller's behavior, and the genotypes that derived this other controller's behavior are sufficiently similar (defined by the genotype difference metric), then recombination occurs between the two populations. Such an approach facilitates the propagation of beneficial behaviors that are shared between multiple controllers.

The main reason that HyperNEAT was not included (besides being a controller design method that uses indirect encoding) as part of the related methods comparison for each of the collective behavior case studies, is that HyperNEAT was devised by its creators after experiments for each of the collective behavior case studies had been completed. Given this, further study is required in order to properly analyze and ascertain the types of collective behavior tasks that are appropriately solved by CONE versus those that more appropriately solved by methods such as HyperNEAT.

3.9 Conclusions

Collective Neuro-Evolution (CONE) is a cooperative co-evolution controller design method that uses NE to evolve solutions to collective behavior tasks that benefit from, or require, behavioral specialization. CONE creates n genotype populations from which n ANN controllers are evolved. These controllers must cooperate and behaviorally specialize in order to accomplish a given collective behavior task. CONE facilitates and uses emergent behavioral specialization as a problem solver, for the purpose of increasing collective behavior task performance. In order to effectuate specialization CONE utilizes *genotype* and *behavioral specialization* difference metrics for regulating recombination between genotype populations. The genotype difference metric recombines and propagates genotypes specialized to similar functionalities. The specialization difference metric recombines and propagates controllers with beneficial behavioral specializations. Behavioral specialization is defined and measured according to a specialization metric, which forms an integral part of CONE.

4. COLLECTIVE BEHAVIOR CASE STUDY: PURSUIT-EVASION TASK

This chapter investigates the application of the *Homogenous Conventional Neuro-Evolution* (HomCNE), *Heterogenous Conventional Neuro-Evolution* (HetCNE), *Cooperative Co-evolutionary Genetic Algorithm* (CCGA), *Multi-Agent Enforced Sub-Populations* (Multi-Agent ESP[1]), and *Collective Neuro-Evolution* (CONE) methods to the evolution of collective behaviors in a multi-robot pursuit-evasion task. The efficacy of each of these methods for evolving a prey-capture behavior is investigated. A prey-capture behavior is a collective behavior that requires the cooperation of multiple predators (pursuers) that attempt to immobilize one or more prey (evaders). This chapter also investigates the capability of HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE for evolving behavioral specialization that results in high performance prey-capture behaviors. The investigation of emergent specialization remains a relatively unexplored area of research in the pursuit-evasion task [110].

4.1 Pursuit-Evasion Task

In this pursuit-evasion task a team of pursuer robots (*predators*) are required to collectively immobilize (capture) at least one evader robot (*prey*). Prey do not move deterministically, so it is impossible for predators to consistently predict a prey's movement. The prey capture task is made more complex via giving the prey an advantage of greater speed. Previous research [110] elucidates that at least two predators are required to accomplish this task, and that predators with complementary behavioral specializations often increase prey capture time. The degree of behavioral specialization exhibited by a given predator controller is defined by the frequency with which the predator switches between v distinct behaviors over the course of its lifetime (section 3.2.1). Predator controllers do not produce a set of motor outputs that directly correspond to a set of distinct behaviors. Rather, a set of varying wheel speeds and orientations produce behaviors that are not readily distinguishable. Methods from related work [118], [119], [125] are used in order to correlate sensory activation and motor output value ranges with behaviors observed in pursuit-evasion experiments. These methods correlate specific sensory input values to particular motor output values, where such motor outputs correspond to a specific behavior. Section 4.6 presents the methods that identify specialized and collective behaviors.

[1] Multi-Agent ESP and MESP are used interchangeably.

A continuous environment corresponding to a 1000cm x 1000cm area is simulated using an extended version of the EvoRobot Khepera simulator [120]. Details of sensor and actuator properties are presented in table 4.1, and details regarding how Khepera sensors and actuators are simulated, as well as assumptions made by the continuous simulation model are described in related work [110]. The sensor and actuator configuration of each prey is that of a Khepera mobile robot [101]. As presented in figure 4.1(A), a prey is equipped with eight infrared proximity sensors ([SI-0, SI-7] in figure 4.1(A)) positioned on its periphery, as well as having a light on its top (L-0 in figure 4.1(A)). This light can be detected by predator light sensors, and is used so each predator can distinguish fellow predators from a prey. When an obstacle comes within range (table 4.1) of a prey's proximity sensor, that sensor is activated with a value inversely proportional to the distance to the obstacle [120]. Sensor values are normalized within the range [0.0, 1.0]. Each prey is also equipped with two wheel motors ([MO-0, MO-1] in figure 4.1) that control its speed and orientation.

As illustrated in figure 4.2, each prey uses a feed-forward ANN controller, consisting of eight sensory input neurons and two motor output neurons, fully connected to five hidden layer neurons. Sensory inputs encode the state of the eight infrared proximity sensors, and motor outputs encode the speed of a prey's wheels. Motor neuron output values are computed via applying the sigmoid function [70], and then multiplying the output value by 1.2. This sets prey speed to be 20% faster than predator speed.

Predator sensor and actuator configuration is that of a Khepera mobile robot [101]. As presented in figure 4.1(B), each predator is equipped with eight infrared proximity sensors ([SI-0, SI-7] in figure 4.1(B)), as well as eight light ([SI-8, SI-15] in figure 4.1(B)) sensors, positioned on its periphery. When an obstacle (another predator or wall) comes within range (table 4.1) of a given proximity sensor, that sensor is activated with a value inversely proportional to the distance to the obstacle [120]. Likewise, when a prey comes within range (table 4.1) of a given light sensor, that sensor is activated with a value inversely proportional to the distance to the prey [120]. Sensor values are normalized within the range [0.0, 1.0]. Also, each predator is equipped with two wheel motors ([MO-0, MO-1] in figure 4.1) that control its speed and orientation.

A predator controller (figure 4.3) is a simple recurrent ANN [44], used in order to emulate short term memory, where memory is a prerequisite for a predator team to collectively capture a prey. A hidden layer of six sigmoidal units fully connects 22 sensory input neurons to two motor output neurons. Sensory input neurons encode the state of eight infrared proximity sensors and eight light sensors ([SI-0, SI-15] in figure 4.3), as well as the hidden layer activation values from the previous simulation iteration ([SI-16, SI-21] in figure 4.3). Motor outputs ([MO-0, MO-1] in figure 4.3) encode the speed of the two wheels. The output value of each motor neuron updates the speed of the corresponding wheel at each simulation iteration. The initial number of hidden layer neurons in a predator controller was selected from exploratory experiments, which indicated six hidden layer neurons to be appropriate for collective prey-capture.

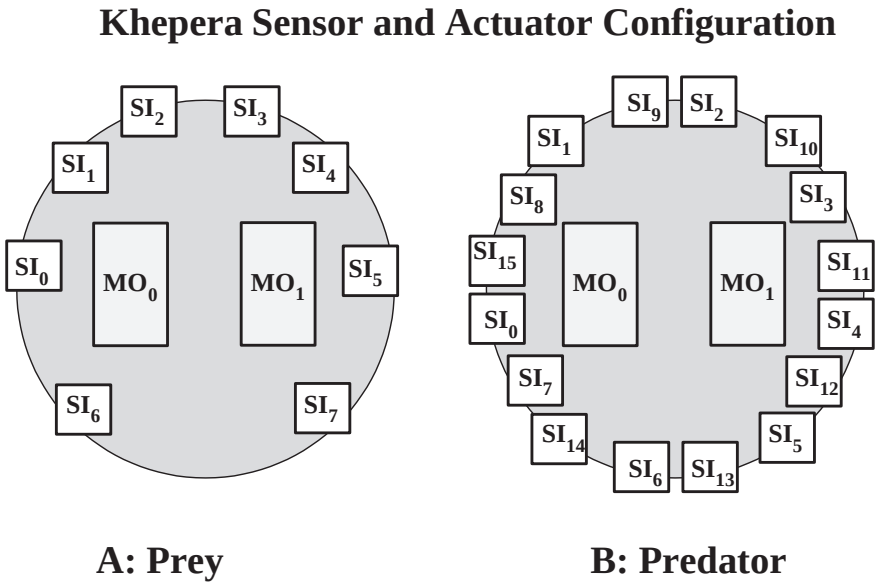

Fig. 4.1: Sensor and Actuator Configuration. Prey (A) have only proximity sensors as well as a light. Predators (B) have light and proximity sensors.

Tab. 4.1: Simulation Parameters. Pursuit-evasion experiment settings.

Pursuit-Evasion Simulation Parameter Settings	
Pursuit-evasion experiment	20 runs
Prey shaping experiment	20 runs
Team types per experiment	[1, 10]
Environment size	1000cm x 1000cm
Predators / Prey	Khepera
Predator / Prey proximity, light sensor range	22cm
Predator team size	[2, 6]
Prey per pursuit-evasion experiment	[1, 2]
Obstacles per prey training experiment	[2, 6]
Obstacles shape	Cylinder
Obstacles diameter	5.5cm

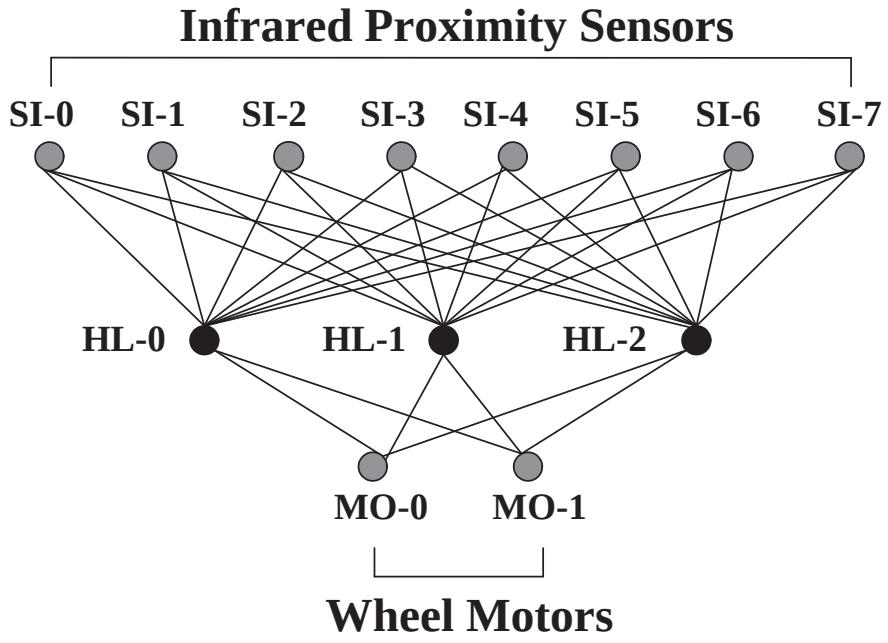

Fig. 4.2: Prey Feed-Forward ANN Controller. The controller is evolved by the CONE method prior to being placed in the pursuit-evasion experiments.

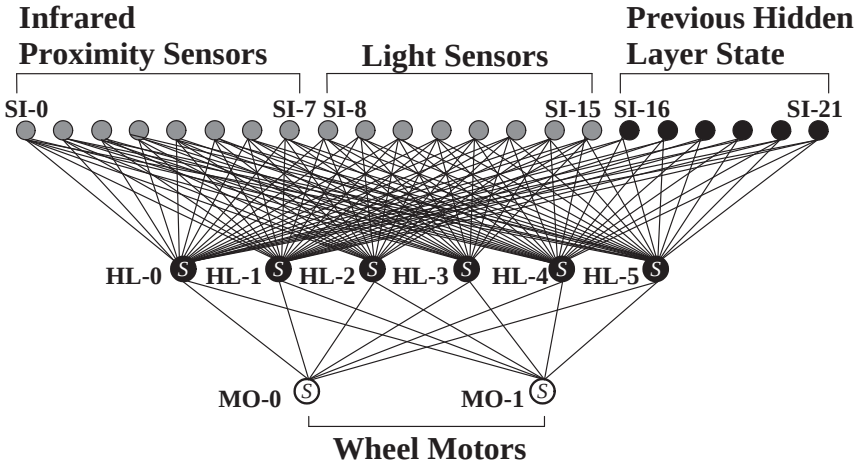

Fig. 4.3: Predator Recurrent ANN Controller. The controller is evolved by either the HomCNE, HetCNE, CCGA, Multi-Agent ESP or CONE method.

Tab. 4.2: Team Types. Ten team types are tested in the pursuit-evasion experiments.

Team Type	**Team Composition**
1	Two predators and one prey robots
2	Three predators and one prey robots
3	Four predators and one prey robots
4	Five predators and one prey robots
5	Six predators and one prey robots
6	Two predators and two prey robots
7	Three predators and two prey robots
8	Four predators and two prey robots
9	Five predators and two prey robots
10	Six predators and two prey robots

4.2 Parameters and Experimental Setup

This section presents the Neuro-Evolution (NE) and simulation parameters, and setup used for the pursuit-evasion experiments. Table 4.1 and table 4.3 present the pursuit-evasion and NE parameter settings, respectively. NE parameter settings are those used by the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE methods. Simulation parameter settings are those used by the simulator, such as the number of prey and predators, the environment size, and the number of experimental runs. These parameter settings were selected given the success of similar settings in related evolutionary robotics experiments [100].

Test experiments conducted for a parameter sensitivity analysis, indicated that minor changes to parameter values produced similar results. Changing parameters values to within 20% of the values given in tables 4.1 and 4.3 resulted in the evolutions of predator teams that yielded prey capture times within approximately 15% of the results presented in section 4.4.

One experiment consisted of placing a predator team in a given environment and applying a given NE method to evolve controllers. Each experiment consisted of two distinct phases: an *evolution phase*, and a *testing phase*. The term task performance refers to prey capture time.

- *Evolution phase:* The controllers of a predator team are evolved for 500 generations (table 4.3) using a given NE method and simulation environment. Each generation corresponds to one predator *lifetime*. Each predator lifetime lasts for 50 epochs, where each epoch consists of 1000 simulation iterations. Each epoch represents a scenario that tests different predator and prey orientations and starting positions in the environment. A team's prey capture time is calculated as an average taken over all epochs of a predator's lifetime. Evolved collective prey-capture behaviors are presented in section 4.5.

- *Testing phase:* The fittest n controllers (the fittest team) are selected and set to execute in the same environment for one predator lifetime. The testing phase does not evolve controllers, so the evolved connection weights of each predator's controller remains static. Task performance results presented in this chapter are averages calculated over 20 runs of the fittest controllers in a test environment.

4.3 Shaping Prey Behavior: Controller Evolution

Prior to being placed in pursuit-evasion experiments with a predator team, a set of shaping experiments [132] using CONE were implemented for evolving the connection weights of the prey controller. Shaping experiments used between two and six predator robots together with one or two prey robots in test environments. Shaping experiments evolved static and dynamic obstacle avoidance behaviors. Static obstacles were walls and other obstacles that did not move over the course of a simulation. Dynamic obstacles were predator robots. The objective of the shaping experiments was to incrementally evolve prey behaviors in a set of incrementally complex environments. The result was a static and dynamic obstacle avoidance behavior in prey robots.

Shaping experiments for incrementally evolving a prey's evasion behavior were found to be necessary. Direct artificial evolution produced a trivial obstacle avoidance behavior that did not necessitate the evolution of complementary specialized behaviors in predator teams. Shaping experiments indicated that that the feed forward controller architecture presented in figure 4.2 was sufficient for the purposes of evolving an effective evasion behavior. Furthermore, the fittest controller derived uses three hidden layer neurons.

In each shaping experiment the prey performed static or dynamic obstacle avoidance, where the number of static obstacles and predators situated in the environment was incrementally increased. For each shaping experiment, the fitness function was how far and fast the prey could move in a straight line without colliding with an obstacle [124]. A prey controller *lived* for 50 epochs (one generation), where each epoch consisted of 1000 simulation iterations, and each shaping experiment consisted of 500 generations. Each epoch constituted a test scenario testing randomly generated obstacle positions and different prey orientations and starting positions. Fitness assigned was the average of all epochs of a prey's lifetime. If a prey collided with an obstacle, a zero fitness was awarded and a new epoch started. For each shaping experiment, CONE was selected for controller evolution given that it was found to produce fitter controllers comparative to homCNE, hetCNE, CCGA, and Multi-Agent ESP.

Shaping Experiment 1: Static Obstacle Avoidance

In shaping experiment 1, CONE evolved a prey controller for static obstacle avoidance. This task was selected so a prey obstacle avoidance could be evolved from a random genotype population. In the environment, between two and six

Tab. 4.3: NE Parameters. Parameter settings for HomCNE, HetCNE, Multi-Agent ESP, CCGA and CONE in the pursuit-evasion experiments.

Pursuit-Evasion Neuro-Evolution (NE) Parameter Settings	
Generations	500
Epochs	50
Iterations per epoch	1000
Mutation probability	0.05
Mutation type	Burst (Cauchy distribution)
Mutation range	[-1.0, +1.0]
Fitness stagnation Y	20 Generations (CONE/Multi-Agent ESP)
Fitness stagnation V	10 Generations (CONE)
Fitness stagnation W	10 Generations (CONE)
Genotype Distance (GD)	[0.0, 1.0] (CONE)
Specialization Distance (SD)	[0.0, 1.0] (CONE)
Genotype population elite portion	25%
Weight (gene) range	[-10.0, +10.0]
Crossover	Single point
Predator sensory input neurons	22
Predator hidden layer neurons (Initial number)	6
Predator motor output neurons	2
Total genotypes	600
Genotype	Input-output weights: 1 Neuron (Multi-Agent ESP/CONE), All weights: 1 ANN (HomCNE, HetCNE, CCGA)
Genotype representation	Floating point value vector
Genotype populations	[2, 6] (HomCNE, HetCNE, CCGA, CONE), 6 (Multi-Agent ESP)
Genotype length	24 (CONE, Multi-Agent ESP), 24 x 6 (HomCNE, HetCNE, CCGA)
Genotypes per population	600 (HomCNE, HetCNE), [100, 300] (CCGA, CONE), 100 (Multi-Agent ESP)

(determined randomly) uniformly round objects, each 5.5cm in diameter, are placed in random positions in the environment. This number of obstacles was selected since this is also the minimum and maximum predator team size in the pursuit-evasion experiments. Uniformly round obstacles were used given that they are perceived in the same way independent from a sensors point of view [100]. The fittest evolved controller was placed in shaping experiment 2.

Shaping Experiment 2: Dynamic Obstacle Avoidance

In shaping experiment 2, CONE evolved a prey controller for dynamic obstacle avoidance. The prey was placed in an environment containing between two and six predators. In this case, predators utilized a heuristic controller (appendix F), which dictates that a predator moves in a straight line, at maximum speed, towards the prey robot, when the prey is within light sensor range. Otherwise a predator moves stochastically. The fittest dynamic obstacle avoidance controller was placed in shaping experiment 3.

Shaping Experiment 3: Dynamic and Static Obstacle Avoidance

In shaping experiment 3, CONE evolved one prey controller in an environment containing between two and six obstacles and between two and six predators. The predators in this task used heuristic controllers. The fittest dynamic and static obstacle avoidance controllers were placed in shaping experiment 4. The fittest evolved prey controllers were also placed in pursuit-evasion experiments using one prey robot (section 4.4).

Shaping Experiment 4: Static Obstacle Avoidance with Two Prey Robots

In shaping experiment 4, CONE evolved two prey controllers in an environment containing two prey robots and between two and six static obstacles. The fittest static obstacle avoidance controllers evolved for each prey robot were placed in shaping experiment 5.

Shaping Experiment 5: Dynamic Obstacle Avoidance with Two Prey Robots

In shaping experiment 5, CONE evolved two prey controllers in an environment containing two prey robots and between two and six predators using static obstacles. The fittest dynamic obstacle avoidance controllers evolved for each prey robot were placed in shaping experiment 6.

Shaping Experiment 6: Two Prey Robots

In shaping experiment 6, CONE evolved two prey controllers in an environment containing two prey robots, between two and six static obstacles, and between two and six predators using heuristic controllers. The fittest dynamic and static obstacle avoidance controllers evolved for each prey robot were placed in pursuit-evasion experiments using two prey robots (section 4.4).

4.4 Pursuit-Evasion Experiments

Experiments use between two and six predators together with one or two prey in the pursuit-evasion task. Experiments measure the impact of a *collective behavior design method* and *team type* upon *prey capture time*. The experimental objective is to ascertain which collective behavior design method maximizes prey capture time, and to relate the contribution of emergent behavioral specialization to prey capture time.

- *Collective Behavior Design Method*: Each predator is equipped with a recurrent ANN which is adapted with one of the following NE methods.
  1. HomCNE
  2. HetCNE
  3. CCGA
  4. Multi-Agent ESP
  5. CONE
- *Team Type*: Given a collective behavior design method, ten different team configurations of predators and prey are tested (defined in table 4.2).

Fitness Function: Collective prey capture behaviors and the resulting fitness (prey capture time) yielded from the application of HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE are presented in section 4.5. For any given predator team, a fitness function calculates the average time for which a prey is immobilized (captured). This average is calculated over all epochs of all predator lifetimes. The fittest controller is then selected from all lifetimes (generations) in an experiment. A fitness estimation method is used, which assumes that each predator in a team contributes equally to the capture of a prey. Thus each predator receives an equal fitness reward when a prey is captured.

4.4.1 Evolving Prey-Capture Behavior

The following sections describe the application of HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE to the pursuit-evasion task.

Evolving Prey-Capture Behavior with Homogenous CNE

The HomCNE process is illustrated in figure 2.3 and described in section 2.3.1. For a team of n predators (where, $n \in [2, 6]$), one genotype population containing 600 genotypes is initialized. Each genotype represents the hidden layer connection weights of one predator ANN controller. Each genotype is encoded as vector of 156 floating point values. That is, 24 sensory inputs plus two motor outputs, each multiplied by six hidden layer neurons (table 4.3). A predator controller is derived via randomly selecting one genotype from the elite portion (table 4.3) of the population. This selected genotype is then replicated n times in order to create a team of predator clones.

Evolving Prey-Capture Behavior with Heterogenous CNE

The HetCNE process is illustrated in figure 2.3 and described in section 2.3.1. For a team of n predators (where, $n \in [2, 6]$), one genotype population containing 600 genotypes is initialized. Each genotype represents the hidden layer connection weights of one predator controller. Each genotype is encoded as vector of 156 floating point values. That is, 24 sensory inputs plus two motor outputs, each multiplied by six hidden layer neurons (table 4.3). A predator controller is derived via randomly selecting n genotypes from the elite portion of the population, such that no genotype is selected more than once. The selected genotypes are then decoded into controllers for a team of n predators.

Evolving Prey-Capture Behavior with CCGA

For a team of n predators (where, $n \in [2, 6]$), n genotype populations are initialized. Each population is initialized with [100, 300] genotypes. Each genotype is encoded as vector of 156 floating point values. That is, 24 sensory inputs plus two motor outputs, multiplied by six hidden layer neurons (table 4.3). A predator controller is derived via randomly selecting one genotype from the elite portion of a given population. This process is then repeated n times, in order to derive n different controllers for a team of n predators.

Evolving Prey-Capture Behavior with Multi-Agent ESP / CONE

Multi-Agent ESP and CONE create n (where, $n \in [2, 6]$) genotype populations for deriving n controllers. Population i consists of u sub-populations, where u corresponds to the number of hidden layer neurons used by a controller derived from population i. Each population is initialized with [100, 300] genotypes. Each genotype is encoded as vector of 26 floating point values. That is, 24 sensory inputs plus two motor outputs (table 4.3). A predator controller is derived via randomly selecting one genotype from the elite portion of each sub-population for a given population. These genotypes then collectively form the hidden layer of one controller. This process is repeated n times in order to derive n different controllers for a team of n predators.

Specific to CONE, the number of generations (V in section 3.7) which fitness progress can stagnate within one or more populations (n controllers) before the GD value is adapted (section 3.1.2) is presented as *fitness stagnation* V in table 4.3. Also, the number of generations (W in section 3.7) which fitness can stagnate in any given population before the number of sub-populations is adapted (section 3.6), is presented as *fitness stagnation* W in table 4.3.

4.5 Evolved Prey-Capture Behaviors and Task Performances

This section describes the collective prey-capture behaviors evolved using HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE. Section 4.6 describes the methods used to measure and identify the emergent collective prey-capture behaviors and constituent individual predator behaviors.

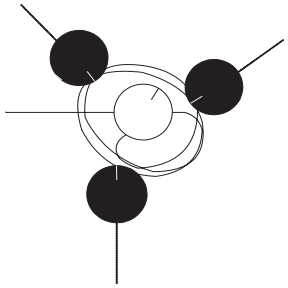

Fig. 4.4: Entrapment Behavior. A tangential bar in a circle indicates the current heading of a predator or prey. Black circle: predators. White circle: prey.

4.5.1 HomCNE / HetCNE Evolved Behavior: Entrapment

Entrapment is a prey-capture behavior that emerges in approximately 80% of experiments applying HomCNE and HetCNE. In the entrapment behavior each predator adopts the same behavioral role of moving in a straight line towards a prey. This behavioral role is termed *pursuer*. Each predator moves towards the prey from a different direction so as to immobilize the prey within a triangular formation. The entrapment behavior is most effective for team types 2 and 3. That is, teams containing one prey and three or four predators. Figure 4.4 illustrates an example of the entrapment behavior using team type 2.

4.5.2 CCGA / Multi-Agent ESP Evolved Behavior: Pursuer-Blocker

Pursuer-blocker is a prey-capture behavior that emerges in approximately 65% of experiments applying CCGA and 70% of experiments applying Multi-Agent ESP. Figure 4.5 illustrates an example of the pursuer-blocker behavior using team type 2. Predators A and B are the *pursuers*, assuming positions behind and to either side of the prey. Predator C assumes the role of a *blocker*. When the prey moves within light sensor range of predator C, this predator moves directly towards the prey. The prey then turns to avoid predator C, however its evasion is halted by pursuing predators. The result is that the prey becomes immobilized between the three predators. The pursuer-blocker behavior is most effective for team types 2 and 3. Team types 4 and 5 yield comparatively poor results due to physical interference that occurs between predators as they collectively approach a prey. Pursuer-blocker fails with team type 1, since two predators are insufficient.

4.5.3 CCGA / Multi-Agent ESP / CONE Evolved Behavior: Spiders-Fly

Spiders-fly is a prey-capture behavior that emerges in approximately 40% of experiments applying CCGA, 35% of experiments applying Multi-Agent ESP, and 50% of experiments applying CONE. Figure 4.6 presents an example of the *spiders-fly* behavior using team type 2. At simulation time t the prey is

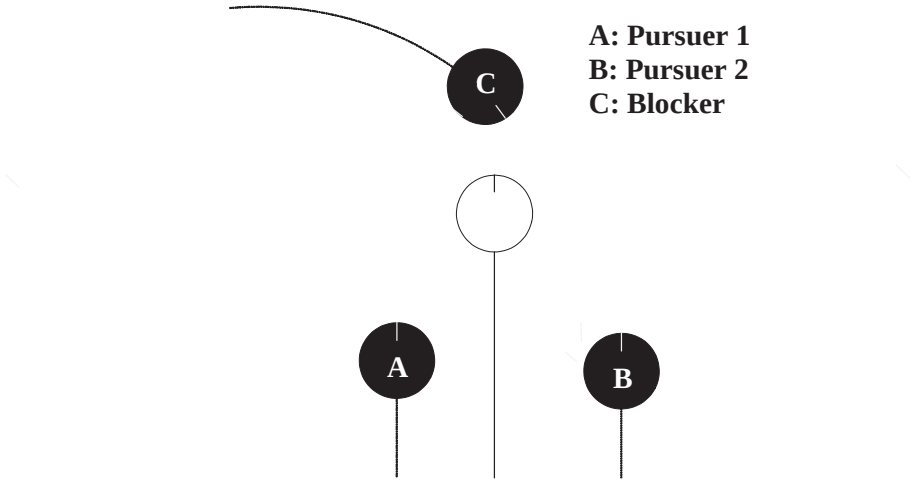

Fig. 4.5: Pursuer-blocker behavior. A tangential bar in a circle indicates the heading of a predator or prey. Black circles: predators. White circle: prey.

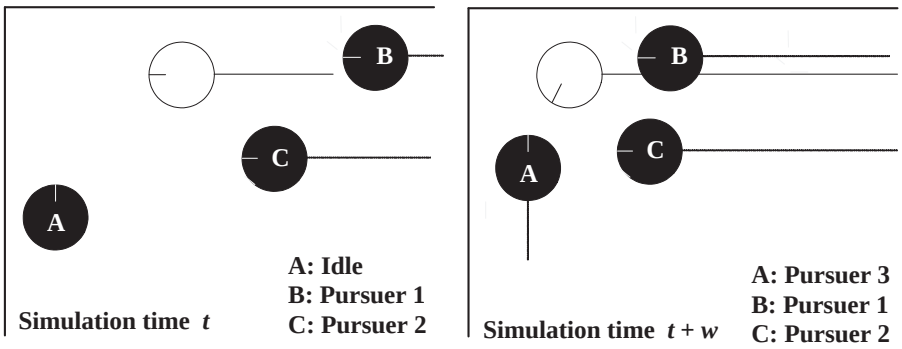

Fig. 4.6: Spiders-Fly Behavior. A tangential bar in a circle indicates the current heading of a predator or prey. Black circles: predators. White circle: prey.

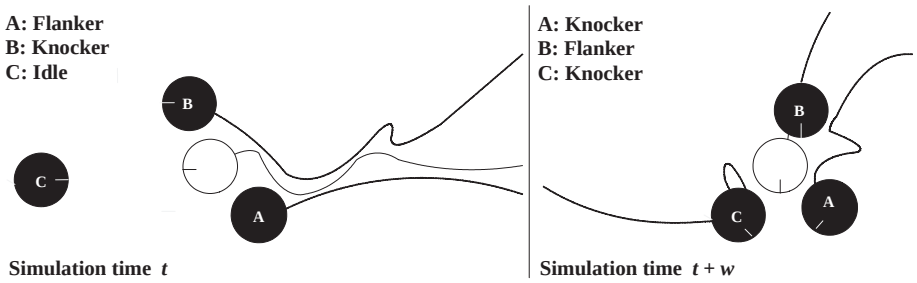

Fig. 4.7: Role switcher behavior 1. A tangential bar in a circle indicates the current heading of a predator or prey. Black circles: predators. White circle: prey.

following a wall, and is being pursued by predators B and C. As the prey reaches the corner and turns about, predator A (previously idle in close proximity to the corner) becomes active. The result is that at simulation time $t + w$ the prey is immobilized between the corner and predators A, B, and C. The *spiders-fly* behavior is most effective when using team types 1 to 3. Team types 4 to 6 fail in the early stages of the CONE evolutionary process ($\leq$ 250 generations) due to physical interference that occurs between predators as they collectively approach a prey. However, team types 4 to 6 often succeed in later stages of the CONE evolutionary process ($>$ 250 generations) given that the fourth, fifth and sixth predators evolve so as to assume *idle* behaviors (section 4.5.6).

4.5.4 CONE Evolved Behavior: Role-Switcher

Role-switcher is a prey-capture behavior that emerges in approximately 80% of experiments applying CONE. Figures 4.7 and 4.8 illustrate two versions of role switcher using team types 2 to 3. Several versions of the role-switcher behavior emerged, however, only two are described and illustrated here due to space constraints. In each version, different predators adopt multiple and complementary behavioral roles, and switch between these roles in order to maintain the effectiveness of the prey-capture behavior. These behavioral roles are named: *flanker*, *knocker* and *idle*. A *flanker* is a predator that remains in close proximity to the left or right hand side of the rear of a prey. A flanker repeatedly collides with a prey so as to force the prey's movement in a particular direction. A *knocker* is a predator that consistently moves in a semi-circular motion so as to repeatedly collide with the prey, and thus slow its movement. An *idle* predator is one that does not move. The role switcher strategy is effective for team types 2 to 5. Given that at least four predators are within sensory range of a prey, the closest three predators participate in the role-switcher behavior, whilst the other predators remain idle. The emergence of these idle predator behaviors is discussed in section 4.5.6. Two predators (team type 1) are insufficient to immobilize a prey in this case.

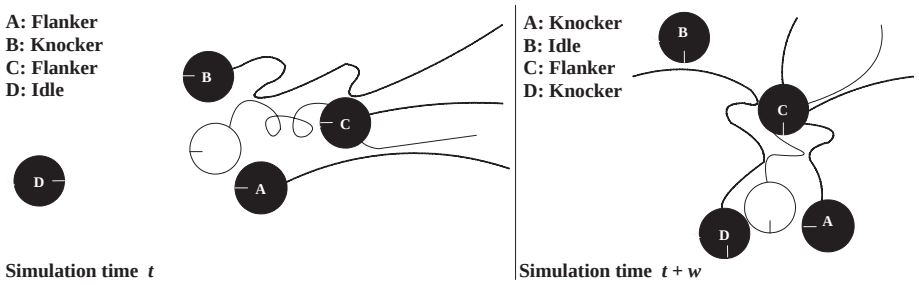

Fig. 4.8: Role switcher behavior 2. A tangential bar in a circle indicates the heading of a predator or prey robot. Black circles: predators. White circle: prey.

Role Switcher Behavior 1

Figure 4.7 illustrates an example of the first version of *role switcher* operating with team type 2. At simulation time t the prey turns left 90 degrees to evade predators A and B. Predator B switches its behavior from a knocker to a flanker role, and predator C switches its behavior from an idle to a knocker role. The result is that the predators stay in close proximity to the prey, and at simulation time $t + w$, capture it within a triangular formation.

Role Switcher Behavior 2

Figure 4.8 illustrates an example of the second version of the *role switcher* behavior, operating with team type 3. At simulation time t the prey turns left 90 degrees to evade predators A and B. Predator B switches its behavior to an idle role, whilst predator A switches to a knocker role. At the same time predator D switches from an idle to a knocker role, whilst predator C maintains its flanker role. The result is that at simulation time $t + w$ predators A, C and D capture the prey within a triangular formation.

Role Switcher Behavior 3

Figure 4.9 illustrates an example of the third version of the *role switcher* behavior, operating with team type 2. At simulation time t the prey turns left 180 degrees to evade predators A and B, both predators A and B switch their behaviors from knocker to flanker roles. At the same time predator C switches its behavior from a flanker to a knocker role. The result is that at time $t + w$ the predators manage to immobilize the prey within a triangular formation.

4.5.5 Pursuit-Evasion Experiments Testing Two Prey

This section presents results of pursuit-evasion experiments that test between two and six predators and two prey. Figure 4.10 presents average prey-capture times, for all methods, as being lower for experiments testing two prey (team

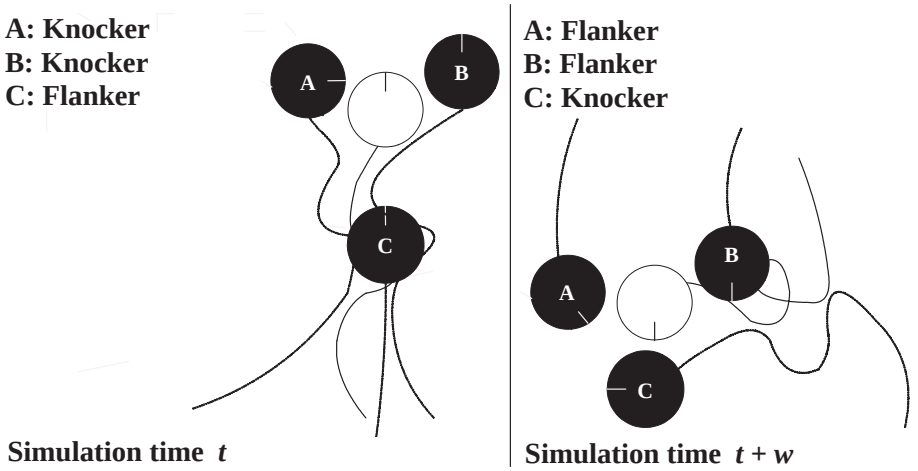

Fig. 4.9: Role Switcher Behavior 3. A tangential bar in a circle indicates predator or prey heading. Black circles: predators. White circle: prey.

types 6 to 10), comparative to experiments testing one prey (team types 1 to 5). This is a result of predators switching between the prey that they pursue. If two prey are within close proximity of each other, then predators often switch between pursuing each of the prey. This inconsistent pursuit behavior, and the need for predators to avoid colliding with each other as they approach a prey, decreases the chance of a predator team forming a prey capture behavior.

4.5.6 Statistical Comparison of Task Performance Results

Figure 4.10 presents average prey capture times yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams for all team types. To draw conclusions, a set of statistical tests are performed in order to gauge task performance differences between respective method results. To compare task performances yielded by two 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 [48] 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 task performance results of any two teams evolved by given methods, an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ.

Appendix A presents the results of this statistical comparison. This comparison supports the hypothesis that CONE evolves teams yielding a higher task

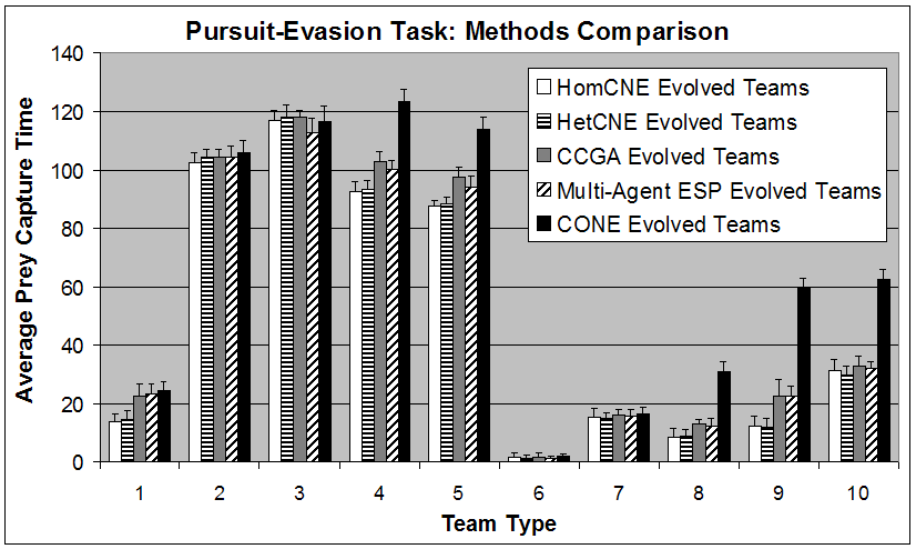

Fig. 4.10: Average Prey-Capture Times (Simulation Iterations). Fitness of HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

performance comparative to those evolved with HomCNE, HetCNE, CCGA and Multi-Agent ESP. That is, CONE evolved teams yield comparatively higher performances for team types 4, 5, 8, 9, and 10, and comparable performances for team types 1, 2, 3, 6, and 7.

The higher performance of CONE evolved teams is supported by the emergence of the *idle behavioral role*. The idle behavior specialization provides a means of reducing physical interference between predators, thereby increasing the effectiveness of prey-capture behaviors in, and thus fitness of, CONE evolved teams. This specialized behavioral role emerges in predator teams evolved with team types 4, 5, 8, 9 and 10. Idle behavioral roles do not emerge in predator teams operating in team types: 1, 2, 3, 6, and 7. It is theorized that the idle predator behavioral role emerges as a means of reducing physical interference between predators. That is, the idle behavior increases the prey capture time of the role switcher, which in turn increases the fitness of CONE evolved teams.

4.5.7 The Role of Difference Metrics in CONE

This analysis supports the efficacy of the *Genotype Difference Metric* (GDM) and the *Specialization Difference Metric* (SDM) for facilitating behavioral specialization, and increasing task performance in CONE evolved teams. As part of this analysis, CONE is re-executed with the following experimental setups.

1. *CONE without GDM (CONE-1):* Predator teams are evolved by CONE without the GDM. The SDM remains active.

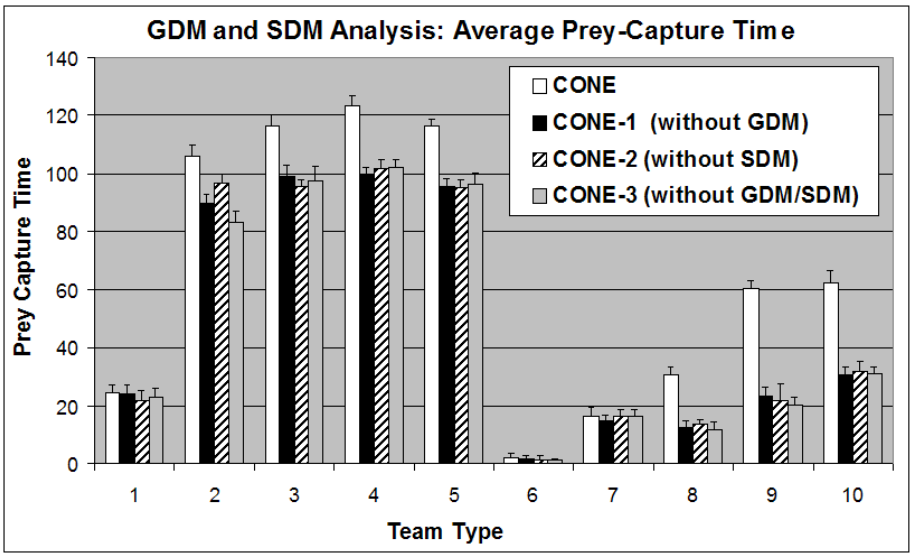

Fig. 4.11: GDM Analysis Results. Prey-capture times yielded by the original CONE setup and each of the CONE variants: CONE-1, CONE-2, and CONE-3.

2. *CONE without SDM (CONE-2):* Teams are evolved by CONE without the SDM. The GDM remains active.

3. *CONE without GDM and SDM (CONE-3):* Teams are evolved by CONE without the GDM and SDM.

Figure 4.11 presents prey-capture times that result from applying each of the variants to the original CONE experimental setup (CONE-1, CONE-2, and CONE-3) for team types 1 to 10. Prey-capture results are averaged over 20 experimental runs. For comparison, results previously attained by CONE evolved teams are also presented in figure 4.11. A statistical comparison of results presented in figure 4.11 indicate that teams evolved by CONE without the GDM (CONE-1), SDM (CONE-2), and both the GDM and SDM (CONE-3), yielded a significantly lower task performance comparable to CONE evolved teams (for all team types except 1, 6 and 7). Appendix A presents t-test values resulting from this statistical comparison. For a majority of the team types tested, both the GDM and SDM are beneficial in terms of increasing the task performance of CONE evolved predator teams.

4.6 Specialized Behaviors Analysis

This section describes an analysis that measures, identifies, and evaluates the contribution of behaviorally specialized predator robots to emergent collective

prey-capture behaviors. A collective prey-capture behavior is defined as a behavior formed by the interaction of multiple individual (potentially specialized) behaviors, where such a collective behavior successfully captures a prey robot.

Section 4.6.1 describes methods for identifying emergent prey-capture behaviors, both specialized and non-specialized. Behavioral identification is accomplished via measuring and correlating sensor and motor actuator activation values for observed prey-capture behaviors.

Individual prey capture behaviors, and the collective prey-capture behaviors that they contribute to, are reproduced via manually setting sensor activation values in test simulation instances. This procedure for reverse engineering observed behaviors verifies that certain motor outputs (behaviors) are yielded by individual predators given a particular set of sensory inputs in a particular simulation instance. Section 4.6.5 evaluates the contribution of individual predators with specialized behaviors to the effectiveness of prey-capture behaviors used by the fittest teams. Section 4.6.6 supports a supposition that multiple predators adopting different and complementary behavioral specializations are required in order to achieve the highest prey-capture times.

4.6.1 Reverse Engineering Observed Predator Behaviors

The identification of individual behaviors, where such behaviors contribute to a collective prey-capture behavior occurs using the following procedure.

1. The experimenter observes the behavior of predators in a team during simulations where the predators collectively capture a prey. The sensory input values for each of the predators during the instances of prey capture are recorded. The experimenter observes 20 simulations (randomly selected from the set of all simulations) where the same collective prey-capture behavior is observed. Sensory inputs values are measured for each predator over the duration of prey capture for each of the 20 simulations.

2. If a predator that operates with the same range of sensory input values for the duration of prey capture is observed in all 20 simulations as participating in the same collective prey-capture behavior, then the sensory input values that correspond to this predator's individual behavior are identified as potentially contributing to an observed prey-capture behavior.

3. In order to validate individual behaviors (identified by measured sensory input values), each individual behavior is reproduced in a test experiment using the measured sensory input values. The reproduction of individual behaviors is described in section 4.6.2.

4. In order to verify that identified individual behaviors contribute to observed collective prey-capture behaviors, the individual behaviors of all predators in a team (that collectively capture a prey) are reproduced and placed together in a test experiment. If the individually reproduced behaviors collectively reproduce an observed prey-capture behavior (in 20

test simulations), then each of the individually identified and reproduced behaviors is considered to have been validated. The reproduction of collective prey-capture behaviors is described in section 4.6.3.

5. The behavioral specialization metric (section 3.2.2) is applied to the individual behavior exhibited by each predator in the fittest team evolved by each method. The specialization metric determines if each predator comprising each of the fittest teams is specialized to one of the identified individual behaviors, or not specialized to any behavior at all. That is, the specialization metric measures the frequency with which a predator switches between a given set of motor outputs (corresponding to an identified individual behavior) and other sets of motor outputs (corresponding to other identified individual behaviors). Predators that are found to be specialized to an individual behavior are assumed to also be specialized to one of the collective prey-capture behaviors. The measuring and validation of behavioral specialization for identified individual behaviors is further described in section 4.6.4 and section 4.6.5.

The light and proximity sensor activation values of each of the 20 sampled instances of individual and collective prey-capture behaviors, for the fittest teams evolved by each method, are presented in appendix B.

4.6.2 Reproducing Individual Predator Behaviors

As presented in section 4.5, several individual behaviors were observed in simulations, and the sensory input values corresponding to these behaviors were recorded. These observed individual behaviors were labeled: *pursuer*, *blocker*, *flanker*, *knocker* and *idle*. In order to reproduce each of these individual behaviors, the inputs of a predator's controller were manually set by the experimenter with the average proximity and light sensor values measured for each sensory input neuron. That is, the average sensor values calculated over 20 observed simulations of *pursuer*, *blocker*, *flanker*, *knocker* and *idle* behaviors were manually input into the sensory input neurons of a predator's controller, and the resulting (motor output) behavior observed. Using this procedure the *pursuer*, *blocker*, *flanker*, *knocker*, and *idle* behaviors were successfully reproduced.

4.6.3 Reproducing Collective Prey-Capture Behaviors

As presented in section 4.5, in multiple simulations of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams, several collective prey-capture behaviors were observed. These prey-capture behaviors were labeled: *entrapment*, *pursuer-blocker*, *spiders-fly* and *role-switcher*. Sections 4.6.1 and 4.6.2 highlighted that five distinct individual predator behaviors were identified by differing sensory-motor value ranges, and that each of these individual behaviors is reproducible. Given this, prey-capture behaviors can be reproduced by first reproducing and then combining at least two individual predator behaviors. Individual behaviors are combined via placing at least two

Tab. 4.4: Behavioral Composition of Fittest Teams. Predators in the fittest teams (evolved by each method) are specialized to one of the identified prey-capture behaviors (entrapment, pursuer-blocker, spider-fly, or role-switcher) or are not specialized to any one prey-capture behavior. NA: Not Applicable.

	Prey-Capture Behavior				
Fittest Team Evolved By:	Entrapment	Pursuer-Blocker	Spider-Fly	Role-Switcher	Non-Specialized
HomCNE (3 predators)	0/3	NA	NA	NA	3/3
HetCNE (3 predators)	0/3	NA	NA	NA	3/3
CCGA (4 predators)	NA	1/4	1/4	NA	2/4
Multi - Agent ESP (4 predators)	NA	2/4	1/4	NA	1/4
CONE (5 predators)	NA	NA	1/5	3/5	1/5

predators preset with either the *pursuer*, *blocker*, *flanker*, *knocker* or *idle* behaviors, within proximity sensor range of each other and light sensor range of a prey. In the case of reproducing the *spiders-fly* prey-capture behavior, the predators and prey are also placed within proximity sensor range of a wall. Using this procedure one is able to reproduce each of the prey-capture behaviors observed in the fittest teams, from a set of constituent individual behaviors. The reproduction of observed prey-capture behaviors indicates that each is comprised of the following individual behaviors.

- *Entrapment*: At least three and at most four *pursuer* predators.
- *Pursuer-blocker*: At least one and at most three *pursuer* predators, and at least two and at most three *blocker* predators.
- *Spiders-fly*: At least one and at most two *blocker* predators, and at least one and at most three *pursuer* predators.
- *Role-switcher*: At least one and at most three *flanker*, at least one and at most three *knocker*, and between zero and three *idle* predators.

4.6.4 Measuring Behavioral Specialization

This section presents the method used to measure if the individual behavior exhibited by any given predator, over the course of its lifetime, is specialized or non-specialized. The behavioral specialization metric is applied to individual predator behaviors that comprise the fittest teams (evolved by each method). The behavioral specialization metric allows each predator in each of the fittest teams to be identified as being *non-specialized* or *specialized* to one of the individual behaviors (identified in section 4.6.2 as *pursuer*, *blocker*, *flanker*, *knocker*, and *idle*). Furthermore, given that a predator (in a fittest team) is found to

be specialized to an individual behavior, the predator is also specialized to one of the prey-capture behaviors. That is, a predator specialized to an identified individual behavior equates with the given predator executing the behavior such that it switches with a low frequency to executing other individual behaviors, and thus infrequently participating in other prey-capture behaviors (which do not include the given individual behavior).

As presented in section 4.6.3, each of the prey-capture behaviors: *pursuer-blocker*, *spider-fly* and *role-switcher*, consists of at least two predators adopting different individual behaviors. The *entrapment* prey-capture behavior consists of at least three predators adopting the same behavioral specialization. Non-specialized predators are considered generalists given that they switch between performing different individual behaviors, or perhaps none of the identified behaviors, with a high frequency. Non-specialized predators thus contribute to multiple different prey-capture behaviors over the course of their lifetimes. In the case of a generalist, there is no particular prey-capture behavior that the predator participates in for the majority of its lifetime. Table 4.4 presents the specialized behavioral composition of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE. For each of the fittest teams (evolved by each method), predators are identified as being specialized to one of the prey-capture behaviors: entrapment, pursuer-blocker, spider-fly or role-switcher, or alternatively not specialized to any prey-capture behavior. Table 4.4 presents the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE as consisting of 3, 3, 4, 4, and 5 predators, respectively. For each of these fittest teams the number of predators specialized (to a prey-capture behavior) or not specialized (to any prey-capture behavior) is presented.

4.6.5 Validating the Role of Behavioral Specialization

This section describes a set of experiments that illustrate that for in order for collective prey-capture behaviors to be effective, these prey-capture behaviors must be composed of a set of complementary specialized individual behaviors. To accomplish this, experiments referred to as behavioral validation experiments evaluate the effectiveness of teams consisting of the same behavioral specialization compared to teams consisting of complementary specialized behaviors. In the behavioral validation experiments, teams are constructed where each predator in the team assumes one of the following behaviors: *pursuer*, *blocker*, *flanker*, *knocker* or *idle*. The method used by the experimenter in order to reproduce each of these individual behaviors is described in section 4.6.2.

Results of the behavioral validation experiments are presented in figure 4.12. Figure 4.12 presents the task performances yielded by teams consisting of predators adopting the same behavior (referred to as *cloned teams* for simplicity) versus teams consisting of predators that adopt different behaviors (referred to as *prey-capture teams*). Each of the cloned and prey-capture teams are tested for team types 1 to 5. An average task performance is calculated over 20 experimental runs for each cloned and prey-capture team.

Only team types 1 to 5 are used since the behavioral validation experiments

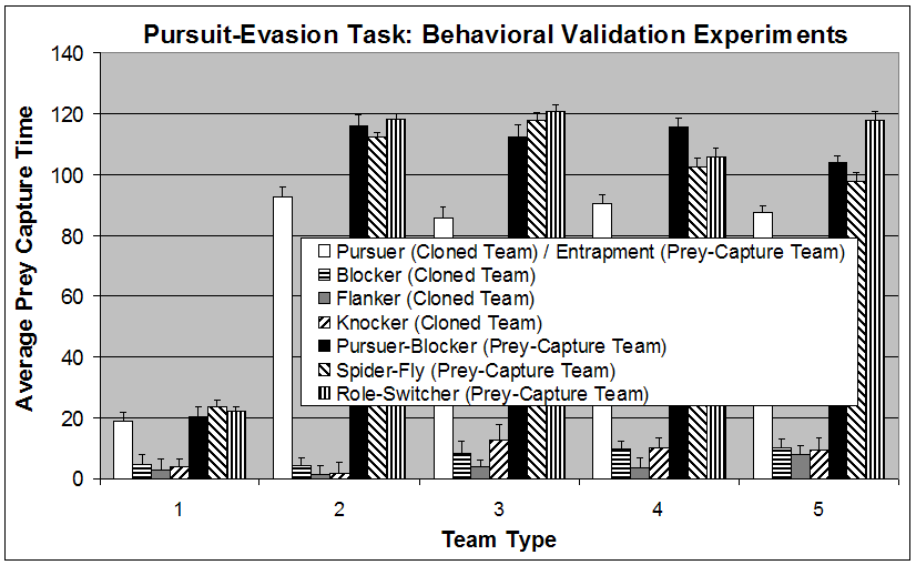

Fig. 4.12: Behavioral Validation Experiments. Comparison of cloned teams and prey-capture teams. See text for explanation.

only test the task performance of predator teams attempting to capture one prey. For each behavioral validation experiment, predators and prey are placed in random locations. However, predators are placed within proximity sensor range of each other and light sensor range of a prey. In the case of the behavioral validation experiments that test the spiders-fly prey-capture behavior, one predator was placed in close proximity to a corner, the prey is placed in close proximity to a wall, and another predator placed within light sensor range of the prey. Cloned teams are defined as follows.

1. *Pursuer-Team*: Consists of predators specialized to the pursuer behavior.
2. *Blocker-Team*: Consists of predators specialized to the blocker behavior.
3. *Flanker-Team*: Consists of predators specialized to the flanker behavior.
4. *Knocker-Team*: Consists of predators specialized to the knocker behavior.

All behavioral validation experiments test prey capture with only one prey (team types 1 to 5) given the ineffectiveness of teams evolved for the task of capturing two prey (section 4.5.5). An idle-team is not presented in the comparison of cloned teams and prey-capture teams presented in figure 4.12, since these teams yield no prey-capture time for all team types. Prey-capture teams are defined, for team types 1 to 5, as follows. For each team type the following team behavioral compositions were selected given that these were the compositions most frequently observed in simulations of the fittest evolved teams.

- *Entrapment-Team*: The same as the cloned team: pursuer-team.
- *Pursuer-Blocker Team*:
  - Team type 1: One *pursuer* and one *blocker* predator.
  - Team type 2: Two *pursuer* predators, and one *blocker* predator.
  - Team type 3: Two *pursuer* and two *blocker* predators.
  - Team type 4: Three *pursuer* and two *blocker* predators.
  - Team type 5: Three *pursuer* and three *blocker* predators.
- *Spiders-Fly Team*:
  - Team type 1: One *pursuer* and one *blocker* predator.
  - Team type 2: Two *pursuer* predators, and one *blocker* predator.
  - Team type 3: Two *pursuer* and two *blocker* predators.
  - Team type 4: Three *pursuer* and two *blocker* predators.
  - Team type 5: Three *pursuer* and three *blocker* predators.
- *Role-Switcher Team*:
  - Team type 1: One *flanker* and one *knocker* predator.
  - Team type 2: Two *flanker* predators, and one *knocker* predator.
  - Team type 3: Two *flanker*, one *knocker* and one *idle* predator.
  - Team type 4: Two *flanker*, two *knocker* and one *idle* predator.
  - Team type 5: Two *flanker*, three *knocker* and two *idle* predators.

A statistical comparison of the task performances yielded by cloned teams versus prey-capture teams, for each team type (presented in figure 4.12) indicates that prey-capture teams (with the exception of the entrapment prey-capture team) yield significantly higher task performances for team types 1 to 5. The *entrapment* team was an exception given that all predators in the team were specialized to the *pursuer* behavior. This made the entrapment prey-capture team the same as the *pursuer* cloned team.

These results indicate that a set of complementary and specialized individual behaviors are required for the *pursuer-blocker*, *spiders-fly*, and *role-switcher*prey-capture behaviors to be effective. Hence, teams consisting of predators using the same behaviors were insufficient for capturing a prey.

4.6.6 Prey-Capture Behavior Lesion Study

Predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams specialized to either the *entrapment*, *pursuer-blocker*, *spider-fly*, or *role-switcher* prey-capture behaviors are systematically removed, and replaced with a non-specialized heuristic controller. The resulting team behavior is then observed. The heuristic controller mandates that a predator

Tab. 4.5: Prey-Capture Behavior Lesion Study. The performance of the fittest teams (evolved by each method), where predators specialized to a given individual behavior are removed and replaced with a non-specialized heuristic controller. Values are percentages of original team task performance. In the first column the names in parentheses are prey-capture behaviors that the given individual behavior contributes to. NA: Not Applicable.

Remove Predators with Behavioral Specialization:	**CONE**	**Multi-Agent ESP**	**CCGA**	**HomCNE**	**HetCNE**
Pursuer (Entrapment / Pursuer-Blocker/Spiders-Fly)	NA	NA	NA	53.4%	55.5%
Blocker (Pursuer-Blocker / Spiders-Fly)	NA	20.0%	60.0%	46.9%	43.7%
Flanker (Role-switcher)	18.9%	26.3%	45.7%	NA	NA
Knocker (Role-switcher)	12.3%	NA	NA	NA	NA
Idle (Role-switcher)	12.3%	NA	NA	NA	NA

moves directly towards a prey when the prey is within light sensor range, or towards a prey's last known location and then stochastically when a prey is beyond light sensor range. Table 4.5 presents the performance of the fittest teams, with predators specialized to individual behaviors removed.

Particular combinations of the individual behaviors: *blocker*, *pursuer*, *flanker*, *knocker*, and *idle* were identified as being the constituent behaviors that form the *entrapment*, *pursuer-blocker*, *spider-fly*, and *role-switcher* collective prey-capture behaviors (section 4.6.3). Removing predators that are specialized to either the: *blocker*, *pursuer*, *flanker*, *knocker*, or *idle* behavior from the fittest teams (evolved by each method), provides an indication of the value of these individual behaviors to the collective prey-capture behaviors. The values in table 4.5 are percentages of the original prey-capture times. These values are calculated over 20 experimental runs. Statistically comparable results are attained with behavioral lesioning of the second and third fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams. As presented in table 4.5, the impact of removing a specialization to any individual behavior is most pronounced in the fittest CONE evolved teams, and less pronounced in the fittest CCGA and Multi-Agent ESP evolved teams, and least pronounced in the fittest HomCNE and HetCNE evolved teams. This result indicates that the fittest CONE evolved team is on average more reliant on its constituent specializations comparative to the fittest teams evolved by related methods.

4.7 Conclusions

This chapter investigated the application of CONE for evolving collective behaviors in a pursuit-evasion case study. The pursuit-evasion task required that a team of simulated *predator* robots evolve a collective *prey capture* behavior in order to immobilize (capture) at least one prey robot.

CONE evolved predator teams yielded a higher average task performance comparative to that yielded by teams evolved by related controller design methods. The highest performing CONE evolved teams used prey-capture behaviors that consisted of a set of complementary and inter-dependent behavioral specializations. An experimental analysis indicated that emergent behavioral specializations made a significant contribution to CONE evolved teams.

The pursuit-evasion case study established that CONE is appropriate for the evolution of collective behaviors, where the exact nature of behavioral specializations required to form an effective collective behavior is initially unknown. The following collective behavior case study, the *multi-rover task*, tests a greater number of controllers in a task where the types of behavioral specializations required for collective behavior task accomplishment, is predefined.

5. COLLECTIVE BEHAVIOR CASE STUDY: MULTI-ROVER

The multi-rover task is a collective behavior task that requires a team of simulated autonomous vehicles (rovers) to derive a collective search behavior that maximizes the value of features of interest (*red rocks*) detected. The term *red rock* refers to discrete high-value features of interest on an unexplored terrain [199]. The term *detected* refers to an instance when a red rock is within range of a rover's *red rock detection* sensors. Detected red rock value is automatically communicated to a base station (*lander*). Discrete and continuous versions of the multi-rover task were introduced by Agogino [2, 3] with the goal of investigating the credit assignment problem. In the multi-rover task described by Agogino [2], rovers select only where to move in a discrete or continuous simulation environment. The multi-rover task described in this chapter differs from that described by Agogino [2] in a number of respects.

1. This is a collective behavior (not a distributed artificial intelligence) task, given that at least two rovers are required in order to accomplish the task.

2. Rovers use detection sensors with variable settings, where as in previous work [2], rovers operate with detection sensors always being active. The detection sensors have a cost-benefit trade-off, and thus provide each rover with the possibility to specialize to a particular sensor setting.

3. In this multi-rover task, task performance evaluation criteria includes the total value of red rocks detected, and the portion of the simulation environment covered by a rover's red rock detection sensors. In previous work [2], task performance was evaluated only according to the total value of red rocks detected. Furthermore, red rock detection could be done individually, and collective behavior was not required.

First, this chapter describes the multi-rover task. This description includes the benefit of behavioral specialization, and how specialization is measured in the multi-rover task. Second, the design of each rover's controller, and its sensors and actuators are described. Third, the experimental design of the multi-rover task is described. Fourth, multi-rover task results are described. The fifth section presents an analysis of the multi-rover task results. The analysis illustrates the contribution of behavioral specialization to task performance. The sixth section presents conclusions drawn from the multi-rover case study.

5.1 Multi-Rover Task

5.1.1 Multi-Rover Simulation Environments

A multi-rover simulation environment is defined by the following features.

1. Classification of the environment as either *simple* or *complex*.
2. Distribution of red rocks of a given *type* (section 5.1.4).

There are five red rock types: [A, B, C, D, E]. At least two rovers are required in order to detect a red rock of any *type*. The red rock detection sensor setting mandated by each rover depends upon the red rock type. Table 5.1 presents the red rock detection sensor settings required for each red rock type to be detected. Environments are classified as either simple or complex as a result of experiments that determined that certain environments are amenable to encouraging emergent behavioral specialization during rover controller evolution. An environment is classified as *simple* if the environment only contains red rocks of one type. Simple environments contain a distribution of only *type E* red rocks. An environment is classified as *complex* if the environment contains red rocks of multiple types. Complex environments contain distributions of *type A*, *type B*, *type C*, and *type D* red rocks.

5.1.2 Rovers in the Simulation Environment

Rovers operate in a continuous simulation environment which is characterized by a two dimensional plane. Only 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 (p), other rovers and red rocks (q), the squared Euclidean norm, bounded by a minimum observation distance δ^2, is used. A minimum distance is to prevent singularities [3] when a rover is very close to a red rock. Equation 6.1 is used as the distance metric.

$$\delta(p, q) = min(\|x - y\|^2, \delta^2) \tag{5.1}$$

5.1.3 Lander (Base station)

The lander has no active role in the detection of red rocks. The purpose of the lander is to receive and record the total value of red rocks detected over the course of each rover's lifetime. Given that a rover detects red rock value at time t, it automatically communicates this value to the lander at time $t+1$. The lander is initialized at a random position in the environment, and remains in this position for the duration of a simulation. It is assumed that all rovers remain within communication range of the lander. Thus the distance to the lander is not an issue for the evolution of rover collective behavior.

Tab. 5.1: Collective Behavior for Red Rock Detection. A given number of rovers are needed to collectively detect a red rock (of a given type). Each rover needs to use a given red rock detection sensor resolution.

Red Rock Type	**Red Rock Value**	**Rovers Required**
A	1	2 Low-Res Detectors
E	1	2 Detectors (using same settings)
B	2	1 Low-Res, 1 Med-Res Detector
C	3	1 Med-Res, 1 Hi-Res Detector
D	4	2 Hi-Res Detectors

5.1.4 Red Rock Distribution

Red rock distributions are defined according to ten *pre-defined* environments. The red rock structures formed by red rock distributions within these environment types are called *red rock canals*. In the experiments conducted by Young *et al.* [198], [200], [199], the term red rock canals is used to denote a set of red rocks that retain a canal like structure. In Young *et al.* [198], [200], [199], it was the goal of rovers to maximize detection of red rocks via following these canals. This was also the motivation for the use of red rock canals for this case study. However, the goal of maximizing red rock detection has been made more complex via introducing collective behavior requirements.

These canals define areas within which red rocks are contained and represent areas that are impassable to rovers. Hence, to move through the environment, rovers must navigate around these red rock canals. The ten canal environments are illustrated in figures 5.1, 5.2, 5.3, 5.4, and 5.5. These ten environments contain varying degrees of structure in the red rock canals. For example, a high degree of structure in environment type 1 (figure 5.1), through to a low degree of structure in environment type 10 (figure 5.5). A low degree of structure refers to sets of red rocks being disjoint and generally not conforming to a canal like structure. A high degree of structure refers to sets of red rocks that are joined and that conform to canal like structures. Red rocks are distributed such that a red rock can placed at each possible x, y position within the confines of a canal. The canal environments were selected from exploratory experiments that found such environments to contain sufficient complexity in order that the most effective rover teams achieved a near optimal task performance, and the least effective teams attained a near zero task performance.

5.1.5 Collective Behavior for Red Rock Detection

Collective behavior is required in order for red rocks to be detected. The probability that a rover detects a red rock is determined by the current red rock detection sensor setting. Table 5.1 indicates that at least two rovers are required in order to detect a red rock of any type. Rovers are required to use the same detection sensor settings in order to detect red rocks of type A, D and

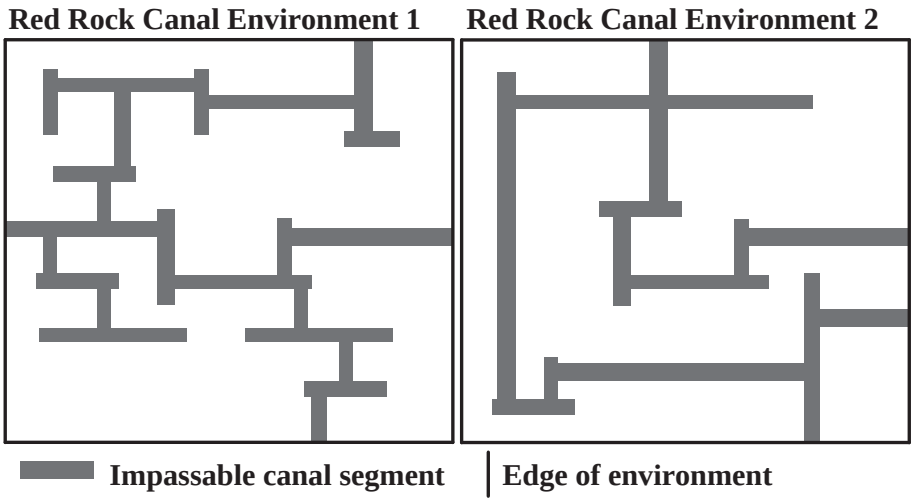

Fig. 5.1: Canal Environments. Environment types [1,2].

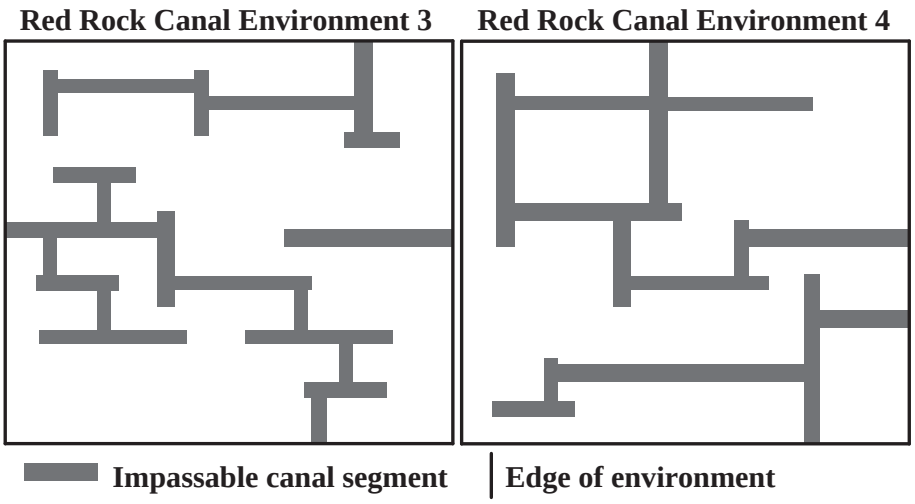

Fig. 5.2: Canal Environments. Environment types [3,4].

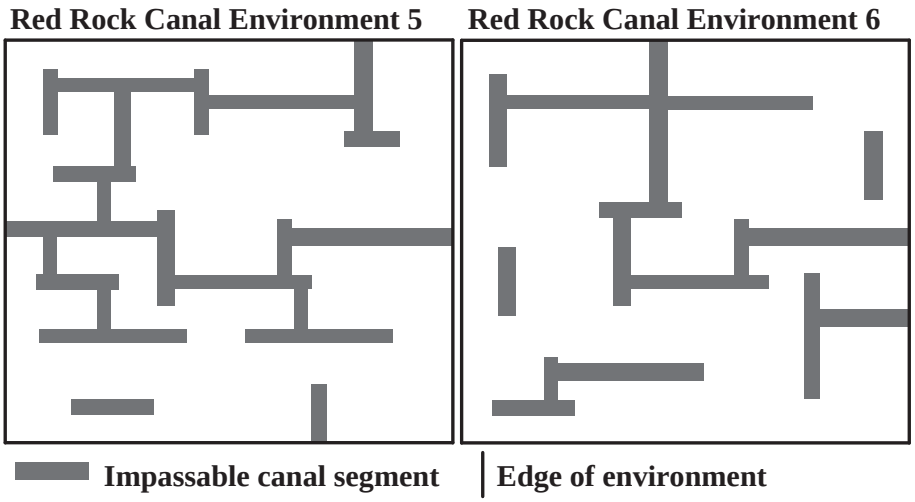

Fig. 5.3: Canal Environments. Environment types [5,6].

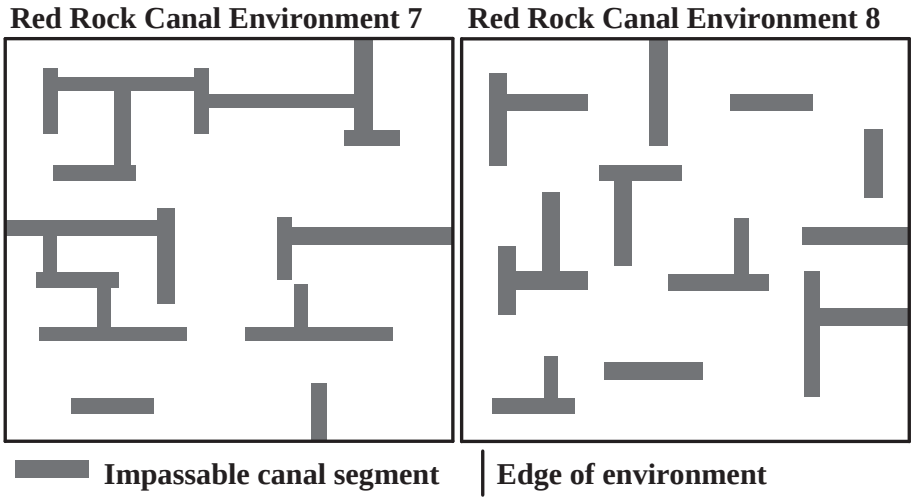

Fig. 5.4: Canal Environments. Environment types [7,8].

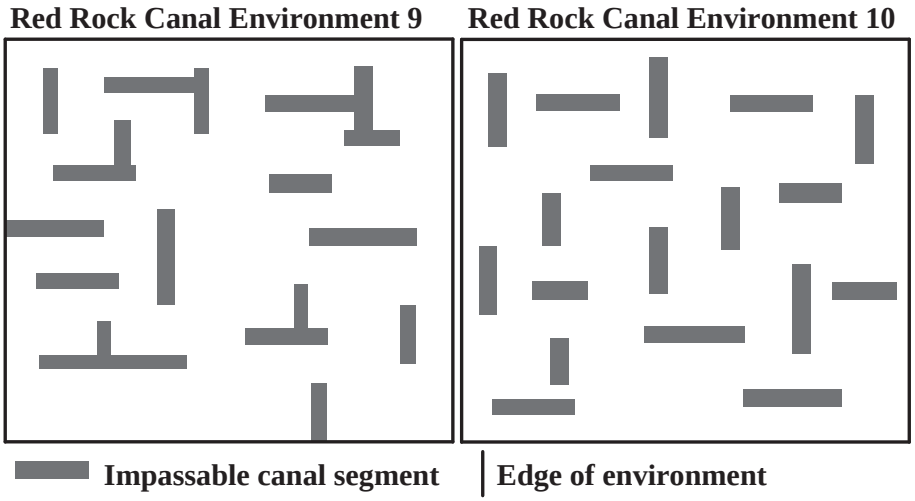

Fig. 5.5: Canal Environments. Environment types [9,10].

Tab. 5.2: Red Rock Detection Sensor Settings. At any simulation iteration a rover can activate its red rock detection sensors with one of three settings.

Detector Sensor Setting	Accuracy	Range	Cost
Low-Res (Setting 0)	1.0	0.05	0.25
Med-Res (Setting 1)	1.0	0.05	0.5
Hi-Res (Setting 2)	1.0	0.05	1.0

E. The requirements specified in table 5.2 for red rock detection were selected from exploratory experiments. These experiments found that mandating that at least two rovers simultaneously using the same sensor settings or complementary settings encouraged the evolution of controllers with different behavioral specializations that operated effectively in the context of a team.

5.2 Rovers

5.2.1 Red Rock Detection Sensors

Each rover is equipped with eight red rock detection sensors ([S-8, S-15] in figure 5.6), where each sensor covers one quadrant in the sensory Field Of View (FOV). Red rock detection sensors need to be explicitly activated. This constitutes one rover action. In the case that a rover activates its red rock detection sensors, then all eight sensors are activated with one of three settings. This provides each rover with a 360 degree FOV (figure 5.6). Red rock detection settings

are: *low-res* (setting 0), *med-res* (setting 1), and *hi-res* (setting 2). The range, cost, and accuracy of the three different red rock detection sensor settings are presented in table 5.2. *Accuracy* is the degree of probability with which red rocks are detected. *Range* is defined as a portion of the environment's size. Given that the length and width are equal in these experiments, sensor range is defined as the length of the side of a square. *Cost* is the amount of energy used each time the red rock detection sensors are activated. When red rocks come within range of a red rock detection sensor, then that sensor is activated with a value inversely proportional to the value of, and distance to, the red rocks. Equation 5.2, for red rock detection sensor q, returns the sum of red rock values in quadrant q, divided by the squared distance to the rover.

$$S_{1(q,t)} = \sum_{j \epsilon J_q} \frac{1}{\delta(L_{v,t}, L_{j,t})} \tag{5.2}$$

Where, q is a sensor quadrant,

v is a rover,

t is simulation time step t,

J_q is the set of all red rock values in quadrant q,

L_j $(j \epsilon J_q)$ is the location of red rock j,

L_v is the location of rover v,

rv_j is the value of red rock j, where $j \epsilon J_q$,

$rv_{j,t} = rv_j$ if $t = d$,

$rv_{j,t} = 0$ if $t \neq d$, where d is the time that red rock j is detected.

5.2.2 Rover Detection Sensors

Each rover is equipped with eight rover detection sensors ([S-0, S-7] in figure 5.6). Rover detection sensors fulfill two functions. First, to prevent collisions between rovers. Second, to provide each rover with an indication of red rock detection sensor settings being used by other rovers within *this* rover's sensory FOV. The eight rover detection sensors are constantly active and have a fixed accuracy, range and cost (table 5.4). As presented in equation 5.3, rover detection sensor q returns a value corresponding to the red rock detection sensor setting being used by the closest rover, divided by the squared distance to *this* rover. Rover detection sensor values are normalized within the range [0.0, 1.0].

$$S_{2(q,v,t)} = \frac{dv'}{\delta(L_{v'}, L_{v,t})} \tag{5.3}$$

Where, q is a sensor quadrant,

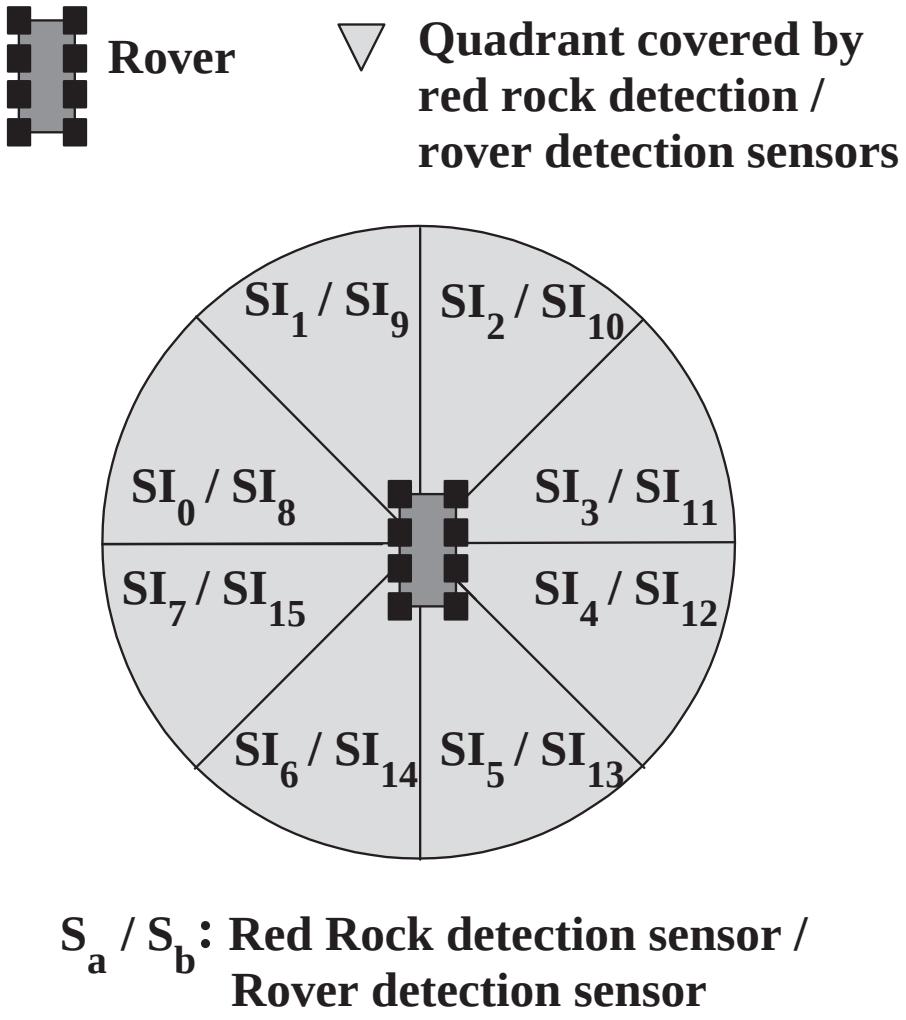

Fig. 5.6: Rover Sensory FOV. A rover's sensor space consists of eight quadrants. One red rock and rover detection sensor covers each quadrant.

v is *this* rover, that is, the rover that is detecting other rovers,

t is simulation time step t,

v' is the closest rover to *this* rover in quadrant q,

dv' is the red rock detection sensor setting of rover *v'*, where *dv'* is a value equal to either 1, 2, or 3, which corresponds to rover *v'* using the *low-res*, *med-res*, or *hi-res* red rock detection sensors setting, respectively.

$L_{v'}$ is the location of the closest rover in sensor quadrant q,

L_v is the location of *this* rover.

5.2.3 Artificial Neural Network Controller

Each rover uses a simple recurrent ANN controller [44] in order to map sensory inputs to motor outputs. The controller uses 26 sensory input neurons fully connected to ten hidden layer neurons (figure 5.7). Five motor output neurons [MO-0, MO-4], are fully connected to the hidden layer neurons. Sensory input neurons [SI-0, SI-7] accept input from each of the eight rover detection sensors. Sensory input neurons [SI-8, SI-15] accept input from each of the eight red rock detection sensors. Sensory input neurons [SI-16, SI-25] accept the previous activation values of each of the hidden layer neurons. The neurons comprising the hidden and output layers are sigmoidal units [70]. All motor output values generated by the rover controller are normalized in the range: [0, 1].

Action Selection

At each simulation iteration, a rover can execute one of four actions. The motor output with the highest value is the action executed.

1. *MO-0:* Activate all red rock detection sensors with setting 0 (figure 5.7).
2. *MO-1:* Activate all red rock detection sensors with setting 1 (figure 5.7).
3. *MO-2:* Activate all red rock detection sensors with setting 2 (figure 5.7).
4. *MO-3, MO-4:* Move in a direction calculated from *dx* and *dy*. If either MO-3 or MO-4 (figure 5.7) is the highest value, then the rover moves.

5.2.4 Movement Actuators

A rover's heading is calculated from the motor output values MO-3 and MO-4 (figure 5.7) which determine the vectors *dx* and *dy*. A rover's heading is determined by normalizing and scaling these vectors by the maximum distance a rover can traverse in one simulation iteration. That is: $dx = d_{max}(o_1 - 0.5)$, and $dy = d_{max}(o_2 - 0.5)$. Where, d_{max} is the maximum distance a rover can move in one simulation iteration, o_1 and o_2 are motor output values MO-3 and MO-4, respectively.

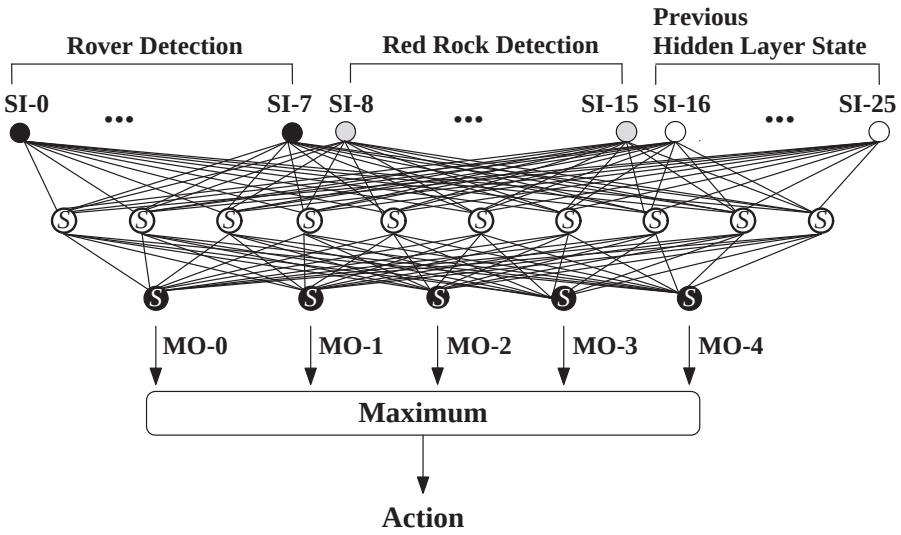

Fig. 5.7: Rover ANN Controller. Not all sensory input neurons are illustrated.

5.2.5 Heuristic Controller

Heuristic controllers are tested in addition to rover teams using ANN controllers. Heuristic controllers are not modified by any adaptive process. Rather, a collective red rock detection behavior is derived via the local interactions of different rovers using different heuristic controllers. Four different heuristic controller types are tested. Each type implements a hard-wired *specialized* or *non-specialized* behavior (Table 5.3). Each controller is defined by a set of probabilistic preferences for selection of one of four actions (the same as those used by the ANN controller) at each simulation iteration. These action selection preference values were derived from a set of exploratory experiments, and were selected given that they produced a specialized behavior. That is, these controllers mandated switching between executing different actions with a low frequency. *Non-specialized* heuristic controllers switched between executing different actions with a high frequency, and had no preference for a specific action.

5.2.6 Specialization in the Multi-Rover Task

In the original rover task [3], each rover could only move at each simulation iteration, and detection sensors were constantly active. It was not possible for different rovers to specialize to different actions. In this multi-rover task, each rover has the possibility of selecting between multiple actions, where each action has an associated benefit and cost. In collective behavior, such specialization potentially increases task performance via exploiting the cost-benefit trade-off of a given action. The sensor and energy constraints of the multi-rover task

Tab. 5.3: *Rover Heuristic Controllers:* Exhibit a specialized (*low-res*, *med-res*, *hi-res* detector) or *non-specialized* behavior. Probabilistic preferences are used for selecting one of four actions at each simulation iteration.

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%

necessitate specialization in order for a team to *detect* an optimal red rock value. These constraints prevent an effective systematic search of an environment. Hence, rovers must use complementary detection sensor settings, where together these settings produce a collective behavior that could not otherwise be attained. This statement is supported by previous research [112], which compared rover teams using hard-wired specialized versus non-specialized behaviors.

Specialization is measured with respect to the behavior exhibited by individual rover controllers, and is defined by applying the behavioral specialization metric (section 3.2.1) to a given rover's lifetime behavior. This metric calculates a degree of specialization (S). Given that $S < 0.5$ for a given rover, it is labeled as *specialized*, and if $S \geq 0.5$ then it is labeled as *non-specialized*. If a rover behavior is specialized, then the label given to the specialization corresponds to the action that is most executed over the course of the course of the rover's lifetime. Specialization labels are assigned at the end of a rover's lifetime, since the time spent executing each action must be known.

- *Low-Res Detector:* Rover specialized to red rock detection(*setting 0*).
- *Med-Res Detector:* Rover specialized to red rock detection (*setting 1*).
- *Hi-Res Detector:* Rover specialized to red rock detection (*setting 2*).
- *Mover:* A rover specialized to moving.

5.3 Multi-Rover Experimental Design

Experiments test 20 rovers together with one lander in the multi-rover task. Experiments measure the impact of a *collective behavior design method* and *environment* upon *red rock value detected* and *area covered* by the red rock detection sensors of a rover team. The experimental objective is to determine which collective behavior design method maximizes the red rock value detected

and area covered. Also, the experiments aim to relate the contribution of emergent behavioral specialization to rover team task performance.

- *Adaptive Collective Behavior Design*: Rover team behavior is adapted with either HomCNE, HetCNE, CCGA, Multi-Agent ESP or CONE.
- *Non-Adaptive Collective Behavior Design*: Each rover uses one of four non-adaptive heuristic controllers.
- *Environment*: Three environment sets labeled *simple*, *complex*, and *extended complex* are tested. Each set contains ten red rock distributions.

Section 5.4 presents task performance results of HomCNE, HetCNE, CCGA, Multi-Agent ESP, CONE evolved teams and heuristic controlled teams.

5.3.1 Rover Team Fitness Evaluation

This section details the global fitness function (G) used to evaluate team performance, and the private fitness function (g_η) used to evaluate individual rover performance. G calculates the total *red rock value detected* multiplied by the portion of the environment's *area covered* by a team's red rock detection sensors. and g_η calculates the same for just rover η. The multiplication of red rock value detected by the area covered is to encourage rovers to explore as wide an area as possible in order to attain a high fitness. 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 performance, based upon the contributions of individual rovers. The private rover fitness function guides the evolution of each rover's controller.

Private Fitness Function

Equation 5.4 presents g_v.

$$g_v = \sum_{0 \leq t \leq T} \sum_{j \epsilon J_{t,v}} \frac{rv_{j,t} A_{v,t}}{min(\delta(L_{v,t}, L_{j,t}))} \tag{5.4}$$

Where, v is a rover,

$rv_{j,t}$ is the value of red rock j at time t,

$J_{t,v}$ is the set of all red rock values within red rock detection sensor range of rover v and detected by rover v,

L_j $(j \epsilon J_q)$ is the location of red rock j,

L_v is the location of rover v,

$min(\delta(L_{v,t}, L_{j,t}))$ is the minimum distance between the location of rover v at time t, and the location of red rock j at time t,

$A_{v,t}$ is the area covered rover v's red rock detection sensors at time t.

Global Fitness Function

Equation 5.5 presents G.

$$G = \sum_{v \epsilon V} g_v \tag{5.5}$$

Where, V is the set of all rovers.

5.3.2 Simulation and Neuro-Evolution Parameters

Tables 5.4 and 5.5 present the simulation and NE parameter settings, respectively. NE parameter settings are those used by the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE methods. Simulation parameters are those used the multi-rover simulator. Any given multi-rover experiment consists of 250 generations. Each generation corresponds to the *lifetime* of each rover in the team. Each lifetime lasts for 10 epochs, where each epoch consists of 2500 simulation iterations. Each epoch represents a task scenario that tests different rover starting positions, and red rock locations (within a given distribution) in the environment. Task performance (red rock value detected and area covered) is calculated as an average taken over all epochs of a rover's lifetime. The best task performance is then selected for each experiment, where an average is calculated over 20 experimental runs. More than 20 runs per experimental setup were not used due to time constraints in running experiments and the time consuming nature of the multi-rover experiments. Less than 20 experimental 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.

Parameter values presented in tables 5.4 and 5.5 were derived in a set of exploratory experiments, which indicated that minor changes to these parameter values produced similar results. Changing the values for *mutation probability*, *fitness stagnation V*, *fitness stagnation W*, *genotype elite portion*, and *initial number of hidden layer neurons* to within 0.20 of the values given in table 5.5 resulted in the evolution of rover teams with a task performance within approximately 0.25 of the task performance results presented in section 5.3. Similar results were attained when the *communication cost*, *movement range*, *movement cost*, *red rock detection sensor range*, *red rock detection sensor cost*, *rover detection sensor range*, *rover detection sensors cost*, and *rover initial energy* parameter values were changed to within 0.30 of the values given in table 5.4.

Changing the iterations per epoch to lower values decreased experiment running time, but did not give rovers enough time to widely explore the environment. Lowering the number of generations and epochs also decreased experiment running time, but did not provide the NE process with sufficient time to derive effective teams. Increasing the number of generations and epochs to within 0.25 of their current values yielded similar teams, and increasing the number of generations and epochs beyond this was considered infeasible due to time limitations and the time consuming nature of these experiments.

Tab. 5.4: Simulation Parameters. Used in each multi-rover simulation.

Multi-Rover Simulation Parameters	
Communication range	1.0
Communication type	Broadcast
Communication cost	0.05
Movement range	0.01
Movement cost	0.01
Red rock detection sensor range	0.05
Red rock detection sensor cost	Variable (section 5.2.1)
Red rock detection sensors accuracy	Variable (section 5.2.1)
Rover detection sensor range	0.02
Rover detection sensors cost	0.05
Rover detection sensors accuracy	1.0
Rover initial energy	1000 units
Initial rover positions	Random
Initial lander position	Random
Environment width	1.0
Environment height	1.0
Individual red rock value	Variable (table 5.6)
Total red rocks in environment	Variable (section 5.6)
Total red rock value in environment	5000
Red rock value distribution	10 Canal environments (section 5.1.4)
Rover lifetime	2500 Iterations

Tab. 5.5: NE Parameters. Used by each NE controller design method.

Rover Neuro-Evolution (NE) Parameter Settings	
Generations	250
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	15 Generations (CONE)
Fitness stagnation W	10 Generations (CONE)
Genotype distance value	[0.0, 1.0] (CONE)
Genotype Distance (GD)	[0.0, 1.0] (CONE)
Specialization Distance (SD)	[0.0, 1.0] (CONE)
Genotype population elite portion	50%
Weight (gene) range	[-10.0, +10.0]
Crossover	Single point
Sensory input neurons	26
Hidden layer neurons (Initial number)	10
Motor output neurons	5
Genotype	Input-output weights: Neuron (Multi-Agent ESP, CONE), All weights: ANN controller (HomCNE, HetCNE, CCGA)
Genotype representation	Floating point value vector
Total genotypes	10000
Genotype populations	20 (CONE, Multi-Agent ESP, CCGA), 1 (HomCNE, HetCNE)
Genotype length	31 (CONE, Multi-Agent ESP), 310 (HomCNE, HetCNE, CCGA)
Genotypes per population	500 (CONE, Multi-Agent ESP, CCGA), 10000 (HomCNE, HetCNE)

5.3.3 Experimental Setups for Neuro-Evolution Methods

This section describes the experimental setup used for the *HomCNE*, *HetCNE*, *CCGA*, *Multi Agent-ESP* and *CONE* methods.

Evolving Red Rock Detection Behavior with Homogenous CNE

The process of *Homogenous Conventional Neuro-Evolution* (HomCNE) is illustrated in figure 2.3, and described in section 2.3.1. For a team of 20 rovers, the population is initialized with 2400 randomly generated genotypes. Each genotype represents the hidden layer connection weights of one rover ANN controller. Each genotype is encoded as vector of 162 floating point values. That is, 22 sensory inputs plus five motor outputs multiplied by six hidden layer neurons (table 5.5). A rover controller is derived via randomly selecting one genotype from the population's elite portion (table 5.5). This selected genotype is then replicated 20 times in order to create a team of rover clones.

Evolving Red Rock Detection Behavior with Heterogenous CNE

The process of *Heterogenous Conventional Neuro-Evolution* (HetCNE) is illustrated in figure 2.3, and described in section 2.3.1. The experimental setup of the HetCNE method is the same as that used for HomCNE, except that a rover controller is derived via randomly selecting 20 genotypes from the population's elite portion such that no genotype is selected more than once. The selected genotypes are then decoded into a team of 20 controllers (rovers).

Evolving Red Rock Detection Behavior with CCGA

For a team of 20 rovers, 20 genotype populations are initialized. Each population uses 120 randomly generated genotypes. Each genotype is encoded as vector of 162 floating point values. That is, 22 sensory inputs plus five motor outputs multiplied by six hidden layer neurons (table 5.5). A rover controller is derived via randomly selecting one genotype from the elite portion of each population. This process is repeated 20 times, in order to derive 20 different controllers. The CCGA process is described in section 4.4.1.

Evolving Red Rock Detection Behavior with Multi-Agent ESP/CONE

For a team of 20 rovers, both Multi-Agent ESP and CONE create 20 genotype populations. Population i consists of u sub-populations, where u (in this case $u = 6$) corresponds to the number of hidden layer neurons used by a controller derived from population i. Each population is initialized with 120 genotypes. Each genotype is encoded as vector of 27 floating point values. That is, 22 sensory inputs plus five motor outputs (table 5.5). A controller is derived via randomly selecting one genotype from the elite portion of each sub-population for a given population. These genotypes then collectively form the hidden layer of one

controller. This process is repeated 20 times in order to derive 20 different controllers. Specific to CONE, the number of generations (V in section 3.7) which fitness progress can stagnate within one or more populations (n controllers) before the GD value is adapted (section 3.1.2) is presented as *fitness stagnation* V in table 5.5. The number of generations (W in section 3.7) which fitness can stagnate in any given population before the number of sub-populations is adapted (section 3.6), is presented as *fitness stagnation* W in table 5.5.

5.4 Multi-Rover Task Results

This section describes task performance results yielded from three multi-rover experiment sets. *Caste* refers to a set of rovers within a given team that are *specialized* to the same behavioral role [83].

1. *Experiment Set 1:* Illustrates that *complex environments*[1] environments are appropriate for encouraging behavioral specialization during controller evolution. Experiment set 1 compares task performance results *between* teams evolved in *simple*[2] and *complex environments*.

2. *Experiment Set 2:* Investigates the research hypotheses (section 1.2) via indicating that CONE evolves teams that yield a higher task performance comparative to the task performance yielded by teams evolved by related methods. These related methods are: HomCNE, HetCNE, CCGA, and Multi-Agent ESP. Experiment set 2 compares the task performance results of teams evolved *within* the *complex environment set*.

3. *Experiment Set 3:* Investigates the research hypotheses (section 1.2) via elucidating that CONE is consistently able to derive a level of behavioral specialization as mandated by the task and the environment. Experiment set 3 shows that specialization results in a higher team task performance comparative to related methods. Experiment set 3 compares the task performance results between teams evolved *within* the *extended complex environment set* (an extension of the complex environment set).

One experiment consists of the evolving teams using each NE method in a given environment. Each experiment consists of two distinct phases.

- *Evolution phase:* The controllers of a rover team are evolved for 250 generations (table 5.5) using a given NE method, and a given environment.

- *Testing phase:* The fittest n controllers (team) are selected and set to execute in the same environment for one rover lifetime. The testing phase does not evolve controllers, so during this phase each the evolved connection weights of each controller remains static. Task performance results presented in this chapter are averages calculated over 20 runs of the fittest controllers in a test environment.

[1] The terms *complex environment set* and *complex environments* are used interchangeably.
[2] The terms *simple environment set* and *simple environments* are used interchangeably.

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

- The Kolmogorov-Smirnov test [48] 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 task performance results of any two teams evolved by given methods, an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ.

The results of all statistical comparisons conducted in this chapter are presented in appendix C.

5.4.1 Experiment Set 1: Environments Appropriate for Behavioral Specialization

This experiment set tests the evolution of teams in two sets of simulation environments. It is hypothesized that the first environment set does not encourage emergent behavioral specialization, and the second environment set does encourage specialization, during controller evolution.

- *Simple environment set*: Contains ten *type E* red rock distributions. *Type E red rocks* are detectable by sensors operating at any setting (section 5.2.1). Given this, it is supposed that this environment set is not appropriate for encouraging specialization during controller evolution.
- *Complex environment set*[3]: Contains ten distributions of *type [A, B, C, D] red rocks*. *Type [A, B, C, D] red rocks* are only detectable by at least two rovers using complementary red rock detection sensor settings (section 5.1.5). Given this, it is supposed that this environment set is appropriate for encouraging behavioral specialization.

Teams Evolved in Simple Environments

Rover teams are evolved using the HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE methods, for each of the ten *simple environments*. For each environment, the average *task performance* yielded by evolved teams is calculated over 20 experimental runs. Figures 5.8 and 5.9 present the average *red rock value detected* and *area covered*, respectively, by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams operating in each of the *simple environments*. The behavioral composition of the fittest team evolved by each method, for each simple environment, are presented in appendix C. Behavioral composition refers to the composite number of rovers in a team that are *non-specialized* or *specialized* to activating red rock detection sensors with either the *low-res*, *med-res*, or *hi-res* settings.

[3] The terms *complex environment set* and *complex environments* are used interchangeably.

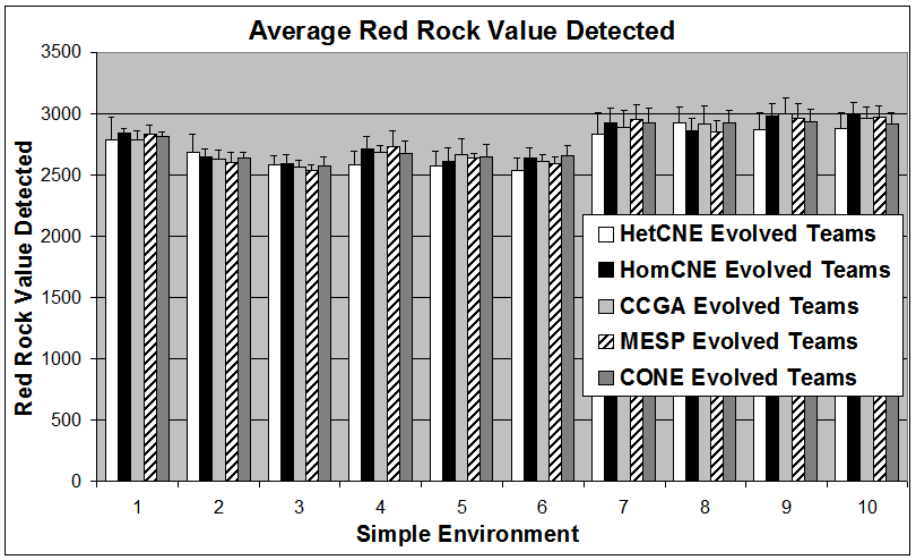

Fig. 5.8: Average Red Rock Value Detected in Simple Environments. Comparative results yielded by teams evolved by each method. Note the scale of the *red rock value detected* axis extends to only 3500.

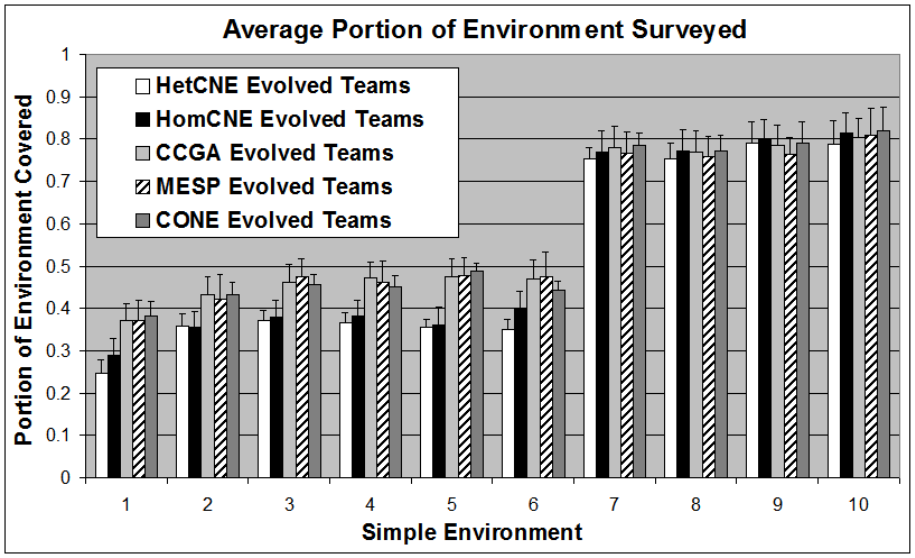

Fig. 5.9: Average Area Covered in Simple Environments. Comparative results yielded by HomCNE, HetCNE, CCGA, MESP, and CONE evolved teams.

Teams Evolved in Complex Environments

Figures 5.10 and 5.11 present the average *red rock value detected* and *area covered*, respectively, for teams evolved by each method in the *complex environments*. Behavioral compositions of the fittest teams evolved by each method, in the complex environments, are presented in appendix C.

Comparing Task Performance Results of Teams Evolved in Simple and Complex Environments

A statistical comparison of task performance results yielded by teams evolved by each method in the *simple* and *complex* environment sets, is conducted.

- First, a comparison is conducted between the *average red rock value detected* by teams evolved with respective methods in the *simple* (figure 5.8) and *complex* (figure 5.10) environments.
- Second, a comparison is conducted between the *average area covered* by teams evolved with respective methods in the *simple* (figure 5.9) and *complex* (figure 5.10) environments.

Data sets of teams evolved by each method (presented in figures 5.8, 5.9, 5.10, and 5.11) were found to conform to normal distributions via applying the Kolmogorov-Smirnov test. A statistical comparison between the average *red rock value detected* and *area covered* for teams evolved in the *simple* and *complex* environments is presented in appendix C.

Results Summary: Teams Evolved in Simple and Complex Environments

A Comparison of task performances yielded by teams evolved in the *simple* and *complex* environment sets, indicate the following results.

1. The *average red rock value detected* by teams evolved by CCGA, Multi-Agent ESP, and CONE in *complex environments* is significantly higher than the *average red rock value detected* by teams evolved by CCGA, Multi-Agent ESP, and CONE in *simple environments*. This result indicates that the complex environments are appropriate for evolving teams using CCGA, Multi-Agent ESP, and CONE, given that these teams yield a higher average red rock value detected comparative to teams evolved by the same methods in 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 the complex environments are appropriate for encouraging the evolution of multiple complementary *castes*. In the case of teams evolved by HomCNE and HetCNE in simple and complex environments, teams consisting of a single rover caste were evolved.

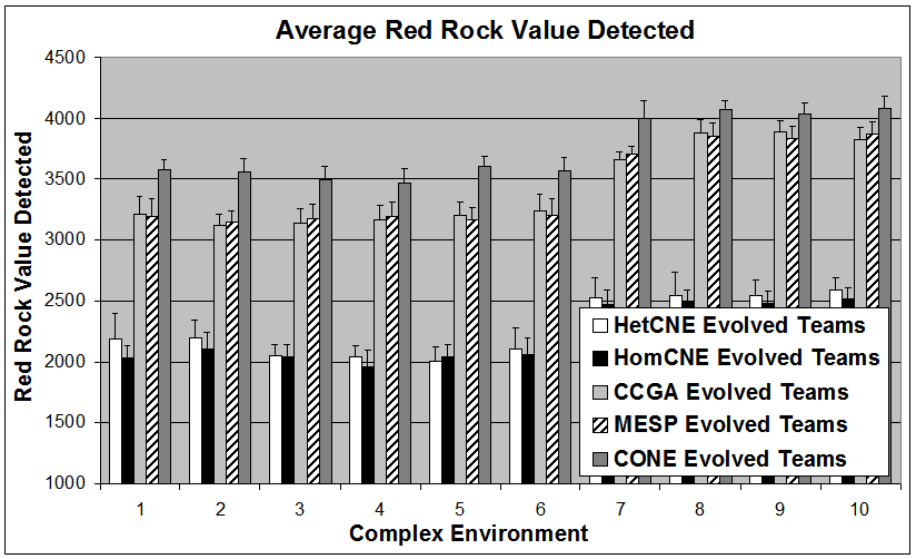

Fig. 5.10: Average Red Rock Value Detected in the Complex Environments. Comparative results yielded by teams evolved by each method. Note the scale of the *red rock value detected* axis extends to only 4500.

3. There is no significant difference between the *average red rock value detected* by teams evolved by HomCNE and HetCNE in both *simple* and *complex* environments.

4. There is no significant difference between the *average area covered* by teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE in a majority of the *simple* and *complex* environments.

Furthermore, results 1 through to 4 indicate that the complex environments are appropriate for evolving teams composed of complementary behavioral specializations, where such specializations result in a higher task performance comparative to teams evolved in simple environments. Further analysis of these results is present in section 5.5.1.

5.4.2 Experiment Set 2: Teams Evolved within the Complex Environment Set

This section presents the results of applying HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in order to evolve teams in the *complex environments*. Figures 5.10 and 5.11 present the average *red rock value detected* and *area covered*, respectively, for teams evolved by each method in the *complex environment set*. The behavioral compositions of the fittest teams evolved by each method in the *complex environments* are presented in appendix C.

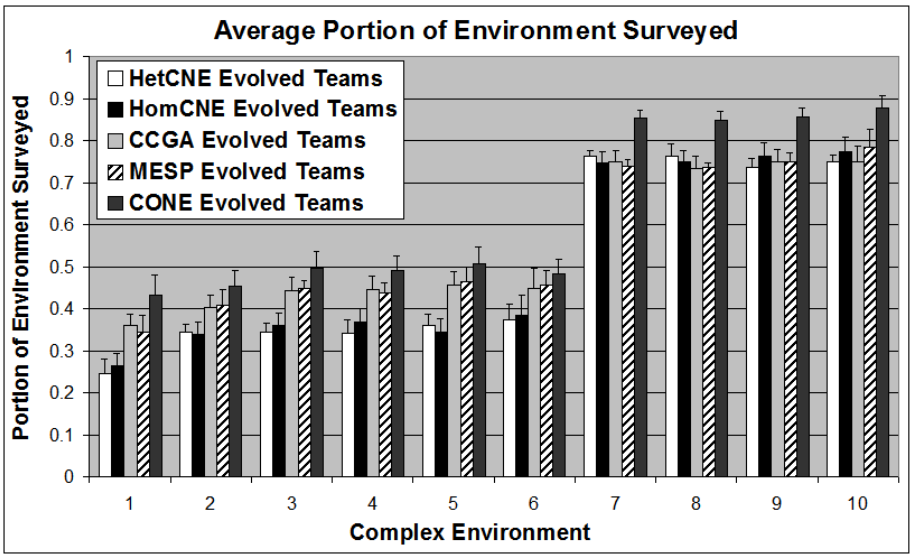

Fig. 5.11: Average Area Covered in the Complex Environments. Comparative results yielded by teams evolved by each method.

Comparing Teams Evolved in the Complex Environment Set

A statistical comparison of task performance results yielded by teams evolved using each method, in the complex environments, is conducted.

- First, a comparison is conducted between the *average red rock value detected* by teams evolved with HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in the *complex* environments (figure 5.10).
- Second, a comparison is conducted between the *average area covered* by teams evolved with HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in the *complex* environments (figure 5.11).

Appendix C presents statistical test results conducted for a comparison of average *red rock value detected* and *area covered* by teams evolved by each method in *complex environments*.

Results Summary: Comparing Teams Evolved in Complex Environments

Comparative task performances of teams evolved by each method in the *complex environments*, indicate the following results.

1. For all complex environments, CONE evolved teams attained a significantly higher average task performance, comparative to HomCNE, HetCNE, CCGA, and Multi-Agent ESP evolved teams. CCGA, Multi-Agent ESP, and CONE evolved teams are comprised of complementary castes.

2. For each complex environment, HomCNE and HetCNE evolved teams comprised of a single specialized or non-specialized caste.

3. For all complex environments, teams evolved by CCGA and Multi-Agent ESP yield a comparable task performance.

4. For all complex environments, teams evolved by CCGA, Multi-Agent ESP, and CONE yield a significantly higher average red rock value detected comparative to HomCNE and HetCNE evolved teams.

5. For complex environments 7 and 9, teams evolved by HomCNE and CCGA yielded a comparable area covered.

6. For complex environments [7, 10], teams evolved by HomCNE and MESP yielded a comparable area covered.

7. For complex environments [7, 8], teams evolved by HetCNE and CCGA yielded a comparable area covered.

8. For complex environments 7 and 9, teams evolved by HetCNE and MESP yielded a comparable area covered.

Results 1 and 2 support the hypothesis that CONE facilitates an appropriate level of behavioral specialization, where such specialization results in a higher task performance, comparative to related methods. Result 3 indicates that the HomCNE and HetCNE methods are not appropriate for deriving teams that consist of a set of complementary castes. Result 4 indicates that the CCGA and Multi-Agent ESP evolve teams with comparable task performances in a majority of complex (nine out of ten) environments. Result 5 indicates that the cooperative co-evolutionary methods (CCGA, Multi-Agent ESP, and CONE), are effectively able to evolve teams that yield a significantly higher red rock value detected comparative to the single population conventional NE methods (HomCNE and HetCNE) in all complex environments. Results 6 through to 9 indicate that the CCGA and Multi-Agent ESP yield no advantage over the CNE methods in terms of area covered by evolved teams.

5.4.3 Experiment Set 3: Teams Evolved within the Extended Complex Environment Set

This section presents the results of applying HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE to evolve rover teams in an *extended complex environment set*. The purpose of these environments was to test the impact of different quantities of each red rock type upon controller evolution. The complex environments contained an equal number of each red rock type (table 5.6). The extended complex environments contain different combinations of quantities of type [A, B, C, D] red rocks (table 5.6). Also, in the extended complex environments the location of red rocks in each environment is the same as *environment 10* in the simple and complex environments (figure 5.5). Environment 10 was

Tab. 5.6: *Distribution of Red Rock Types per Environment Set:* Each environment is defined by a different red rock distribution. *ENV-SET:* Environment Set. *SM:* Simple Environment Set. *CM:* Complex Environment Set. *ECM-x:* Extended Complex Environment *x*.

ENV-SET	Type-A Red Rocks	Type-B Red Rocks	Type-C Red Rocks	Type-D Red Rocks	Type-E Red Rocks	Red Rock Value
CM	0	0	0	0	5000	5000
SM	500	500	500	500	0	5000
ECM-1	3502	250	166	125	0	5000
ECM-2	500	1750	166	125	0	5000
ECM-3	502	250	1166	125	0	5000
ECM-4	502	250	166	875	0	5000
ECM-5	2000	500	250	125	0	5000
ECM-6	1001	1000	333	250	0	5000
ECM-7	1002	500	666	250	0	5000
ECM-8	1001	500	333	500	0	5000
ECM-9	2500	1250	0	0	0	5000
ECM-10	1	0	833	625	0	5000

selected given that the highest average task performance was achieved in this environment for teams evolved by all methods (figures 5.10 and 5.11). The total value of all red rocks sums to 5000 (table 5.6). As an extension to the complex environments, the goal was to ascertain if specialized behaviors would still emerge in environments where the quantity of each red rock type varies between environments, but the structure and distribution of red rocks remains the same.

Teams Evolved in Extended Complex Environments

Rover teams are evolved using HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE, for each extended complex environment. For each environment, the average *task performance* yielded by evolved teams is calculated over 20 experimental runs. Figures 5.12 and 5.13 present the average *red rock value detected* and *area covered*, respectively, for teams evolved by each method. Behavioral compositions of the fittest teams evolved by each method in the extended complex environments are presented in appendix C.

Comparing Results of Teams Evolved in Extended Complex Environments

A statistical comparison between the task performance results yielded by teams evolved in the extended complex environment set is conducted.

- First, a comparison is conducted between the *average red rock value detected* by teams evolved with HomCNE, HetCNE, CCGA, Multi-Agent

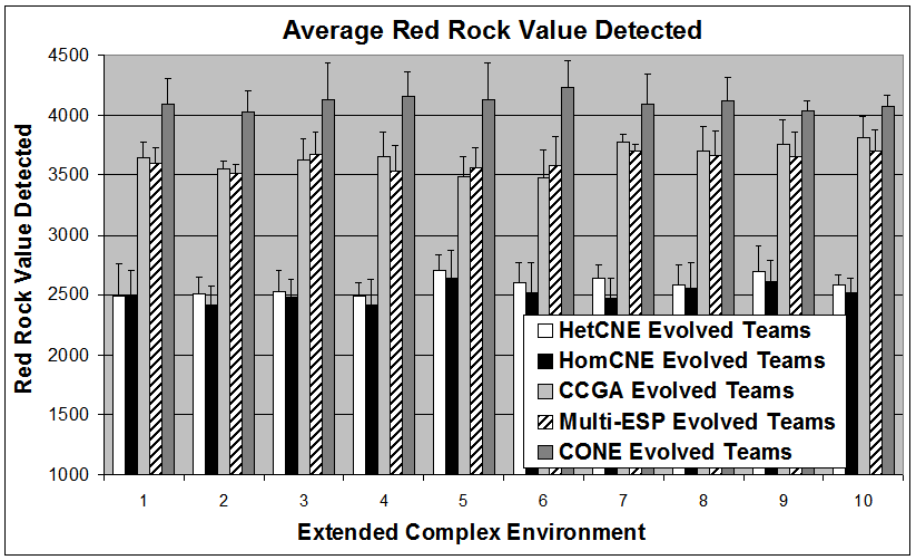

Fig. 5.12: Average Red Rock Value Detected in Extended Complex Environments. Comparative results yielded by teams evolved by each method. Note the scale of the *red rock value detected* axis extends to only 4500.

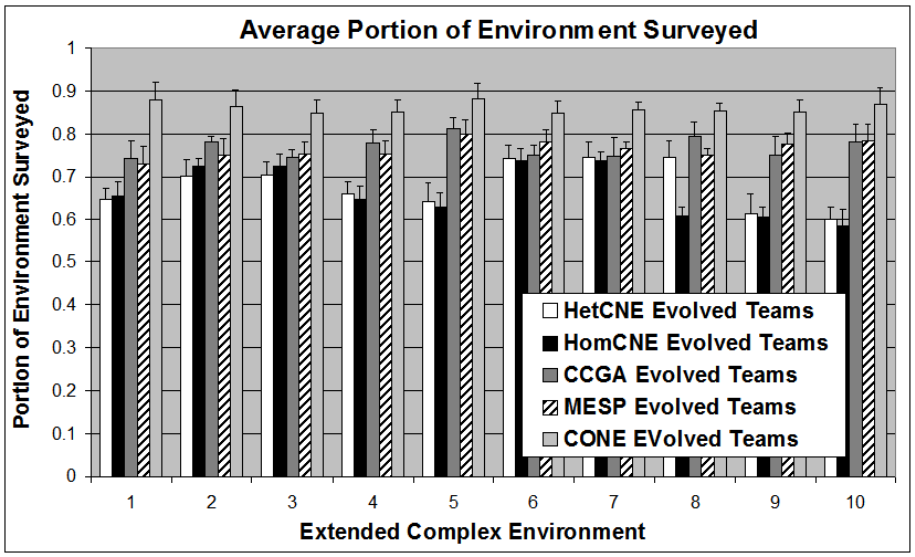

Fig. 5.13: Average Area Covered in Extended Complex Environments. Comparative results yielded by teams evolved by each method.

ESP and CONE in the *extended complex* environment set (figure 5.12).

- Second, a comparison is conducted between the *average area covered* by teams evolved with HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in the *extended complex* environment set (figure 5.13).

Data sets of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams presented in figures 5.12 and 5.13, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test. Appendix C presents statistical test results for comparisons between the *average red rock value detected* by teams evolved by each method.

Results Summary: Rover Teams Evolved in Extended Complex Environments

Statistical comparisons of task performances for teams evolved in the extended complex environments, indicate the following results.

1. The *average red rock value detected* by CONE evolved teams is significantly higher than that of teams evolved by comparative methods.
2. Teams evolved by CCGA, Multi-Agent ESP and CONE are comprised of complementary castes. Behavioral compositions of the fittest evolved teams are presented in appendix C.
3. Teams evolved by CCGA and Multi-Agent ESP yield a comparable red rock value detected. However, there is a significant difference between the area covered by these teams for environments [1, 3, 5, 7, 10].
4. Area covered by HomCNE evolved teams is significantly lower comparative to that of teams evolved by CCGA in environments [3, 6, 7].
5. Area covered by HetCNE evolved teams is significantly lower comparative to that of teams evolved by CCGA in environments [6, 7].
6. Area covered by HomCNE evolved teams is significantly lower comparative to that of Multi-Agent ESP evolved teams in environments 3 and 7.
7. Area covered by HetCNE evolved teams is significantly lower comparative to that of Multi-Agent ESP evolved teams in environments [7, 8].
8. Teams evolved by CCGA, Multi-Agent ESP, and CONE yield a significantly higher average *red rock value detected* comparative to HomCNE and HetCNE evolved teams. Also, HomCNE and HetCNE evolves teams comprised of either a single specialized or non-specialized caste.

Results 1 and 2 support the hypotheses that CONE is appropriate for facilitating emergent behavioral specialization, where such specialization results in a higher task performance comparative to related methods. Result 3 indicates that the CCGA and Multi-Agent ESP methods yield a comparable task performance for a majority (nine out of ten) extended complex environments. Results 4 to 8 indicate that CCGA, Multi-Agent ESP, and CONE out-perform the CNE methods in all extended complex environments.

5.5 Discussion of Multi-Rover Experimental Results

This analysis investigates the hypothesis that CONE facilitates a degree of behavioral specialization appropriate for achieving a higher task performance comparative to related methods. Sections 5.5.1, 5.5.2, 5.5.3, and 5.5.4 describe an analysis that evaluates the contribution of emergent behavioral specialization to collective red rock detection behaviors. Section 7.2.2 describes an analysis of the function and contributions of the *genotype* and *specialization* metrics to emergent behavioral specialization in CONE evolved teams.

5.5.1 Behavioral Specialization in the Multi-Rover Task

In order to illustrate that emergent specialization is beneficial in the multi-rover task, a set of experiments, conducted in section 5.4.1, highlighted that the *complex environments* were appropriate for encouraging behavioral specialization during controller evolution. The fittest teams evolved by CCGA, Multi-Agent ESP and CONE in the complex environments consisted of multiple complementary castes. However, the emergence of such castes did not occur in the fittest HomCNE and HetCNE evolved teams. Furthermore, the red rock value detected by teams evolved by CCGA, Multi-Agent ESP and CONE in the complex environments was significantly higher comparative to the red rock value detected by teams evolved in the simple environments. This indicates that the multiple population architecture and cooperative co-evolutionary nature of CCGA, Multi-Agent ESP, and CONE are better suited for attaining collective behavior solutions in the complex environments. That is, the complex environments serve to encourage emergent specialization and increases the red rock value detected by such teams. This is supported by the multiple complementary castes derived by CCGA, Multi-Agent ESP and CONE (appendix C) in the complex environments, and the corresponding task performances of the fittest teams evolved by CCGA, Multi-Agent ESP and CONE in these environments.

The advantage of applying CCGA, Multi-Agent ESP and CONE to the multi-rover task in the complex environments, is evident from examining the mechanism used for genotype selection and the multiple population architecture. The CCGA, Multi-Agent ESP and CONE methods construct a team of n rovers via selecting and decoding a single genotype from each of n populations. In a cooperative co-evolutionary process, the controllers corresponding to these genotypes are evaluated in the multi-rover task. Different fitness values are assigned to different genotypes based upon the success of corresponding controllers over the course of rover lifetimes. This encourages the evolution of complementary (and sometimes specialized) behaviors.

This is not the case for rover teams evolved by HomCNE and HetCNE in the complex environments. The HomCNE method constructs a team of n rovers via selecting, decoding and cloning a single genotype n times from one population. The HetCNE method constructs a team of n rovers via selecting, decoding n different genotypes from one population. Genotype selection from a single population encourages all controllers in a team to converge to a single behavior

that effectively accomplishes the multi-rover task. This statement is supported and exemplified by the emergent single caste in the fittest teams evolved by HomCNE and HetCNE in the complex environments (appendix C), and the comparatively low task performances of HomCNE and HetCNE evolved teams (figures 5.10 and 5.8).

5.5.2 Analysis of Evolution in Complex Environments

Task performance results presented in figures 5.10, 5.11, 5.12, and 5.13 and the supporting statistical comparison presented in appendix C indicate the average task performance of CONE evolved teams is significantly higher than the average task performance of HomCNE, HetCNE, CCGA, and Multi-Agent ESP evolved teams. The fittest team evolved by CONE for each environment is comprised of one non-specialized and multiple specialized castes. Specialized castes are those rovers specialized to *low-res*, *med-res*, and *hi-res* red rock detection behavior. This result supports the hypothesis that CONE is appropriate for evolving behavioral specialization such that a higher collective behavior task performance, comparative to related methods, is achieved.

Also, the fittest teams evolved by CCGA and Multi-Agent ESP in each complex environment (appendix C) are similarly comprised of one non-specialized caste and a combination of either low-res, med-res, or hi-res detector castes. The evolution of such castes in the fittest teams evolved by CCGA, Multi-Agent ESP and CONE in complex environments, was a consequence of the multi-population architecture, the mechanism to select genotypes and create rover teams, and the multi-rover task and environment constraints.

The Role of Castes

The emergence of *non-specialized* and *specialized* castes in the fittest CCGA, Multi-Agent ESP, and CONE evolved teams is attributed to the multi-rover task collective behavior requirement. At least two rovers are required in order to detect a red rock. In order for rovers to detect an optimal value of red rocks, teams are required to adopt specializations to different detection sensor settings. This in turn results in the emergence of low-res, med-res, hi-res detector and non-specialized castes. There is a non-specialized caste in each of the fittest teams for each complex environment (appendix C). This non-specialized caste is a result of rovers that frequently switch between executing low-res, med-res, and hi-res red rock detection, as well as move behaviors.

It is theorized that the non-specialized caste complements the low-res, med-res, and hi-res castes for the purpose of detecting a high value of red rocks. When coupled with a specialized rover, a non-specialized rover provides the necessary second detection sensor setting in order that a red rock can be detected. This is supported by the *caste lesion study* presented in section 5.5.3. The caste lesion study elucidates that castes provide behavioral contributions that are essential in order for the fittest CCGA, Multi-Agent ESP and CONE evolved teams to achieve a near optimal task performance.

5.5.3 Rover Caste Lesion Study

The goal of the caste lesion study is to ascertain the contribution of specialized and non-specialized castes to the task performance of the fittest teams. The caste lesion study evaluates the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE via systematically removing specialized and non-specialized castes and replacing them with specialized or non-specialized heuristic controllers (section 5.2.5). Each lesioned team is then executed in 20 new experimental runs for each complex environment. An average *red rock value detected* and *area covered* is calculated over these 20 experimental runs for each environment. The remainder of this section describes the procedure followed for conducting a lesion study for each of the fittest teams evolved by each method. 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. In the following *env* denotes *environments*.

- *Fittest Teams Evolved by HomCNE:*
  - Env [1, 6]: *Med-res* detector caste is replaced with 20 med-res heuristic controllers (table 5.3). These teams are executed in env [1, 6].
  - Env [7, 10]: *Hi-res* caste is replaced with 20 hi-res heuristic controllers (table 5.3). These teams are executed in env [7, 10].
- *Fittest Teams Evolved by HetCNE:* Env [1, 10]: *Hi-res* detector caste is replaced with 20 hi-res heuristic controllers, and executed in env [1, 10].
- *Fittest Teams Evolved by CCGA:*
  - Env [1, 4]: *Low-res* detector caste is replaced with low-res heuristic controllers (table 5.3). These teams are executed in env [1, 4].
  - Env [1, 8]: *Med-res* detector caste is replaced with med-res heuristic controllers (table 5.3). These teams are executed in env [1, 8].
  - Env [1, 10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers. These teams are executed in env [1, 10].
  - Env [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers (table 5.3), and executed in env [1, 10].
- *Fittest Teams Evolved by Multi-Agent ESP:*
  - Env [1, 3]: *Low-res* detector caste is replaced with low-res heuristic controllers. These teams are executed in env [1, 3].
  - Env [1, 10]: *Med-res* detector caste is replaced with med-res heuristic controllers. These teams are executed in env [1, 10].
  - Env [1, 10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers. These teams are executed in env [1, 10].

  - Env [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers. These teams are executed in env [1, 10].

- *Fittest Teams Evolved by CONE:*
  - Env [1, 10]: *Low-res* detector caste is replaced with low-res heuristic controllers. These teams are executed in env [1, 10].
  - Env [1, 10]: *Med-res* detector caste is replaced with med-res heuristic controllers. These teams are executed in env [1, 10].
  - Env [1, 10]: *Hi-res* detector caste is replaced with hi-res heuristic controllers. These teams are executed in env [1, 10].
  - Env [1, 10]: *Non-specialized* caste is replaced with non-specialized heuristic controllers. These teams are executed in env [1, 10].

These 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 and the interaction of these castes. These results also indicate that the fittest CONE evolved teams are more reliant upon the constituent specialized and non-specialized castes, comparative to the fittest CCGA and Multi-Agent ESP evolved teams evolved in the same environments. The task 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 overall 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 of an 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 of castes. Further analysis of the inter-dependency between the specialized and non-specialized castes is presented as part of the behavioral validation study in section 5.5.4. The smallest difference between original task performance and the task performance of teams with a caste removed was measured for the fittest HomCNE and HetCNE evolved teams. For each complex environment, the task performances yielded by lesioned HomCNE and HetCNE teams, was lower comparative to the original task performance yielded by the fittest evolved teams. For both HomCNE and HetCNE lesioned teams, the reduction in task performance was not as large as that measured for the fittest CCGA, Multi-Agent ESP and CONE evolved teams, re-evaluated with removed castes. This result supports previous work that indicated that a rover team comprised of multiple complementary castes is mandated in order to achieve a near optimal task performance.

5.5.4 Validating the Role of Behavioral Specialization

The purpose of the behavioral validation experiments is to test, in isolated experiments, each of the rover behavioral specializations. Teams consisting entirely of rovers specialized to one behavior are tested with the goal of demonstrating that such teams are insufficient for attaining an optimal task performance.

Tab. 5.7: *Red Rock Value Detected by Fittest HomCNE Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Med-Res/Hi-Res Detectors:* Rovers specialized to med-res and hi-res detection, respectively. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	Med-Res Detector	Hi-Res Detector
1	73.44 %	85.80 %
2	75.50 %	87.10 %
3	78.00 %	84.30 %
4	77.05 %	85.70 %
5	80.14 %	89.10 %
6	82.25 %	86.23 %
7	NA	81.45 %
8	NA	83.90 %
9	NA	84.07 %
10	NA	80.90 %

Tab. 5.8: *Red Rock Detected by Fittest HetCNE Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original performance. *Hi-Res Detector:* Specialized to hi-res detection. *RRD:* Red Rock Distribution.

RRD	Hi-Res Detector
1	80.05 %
2	82.13 %
3	89.55 %
4	80.10 %
5	85.12 %
6	82.33 %
7	85.55 %
8	89.50 %
9	88.75 %
10	80.44 %

Tab. 5.9: *Area Covered by Fittest HomCNE Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Med-Res/Hi-Res Detectors:* Rovers specialized to med-res and hi-res detection, respectively. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	Med-Res Detector	Hi-Res Detector
1	83.86 %	79.70 %
2	85.10 %	78.31 %
3	84.04 %	80.13 %
4	85.15 %	84.27 %
5	83.94 %	85.15 %
6	88.56 %	83.25 %
7	NA	87.05 %
8	NA	88.98 %
9	NA	88.23 %
10	NA	85.76 %

Tab. 5.10: *Area Covered by Fittest HetCNE Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original performance. *Hi-Res Detector:* Specialized to hi-res detection. *RRD:* Red Rock Distribution.

RRD	Hi-Res Detector
1	75.70 %
2	75.24 %
3	77.30 %
4	74.75 %
5	75.35 %
6	79.53 %
7	77.95 %
8	80.88 %
9	82.25 %
10	81.50 %

Tab. 5.11: *Red Rock Value Detected by Fittest CCGA Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	Low-Res Detector	Med-Res Detector	Hi-Res Detector	Non-specialized
1	78.24 %	68.38 %	55.20 %	97.78 %
2	89.75 %	59.81 %	57.72 %	95.22 %
3	90.05 %	74.93 %	59.23 %	75.97 %
4	96.98 %	67.97 %	58.46 %	67.74 %
5	NA	81.04 %	47.93 %	70.32 %
6	NA	86.22 %	49.15 %	62.51 %
7	NA	84.14 %	55.24 %	65.29 %
8	NA	80.87 %	57.48 %	59.73 %
9	NA	NA	48.12 %	55.65 %
10	NA	NA	43.34 %	60.22 %

Tab. 5.12: *Area Covered by Fittest CCGA Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	Low-Res Detector	Med-Res Detector	Hi-Res Detector	Non-specialized
1	85.14 %	79.82 %	66.42 %	98.88 %
2	96.17 %	70.21 %	68.25 %	97.92 %
3	95.96 %	83.30 %	71.22 %	86.74 %
4	98.19 %	78.79 %	69.60 %	75.13 %
5	NA	90.16 %	58.32 %	79.01 %
6	NA	95.07 %	62.35 %	74.75 %
7	NA	90.10 %	68.26 %	72.20 %
8	NA	88.72 %	70.04 %	76.32 %
9	NA	NA	62.20 %	69.85 %
10	NA	NA	55.94 %	75.20 %

Tab. 5.13: *Red Rock Value Detected by Fittest MESP Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	**Low-Res Detector**	**Med-Res Detector**	**Hi-Res Detector**	**Non-specialized**
1	85.04 %	69.98 %	65.32 %	74.20 %
2	78.95 %	72.26 %	62.07 %	78.89 %
3	79.58 %	68.35 %	70.18 %	75.13 %
4	NA	65.17 %	68.73 %	55.28 %
5	NA	69.43 %	55.30 %	47.21 %
6	NA	70.02 %	51.10 %	49.93 %
7	NA	77.25 %	65.33 %	46.78 %
8	NA	73.19 %	54.12 %	57.82 %
9	NA	74.13 %	60.98 %	51.58 %
10	NA	79.90 %	44.48 %	59.23 %

Tab. 5.14: *Area Covered by Fittest MESP Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution. *NA:* Not Applicable.

RRD	**Low-Res Detector**	**Med-Res Detector**	**Hi-Res Detector**	**Non-specialized**
1	90.12 %	78.80 %	76.37 %	85.10 %
2	86.51 %	81.20 %	73.12 %	85.21 %
3	85.82 %	80.55 %	82.28 %	89.87 %
4	NA	76.75 %	79.83 %	68.80 %
5	NA	80.49 %	68.35 %	59.96 %
6	NA	86.20 %	67.77 %	60.19 %
7	NA	85.58 %	74.03 %	57.27 %
8	NA	79.90 %	65.20 %	68.98 %
9	NA	85.92 %	70.25 %	58.90 %
10	NA	88.20 %	49.76 %	67.78 %

Tab. 5.15: *Red Rock Value Detected by Fittest CONE Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	Low-Res Detector	Med-Res Detector	Hi-Res Detector	Non-specialized
1	40.35 %	34.72 %	35.95 %	61.90 %
2	42.50 %	30.05 %	32.60 %	75.15 %
3	45.72 %	42.02 %	30.08 %	68.78 %
4	44.80 %	39.77 %	29.68 %	65.16 %
5	43.53 %	44.67 %	33.55 %	50.85 %
6	50.21 %	45.20 %	39.26 %	52.36 %
7	48.78 %	49.53 %	31.58 %	47.65 %
8	47.12 %	43.29 %	28.98 %	51.10 %
9	57.95 %	42.11 %	20.79 %	45.58 %
10	52.26 %	43.93 %	21.02 %	48.20 %

Tab. 5.16: *Area Covered by Fittest CONE Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original task performance. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	Low-Res Detector	Med-Res Detector	Hi-Res Detector	Non-specialized
1	54.55 %	48.72 %	55.98 %	79.80 %
2	52.35 %	43.13 %	49.67 %	88.98 %
3	55.72 %	47.02 %	33.15 %	71.00 %
4	53.28 %	51.16 %	39.62 %	69.05 %
5	57.02 %	50.97 %	40.03 %	61.25 %
6	52.29 %	49.82 %	35.12 %	67.73 %
7	58.03 %	44.95 %	39.98 %	65.13 %
8	50.92 %	48.90 %	38.22 %	64.91 %
9	59.86 %	49.18 %	39.03 %	65.28 %
10	52.26 %	43.93 %	40.19 %	68.30 %

The behavioral validation experiments elucidate that the fittest teams consist of rovers specialized to different and complementary behavioral specializations. The behavioral validation experiments evaluate the contribution of each rover behavior to a team's collective behavior performance.

1. *Behavioral Validation Experiment 1:* Executes a team where each rover uses the fittest controller specialized to the *low-res detection* behavior. This team is a *Low-Res Detector Team*.

2. *Behavioral Validation Experiment 2:* Executes a team where each rover uses the fittest controller specialized to the *med-res detection* behavior. This team is a *Med-Res Detector Team*.

3. *Behavioral Validation Experiment 3:* Executes a team where each rover uses the fittest controller specialized to the *hi-res detection* behavior. This team is a *Hi-Res Detector Team*.

4. *Behavioral Validation Experiment 4:* Executes a team using only the fittest *non-specialized* controller. This team is a *Non-Specialized Team*.

Figures 5.14 and 5.15 present the task performance results of low-res, med-res, hi-res detector, and non-specialized teams evaluated for these behavioral validation experiments. For each behavioral validation experiment, each rover in a team was set with an identical behavior. This was done as follows. A rover was initialized with the controller selected from the fittest team evolved by a given method. This selected controller is the fittest that exhibits either a specialized low-res, med-res or hi-res detection behavior, or no specialization. These controllers are always selected from the fittest teams evolved by CONE, since these teams yielded the highest task performance in all environments. For each experiment, each selected controller is cloned 20 times in order to create either a non-specialized, low-res, med-res, or hi-res detector team. Each team is then executed in each of the complex environments. The average red rock value detected and area covered by the team is calculated over 20 experimental runs for each of the environments. For each behavioral validation experiment, rovers are placed in random locations and executed for one rover *lifetime*.

The behavioral validation experiments do not employ any methods for adaptation of controllers. Thus, the task performance achieved by the specialized and non-specialized teams results from the interactions of individual rover behaviors. Teams consisting entirely of rovers specialized to the *move* behavior are not tested, given that the move action does not directly contribute to task performance. That is, only behaviors corresponding to the activation of red rock detection sensors directly contributes to rover team task performance.

Comparisons for Behavioral Validation

This section presents the results of statistical tests that compare the task performances of non-specialized and specialized teams (section 5.5.4) with that of evolved teams. For simplicity, the non-specialized and specialized teams are

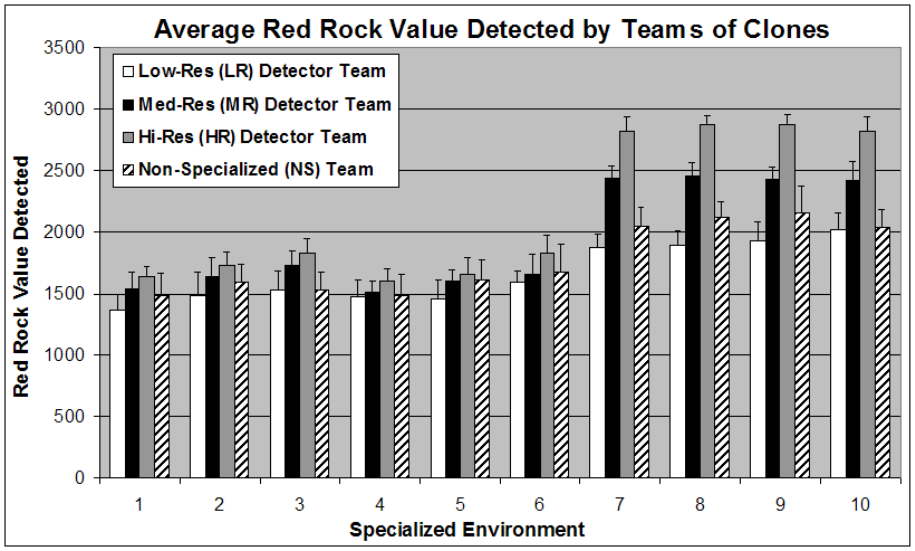

Fig. 5.14: Red Rock Value Detected by Teams of Clones in the Complex Environments. Teams consist only of *non-specialized* or *specialized* rovers.

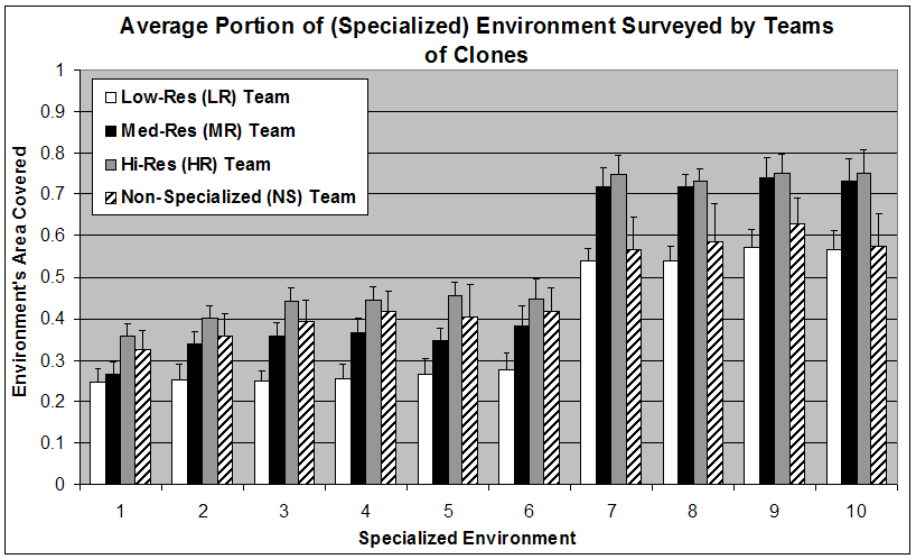

Fig. 5.15: Area Covered by Teams of Clones in the Complex Environments. Teams consist only of *non-specialized* or *specialized* rovers.

referred to as *cloned teams*. Task performance data sets of the cloned teams were found to conform to normal distributions via applying the Kolmogorov-Smirnov test. The results of statistical task performance comparisons between cloned and evolved teams is presented in appendix C.

Results of Behavioral Validation Experiments

Results of the behavioral validation experiments indicate that teams consisting of rovers with the same behavior are insufficient for achieving a task performance comparable to that achieved by evolved teams. That is, the average task performance of *cloned* teams is significantly lower comparative to the average task performance of CCGA, Multi-Agent ESP, and CONE evolved teams (appendix C). These results support the statement that the high task performance yielded by the fittest CCGA, Multi-Agent ESP, and CONE evolved teams is a consequence of the interaction between non-specialized and specialized castes (section 5.5.3). In support of this, there was no significant difference between the average red rock value detected by *non-specialized* and *med-res* detector teams, and teams evolved by HomCNE (containing a single caste), for complex environments [1, 6]. There was no significant difference between the average red rock value detected by *non-specialized* and *hi-res* detector teams, and teams evolved by HomCNE (containing a single caste), for environments [7, 10]. Similarly, there was no significant difference between the average red rock value detected by *hi-res* teams, and teams evolved by HetCNE (containing a single caste), for environments [1, 6]. Also, there was no significant difference between the average red rock value detected by *non-specialized* teams, and teams evolved by HetCNE, for environments [7, 10].

This result was expected since the fittest HomCNE and HetCNE evolved teams consisted of rovers specialized to the med-res (or hi-res for HetCNE evolved teams) detection for environments [1, 6], and rovers specialized to hi-res detection for environments [7, 10] (appendix C). The comparable performance between non-specialized and specialized cloned teams indicates that rovers specialized to one behavior are not able to achieve the task performance achieved by the fittest HomCNE and HetCNE evolved teams.

5.5.5 The Role of Difference Metrics in CONE

This section investigates the efficacy of the *Genotype Difference Metric* (GDM) and *Specialization Difference Metric* (SDM) for facilitating behavioral specialization, and increasing task performance in CONE evolved teams. The following three variants of CONE are executed in the same experimental setup.

1. *CONE without GDM (CONE-1):* Teams are evolved using CONE without the GDM meaning that genotype recombination only occurs *within* populations. The SDM remains active.

2. *CONE without SDM (CONE-2):* Teams are evolved using CONE without the SDM. The GDM remains active.

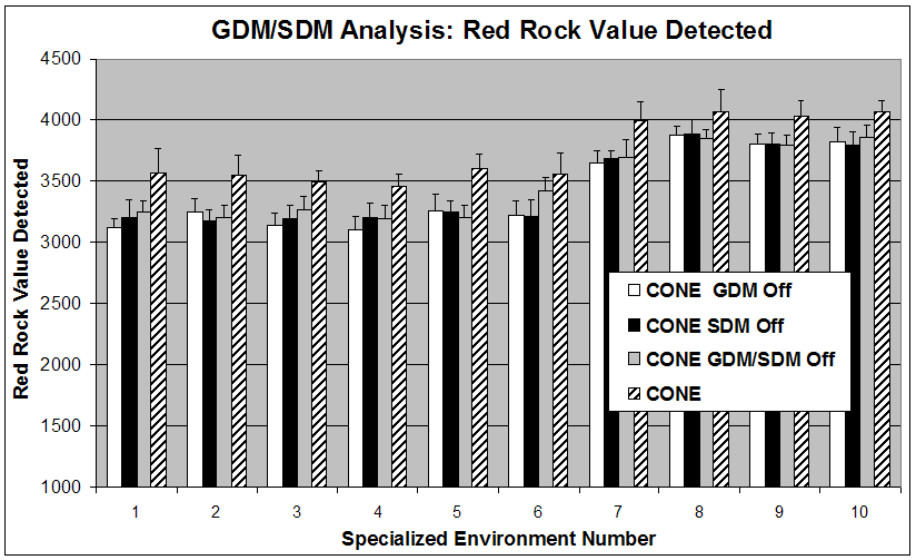

Fig. 5.16: Red Rock Value Detected by Teams Evolved by CONE Variants in Complex Environments. Results of teams evolved by CONE-1, CONE-2 or CONE-3. Note that the red rock detected axis range is: [1000, 4500].

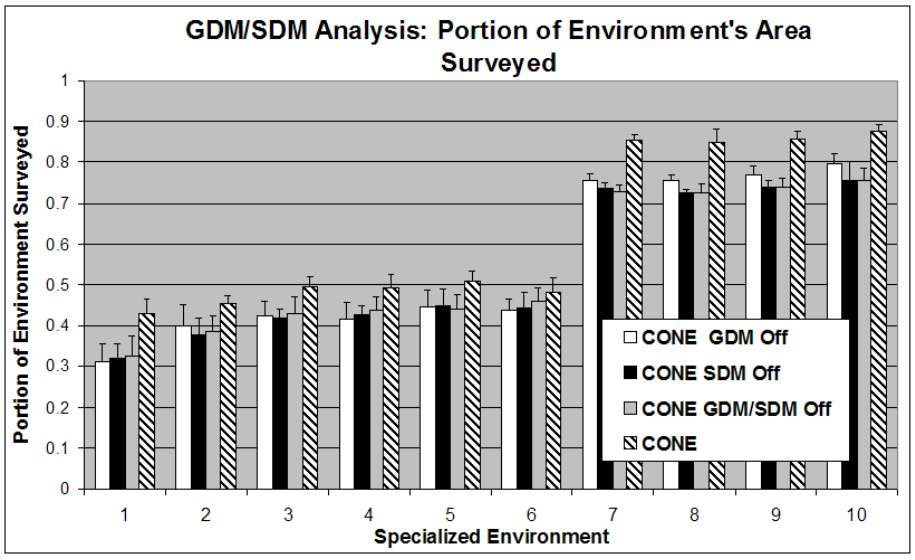

Fig. 5.17: Area Covered by Teams Evolved by CONE Variants in Complex Environments. Results of CONE-1, CONE-2, CONE-3 evolved teams.

3. *CONE without GDM and SDM (CONE-3):* Teams are evolved using CONE without both the GDM and SDM.

Figure 5.16 presents the *average red rock value* detected by teams evolved using the CONE variants (CONE-1, CONE-2, CONE-3), for all complex environments. Figure 5.17 presents the *average area covered* by teams evolved using CONE-1, CONE-2 and CONE-3, for all complex environments. These task performance results are averaged over 20 experimental runs. For comparison, results previously attained by CONE evolved teams are also presented in figures 5.16 and 5.17. Data sets of CONE-1, CONE-2, and CONE-3 evolved teams were found to conform to normal distributions via applying the Kolmogorov-Smirnov test. A statistical comparison between the task performances of teams evolved by the CONE variants and CONE is presented in appendix C. This comparison indicates that, for all complex environments, there is a significant difference between the task performances of teams evolved by CONE, and that of teams evolved by CONE-1, CONE-2, and CONE-3. Teams evolved by CONE yield a performance advantage over the CONE variants.

This result supports the hypothesis that both the GDM and SDM are 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 caste lesion study (section 5.5.3) supports the hypothesis that the GDM and SDM enables CONE to evolve behavioral specialization that achieves a higher task performance, comparative to related methods. In this case, the behavioral specialization takes the form of a set of complementary and interacting rover castes.

5.6 Conclusions

This chapter investigated the application of CONE for evolving collective behaviors in a team of simulated rovers. The multi-rover task stipulated that a team must maximize the value of features of interest (*red rocks*) cooperatively detected in an environment. CONE was successful for evolving collective behaviors that out-performed teams evolved by comparative controller design methods. CONE evolved teams consisting of complementary behavioral specializations. An analysis indicated these specializations were necessary for a high task performance to be achieved. The next collective behavior case study is the *Gathering and Collective Construction* (GACC) task, which is an extension of the multi-rover task. The GACC task uses a greater number of controllers and requires more complex forms of collective behavior.

6. COLLECTIVE BEHAVIOR CASE STUDY: GATHERING AND COLLECTIVE CONSTRUCTION

Gathering and Collective Construction (GACC) is a simulation of a multi-robot system operating in a continuous environment that attempts to solve a collective behavior task. The nature of the GACC task draws inspiration from biological collective behavior systems [17]. The GACC task is designed with potential applications that include the cooperative construction of complex structures in hazardous or uninhabitable environments, such as underwater human habitats or orbiting space stations [183]. The GACC task is an extension of the *multi-rover task* (chapter 5) that includes two additional components.

1. *Atomic Object Detection and Gathering:* The first component is the detection, and transportation of (gathering) *atomic objects*[1] to a *home area* within the confines of a simulation environment.

2. *Complex Object Construction:* The second component is the use of gathered atomic objects as the building blocks for the construction of a complex object at a construction zone within the home area[2].

Specifically, in the GACC task, a team of simulated robots search for, and transport atomic objects towards a construction zone within a home area. In order to be accomplished, the GACC task has the following requirements.

1. Gathered atomic objects are transported to a home area. The home area contains a smaller area within called the construction zone. Complex objects are constructed at the construction zone.

2. At least two robots are required in order to cooperatively transport objects from where they are detected in the environment, to the construction zone.

3. Atomic objects must be cooperatively transported to the construction zone in a particular sequence. A particular sequence is required in order that a complex object be constructed at the construction zone.

The performance measure of a team attempting to solve the GACC task, is the number of atomic objects delivered in the correct sequence to the construction zone. This in turn determines the total number of complex objects that are constructed over the course of the team's lifetime.

[1] Throughout this chapter the terms *atomic objects* and *objects* are used interchangeably. Both refer to objects that are used as the building blocks of complex objects.

[2] The terms *construction zone* and *home area* are used interchangeably.

This chapter is structured as follows. Section 6.1 describes the GACC task. Section 6.2 describes the sensors, actuators and controllers used by robots. Section 6.3 describes the experimental setup of the GACC task. Section 6.4 describes experimental results yielded from applying various NE methods in order to evolve collective behavior. Section 6.5 presents an analysis and discussion of results. Section 6.6 presents the chapter's conclusions.

6.1 Gathering and Collective Construction (GACC) Task

6.1.1 Specialization in the GACC Task

Behavioral specialization as part of controller behavior, and overall team behavior is mandated in order to optimally accomplish the GACC task. An optimal task performance is defined as the delivery of all atomic objects in the environment, in the correct sequence, to the construction zone, and the subsequent construction of the maximum number of complex objects from these atomic objects. To illustrate the benefit of specialization, consider the following description of the GACC task. In an environment, there is a set of Q *atomic objects* of n different types, which robots have to *detect*, and *cooperatively transport* to a construction zone. n is the total number of object types. *Complex objects* need to be assembled from a given set of atomic objects at the construction zone. The construction zone is at the center of the home area. A complex object is defined as a combination of m different atomic object types, where $m \leq n$. Each atomic object must be delivered to the construction zone in a predefined order [a, b,...,z], where a, b and z is the number of each object type in a sequence of object types that are to be delivered to the construction zone.

At least two robots are required to transport an object to the construction zone. Delivery of an object to the construction zone, where the object is the next required object type in predefined sequence, represents one step in the construction process of a complex object. All robots in the simulation are morphologically identical. However, robot controllers are behaviorally heterogenous, and each controller retains the capability to specialize to detecting or gathering different object types. Initially each robot in the team adopts a behavior such that it searches for and gathers objects of any type. Although, this is not an optimal complement of behavioral roles if there are differing numbers of each object type. For example, consider an environment where there are two object types A and B, and where object type A is particularly scarce, and object type B is plentiful. Given that the object type sequence required in order to construct a complex object first requires the delivery of a set of type A objects, and then a set of type B objects to the construction zone, then an appropriate team behavioral composition is for most robots to search for type A objects, and a few robots to search for type B objects.

Defining Behavioral Specialization in the GACC Task

Specialization is measured with respect to individual controller behavior. Specialization is defined by applying the behavioral specialization metric (section 3.2.1) to a given controller's behavior. This specialization metric calculates a degree of specialization (S) for a given behavior. If $S < 0.5$ for a given behavior, the robot is labeled as *specialized*. If $S \geq 0.5$ then the robot's behavior is labeled as *non-specialized*. Given that a behavior is defined as specialized, then the specific label given to a specialization corresponds to the action that is most executed over the course of the the robot's lifetime. In terms of the GACC task, this implies that a specialized robot is generally categorized as either a *detector*, *gatherer*, or *constructor*. A specialization is defined in the case that a robot executes a given detection, gathering or construction behavior, such that, over the course of the robot's lifetime, this behavior is executed for a time that exceeds the time dedicated to the execution of any other action. These specializations are described in the following, where $x \in$ A, B, C (the set of object types).

- *Object-x Detector:* A robot that specializes to activating it's object detection sensors with *setting x*.
- *Object-x Gatherer:* A robot that specializes to activating it's gripper with *setting x*, and then moves with a gripped type x object,
- *Object-x Constructor:* An object x gatherer that specializes to correctly delivering type x objects to the construction zone.

6.1.2 GACC Simulation Environments

A GACC environment is defined by the following features.

1. *Home Area / Construction Zone:* The home area is at the environment's center. The construction zone is a smaller area within the home area, and is where complex objects are constructed. The home area is where gathered objects that are not ready to be used in construction, are dropped. Figure 6.1 illustrates the home area and the construction zone.
2. *A Set of Complex Objects (P):* P is to be constructed from q objects, where there are n objects in total. A complex object in P is comprised of a subset of all objects (Q). There are p subsets in Q. Each of the p subsets corresponds to a complex object. Each complex object consists of a predefined order of m different objects, where $m \leq n$.
3. *Complex Object Construction:* Requires that at least two robots cooperatively deliver an object to the construction zone, where this object is the next required in a predefined sequence.
4. *Atomic Objects:* A set of q objects, of n different types are randomly distributed throughout the environment, but not in the home area.

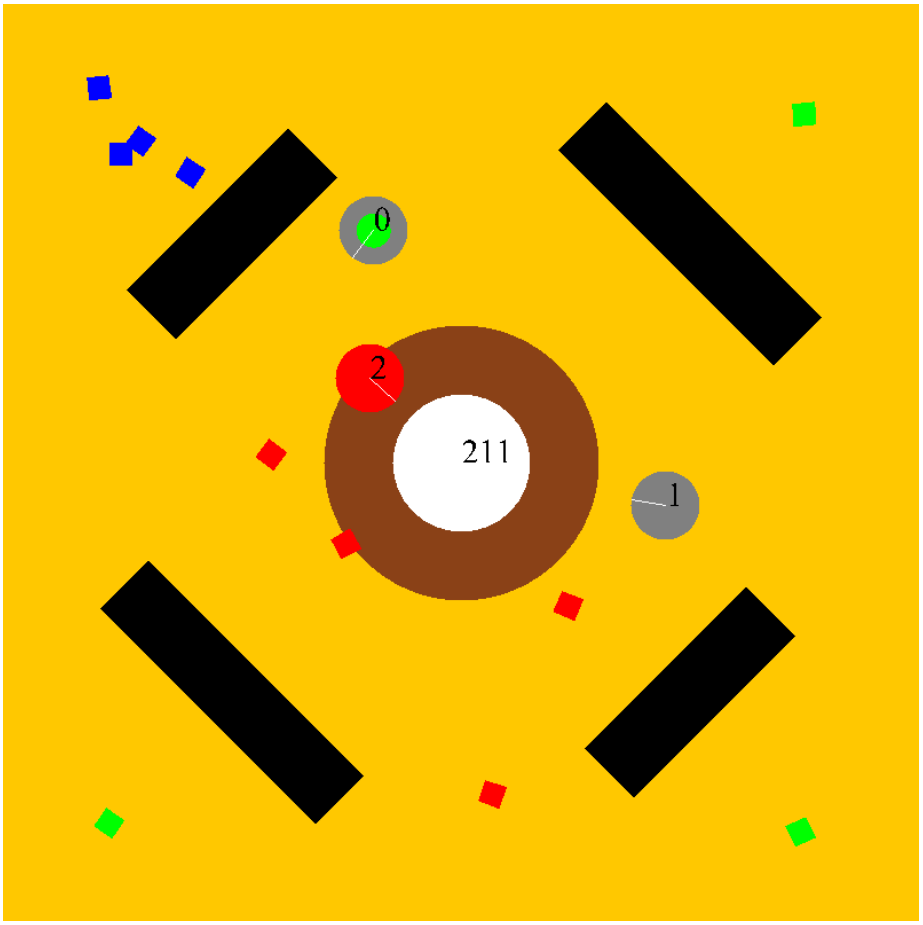

Fig. 6.1: Example GACC Simulation Environment. A screen shot of the simulation environment. The brown area is the home area. The white area at the center of the home area is the construction zone. The construction zone is the foundation for the construction of one or more complex objects. A set of atomic objects must be delivered in a given sequence of object types in order for a complex object to be constructed. In this example, robot 2 (red) is gripping a type A object and awaiting assistance from robot 1 (grey) in order to deliver the type A object to the construction zone. Robot 0 (green) is searching for type B objects whilst avoiding obstacles. The label in the construction zone *211* indicates that the complex object being constructed requires a type C object (labeled by identifier: 2), and then two type B objects (labeled by identifier: 1) in order to be completed.

Tab. 6.1: Detection and Transportation of Objects. Required sensor and gripper settings depend upon the object type.

Object Type	**Required Detection Sensor Setting**	**Required Gripper Setting**	**Robots to Transport / Construct**
A	Detection Setting A	Gripper Setting A	2
B	Detection Setting B	Gripper Setting B	3
C	Detection Setting C	Gripper Setting C	4

5. *Obstacles:* Obstacles are placed randomly in the environment, but not in the home area. These obstacle represent *blind spots* that block the fields of view of a robot's object detection sensors.

Robots in the GACC Environment

Robots operate in a continuous bounded two dimensional environment. Only one robot can occupy any given x, y position in the environment. Movement is calculated in terms of real valued vectors. In order to calculate the distances between robots and objects or obstacles, the squared Euclidean norm, bounded by a minimum observation distance δ^2_{min}, is used (equation 6.1).

$$\delta(p, q) = min(\|x - y\|^2, \delta^2_{min}) \tag{6.1}$$

Atomic Objects and Distribution of Atomic Objects

There is a set of q atomic objects placed in random locations throughout the environment, except in the home area. These objects are referred to as atomic objects since they are used in the construction of one or more complex objects. For this set of q atomic objects, there is a distribution n different object types. There are three different atomic object types: [A, B, C]. These object types are only detectable by detection sensors with settings A, B, and C, respectively. Type A, B, C objects can only be transported by robots using gripper settings A, B, and C, respectively (table 6.1). The object type distribution (the number of each object type) depends upon the experiment (section 6.3). In a given environment, there are enough atomic objects to construct p complex objects (section 6.1.2). The number of objects of each type required to construct each of the p complex objects depends upon the environment (section 6.3).

Obstacles

There are z rectangular obstacles randomly placed throughout the environment. These obstacles block the movement of robots, as well as the *Field Of View* (FOV) of sensors. At maximum sensor range, an object spans the length of circumference of a given detection sensor (figure 6.2).

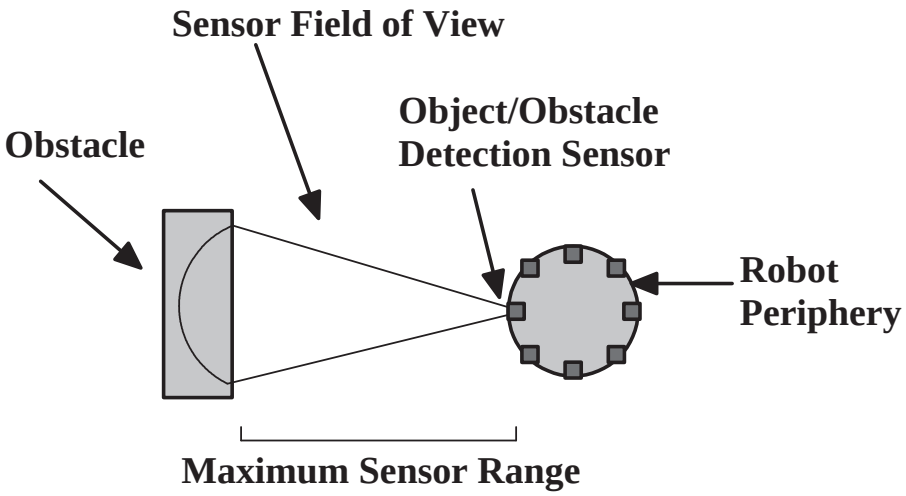

Fig. 6.2: Robot Sensors and Obstacles. An example of an obstacle blocking the FOV of a robot's detection sensor at maximum sensor range.

6.1.3 Complex Object Construction

The goal of the team is to maximize the number of objects delivered in the correct order to the construction zone, and hence maximize the number of complex objects constructed over the team's lifetime. Complex objects are constructed from a set of gathered two-dimensional square atomic objects. The number of complex objects that a team is to construct, the number of each atomic object type, and the correct sequence of object types that comprise each complex object, depend upon the environment that is being tested (section 6.3).

Complex Objects for Guiding Collective Construction

Inspired by related work in multi-robot systems [181], the atomic objects (building blocks) are able to process environmental information, and communicate with the robots in the team over the course of the collective construction process. When at least two objects (a complex object consists of at least three blocks) have been delivered to the construction zone, then this is a semi-formed complex object. This semi-formed object then broadcast signals that direct the collective construction process according to the following procedure.

- The first stage of complex object construction is complete when at least two objects have been delivered to the construction zone.
- After the first stage of construction, the semi-formed complex object broadcasts an *object demand signal* indicating the next object type *required* for the next stage of construction. This signal is interpreted by each robot as a degree of demand for each object type.

- The demand for each object type is a function of the position of the object type in the construction sequence (when it will be required) and the scarcity of the object type in the environment. For example, if the next required object type is the scarcest then it will be perceived by each robot's object demand sensors as the type with the highest demand.
- When a complex object is complete then it will stop broadcasting *object demand signals*. The object demand sensors of each robot will then receive a *zero demand* for each object type.
- If there are no more complex objects to construct, the final complex object will broadcasts a *stop* signal indicating task completion. Otherwise, the team will continue searching for objects.

To illustrate this process, figure 6.3 presents an example of a semi-formed complex object within the construction zone at simulation time t. Also illustrated are the sequence of object types, the number of each type required in order to complete complex object construction, and the object demand signal broadcast from the complex object at time t. Multiple complex objects can be constructed simultaneously, however this renders the task more complex, since each robot will receive signals conveying different degrees of demand for different types to be used in the construction of separate complex objects.

Task Requirements for Collective Behavior

Atomic objects are detectable by individual robots, however, detected objects must be cooperatively transported and used in the construction process by at least two robots. Cooperative delivery of an object to the construction zone constitutes one step in the collective construction process. Table 6.1 presents the detection sensor setting required in order for robots to detect objects of each type, and the number of robots required for cooperative transportation, and subsequently deliver of objects to the construction zone. If a transported object is not the next required type in the sequence required for complex object construction, then the robots transporting the object will adopt a heuristic behavior (section 6.2.8). This directs them to wait for a given time period, before dropping the object in the home area. The robots will then continue to search for other objects. If a transported object is the next required type, then the transporting robots must be using the same gripper setting in order to deliver the object to the construction zone (table 6.1).

6.2 Robots

6.2.1 Object/Obstacle Detection Sensors

Each robot is equipped with eight object/obstablce detection sensors ([S-0, S-7] in figure 6.4), where each sensor covers one quadrant in the sensory FOV. These detection sensors fulfill two functions.

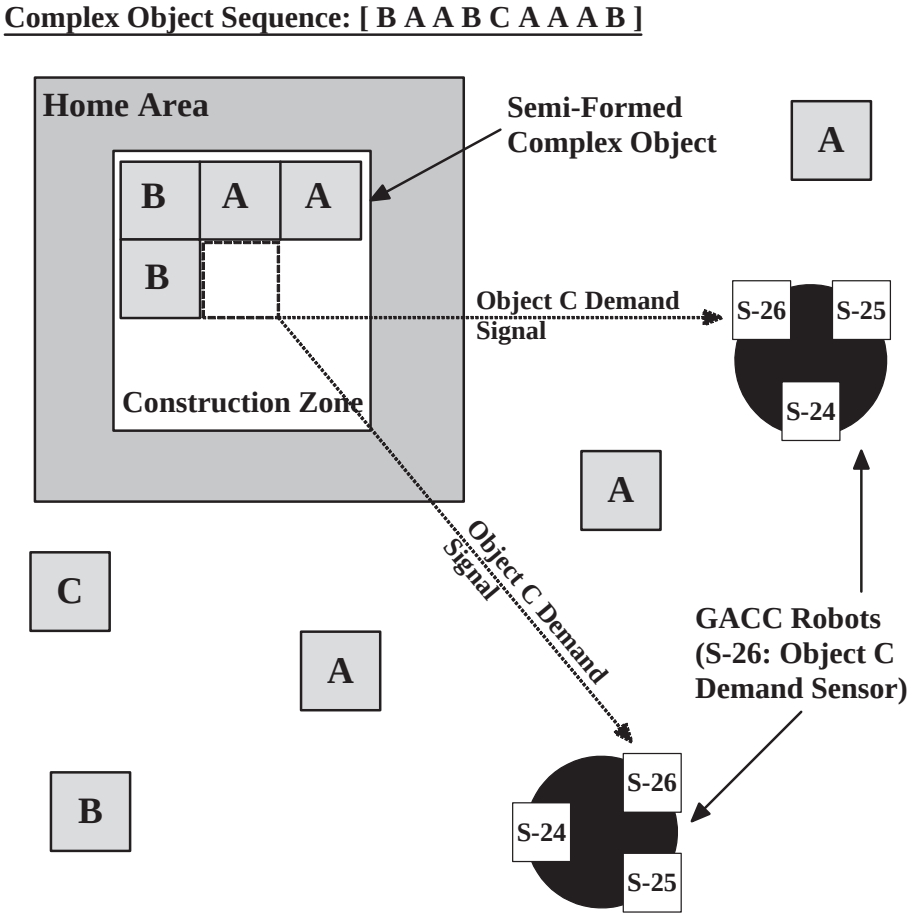

Fig. 6.3: Semi-Formed Complex Object. An example of a part of the environment and a semi-formed complex object being constructed from a set of type A, B, and C objects. A type C object is the next required in the sequence and the least plentiful. Hence, the semi-formed complex object is broadcasting a demand for a type C object. This object demand signal is received by each robot, which in turn influences their search behavior.

Tab. 6.2: Object/Obstacle Detection Sensors. At each simulation iteration a robot can activate it's detection sensors with one of three settings.

Detector Sensor Setting	**Object Type Detected**	**Obstacles Detected**	**Accuracy**	**Range**	**Cost**
Setting A	A	Yes	1.0	0.010	0.25
Setting B	B	Yes	1.0	0.025	0.25
Setting C	C	Yes	1.0	0.050	0.25

1. *Object Detection:* An *Object* is a building block used in the construction of a complex object. Objects are classified as either *type A*, *B*, or *C*.

2. *Obstacle Detection:* Obstacles are represented as either static blocks or as the boundary of the environment. Obstacles are barriers for detection sensors (figure 6.2). Detection sensors have no FOV beyond the edges of the environment, and beyond obstacles.

Detection sensors need to be explicitly activated, where all eight sensors are activated with one of three settings. These settings are: *A*, *B*, and *C*. In table 6.2, *accuracy* is the degree of probability with which objects/obstacles within the eight sensor quadrants (a 360 degree FOV) are detected. *Range* is a portion of the environment's width. Each environment's length and width are equal in these experiments. *Cost* is the energy consumed each time the detection sensors are activated. Sensor activation uses one simulation iteration.

Detection sensor q returns the closest object type, or obstacle, in quadrant q, divided by the squared distance to the robot (equation 6.2).

$$S_{1(q,t)} = j \in J_q \frac{1}{\delta(L_{v,t}, L_{j,t})} \tag{6.2}$$

Where q is a sensor quadrant,

v is a robot,

t is simulation time step t,

J_q is the set of all objects and obstacles in quadrant q,

L_j $(j \epsilon J_q)$ is the location of object/obstacle j,

L_v is the location of robot v,

j is a type [A, B, C] object or obstacle, where $j \epsilon J_q$.

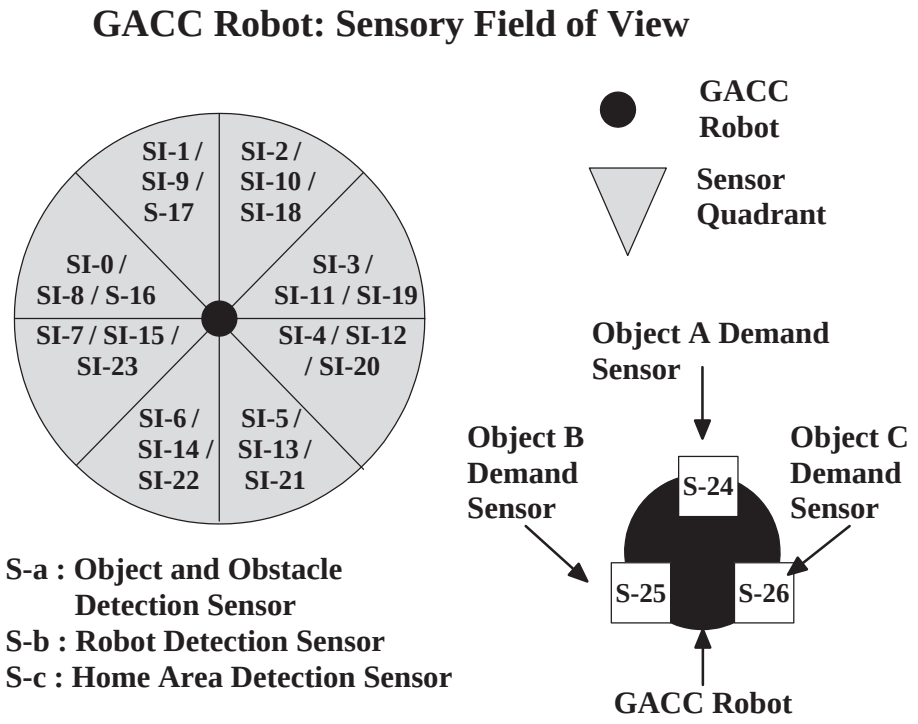

Fig. 6.4: Robot Sensory Field of View (FOV). Sensor space consists of eight quadrants. There is one object/obstacle detection sensor and one robot detection sensor per sensor quadrant ([S-0, S-7] and [S-8, S-15], respectively). Also, there are three object demand sensors ([S-24, S-26]) positioned on the robot's periphery. These sensors have no FOV. Rather, each object demand sensor receives a signal broadcast from complex objects being constructed.

6.2.2 Detection of Other Robots

Each robot is equipped with eight sensors ([S-8, S-15] in figure 6.4), for detecting other robots. These sensors fulfill two functions.

1. First, to prevent collisions between robots.
2. Second, to provide each robot with an indication of the current *state* of other robots within its sensory FOV. *State* refers whether a detected robot is carrying an object and the object *type* being carried.

The eight robot detection sensors are constantly active, and have a fixed accuracy, range and cost (table 6.4). Detection sensor q returns a value corresponding to the object type being carried by the closest robot, divided by the squared distance to *this* robot (equation 6.3).

$$S_{2(q,v,t)} = \frac{dv'}{\delta(L_{v'}, L_{v,t})} \tag{6.3}$$

Where q is a sensor quadrant,

v is *this* robot, that is, the robot that is detecting other robots,

t is simulation time step t,

v' is the closest robot to *this* robot in quadrant q,

dv' is what is being carried by robot *v'*. Note that: *dv'* yields a value equal to either: [0, 1, 2, 3], which corresponds to robot *v'* carrying either: *no object*, a *type A*, a *type B*, or a *type C* object, respectively,

$L_{v'}$ is the location of the closest robot in sensor quadrant q,

L_v is the location of *this* robot.

6.2.3 Object Demand Sensors

Each robot is equipped with three sensors ([S-24, S-26] in figure 6.4) that indicate the current demand for each object type. During the construction process, each complex object broadcasts a signal that is received by each robot's object demand sensors. Equations 6.4, 6.5, and 6.6 present the functions used to calculate a demand value for object types A, B, and C respectively.

$$S_{A(q,v,t)} = D_{j',dA_{j'},t} \in N_A \tag{6.4}$$

$$S_{B(q,v,t)} = D_{j',dB_{j'},t} \in N_B \tag{6.5}$$

$$S_{C(q,v,t)} = D_{j',dC_{j'},t} \in N_C \tag{6.6}$$

Where j' is the complex object that is to be constructed, $j' \in J'$,

J' is the total number of complex objects,

$dA_{j'}$ is the current priority of object type A,

$dB_{j'}$ is the current priority of object type B,

$dC_{j'}$ is the current priority of object type C,

N_A is the set of type A objects that have not yet been delivered,

N_B is the set of type B objects that have not yet been delivered,

N_C is the set of type C objects that have not yet been delivered,

v is *this* robot,

t is simulation time step t.

$D_{j',dA_{j'},t}$, $D_{j',dB_{j'},t}$, and $D_{j',dC_{j'},t}$ are functions that return received object demand sensor values for object types A, B, and C, respectively. Robot v is directed to gather the object type with the highest demand. That is, the object demand sensor yielding the highest value.

If $J' \geq 2$, then there are at least two complex objects concurrently being constructed. If different object types have the highest priority as the next required object type for at least two of the complex objects, then each of these object types is assigned the same priority.

For example, consider that: $J' = 2$. There are a total of four objects required to construct both complex objects 1 and 2. Both complex objects 1 and 2 currently consist of two building blocks. Object type A is the next type required in the construction of complex object 1, and object type B is the final type required for complex object 1. Object type B is the next type required to construct complex object 2. Object type C is the final type required to construct complex object 2. It is assumed that $t = x$, where $x \leq T$, and T is the total number of simulation iterations.

- $D_{j'=1,dA_{j'=1},t=x}$ returns a value of 3, given that two objects have already been used in construction. The next required object type for complex object 1 construction is a type A.
- $D_{j'=1,dB_{j'=1},t=x}$ returns a value of 4, since two objects have already been used in construction, and a type B object is the final building block for complex object 1 construction.
- $D_{j'=1,dC_{j'=1},t=x}$ returns a value of 0, since a type C object is not required for complex object 1 construction.
- $D_{j'=2,dA_{j'=2},t=x}$ returns a value of 0, since a type A object is not required for complex object 2 construction.

- $D_{j'=2,dB_{j'=2},t=x}$ returns a value of 3, since two objects have already been used to construct complex object 2, and a type B object is the next object for complex object 2 construction.
- $D_{j'=1,dC_{j'=2},t=x}$ returns a value of 4, since two objects have already been used in construction of complex objects 1 and 2, and a type C object is the final object for complex object 2 construction.

6.2.4 Home Area Detection

Each robot uses eight home area detection sensors ([S-16, S-23] in figure 6.4). Each sensor covers one quadrant in its 360 degree sensory FOV. Home area sensors are constantly active, and have a fixed accuracy, range and cost (table 6.4). Detection sensor q returns a value inversely proportional to the distance to the home area, divided by the squared distance to *this* robot (equation 6.7).

$$S_{3(q,v,t)} = h' \frac{1}{\delta(L_{h'}, L_{v,t})} \tag{6.7}$$

Where, h' denotes if the home area is within range of sensor q of v (*this* robot). h' yields a value equal to either: [0, 1], which corresponds to v not detecting, or detecting the home area, respectively. L_h is the location of the home area, and $L_{i,t}$ is the location of v at time t.

6.2.5 Movement Actuators

Each robot uses two wheel motors that control its heading at a constant speed. The distance a robot can move per simulation iteration, and the cost of movement is presented in table 6.4. A robot's heading is determined by normalizing and scaling the vectors dx and dy generated by motor outputs MO-4 and MO-5 (figure 6.6). That is, $dx = d_{max}(o_1 - 0.5)$, and $dy = d_{max}(o_2 - 0.5)$, where, d_{max} is the maximum distance a robot can traverse in one simulation iteration, and o_1 and o_2 are values of motor outputs MO-4 and MO-5, respectively.

6.2.6 Object Gripper

Each robot is equipped with a gripper turret (figure 6.5), for gripping and transporting detected objects. The gripper has three actuator settings: A, B, and C (table 6.1), allowing a robot to grip and transport type A, B, and C objects, respectively. The gripper needs to be explicitly activated. This uses one simulation iteration. The minimum distance between a robot and an object that is to be to gripped, and the cost of gripping is presented in table 6.1. When a robot is within the home area, and is gripping an object, predefined heuristic behaviors are activated (section 6.2.8), taking preference over ANN behaviors.

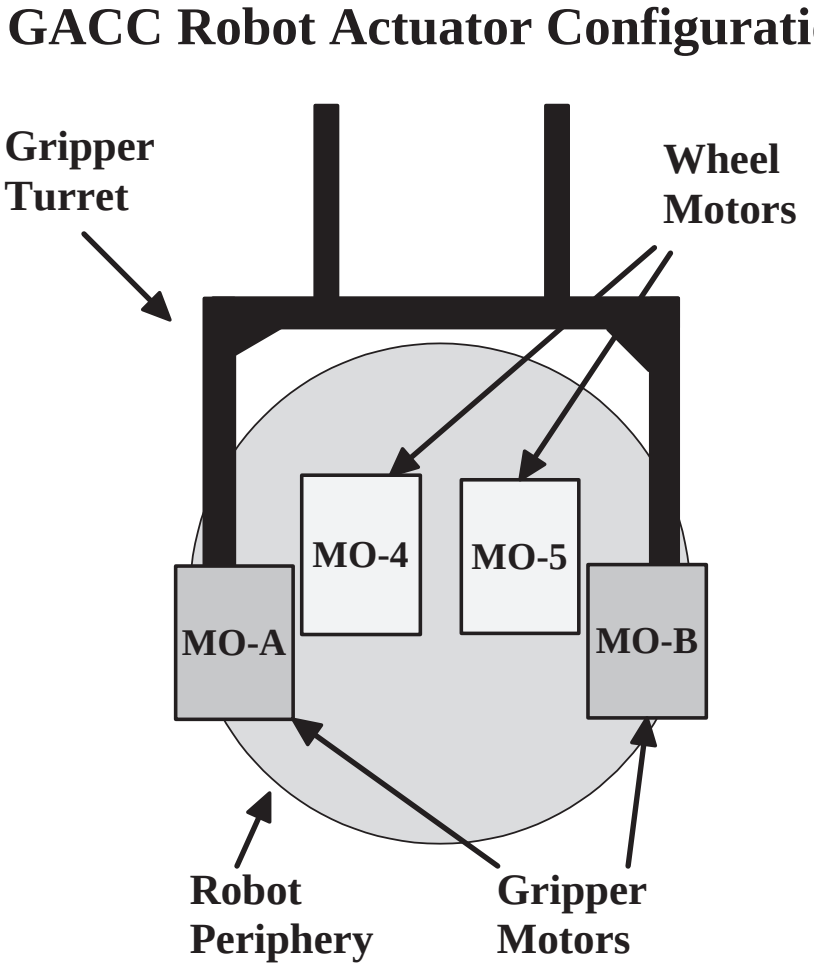

Fig. 6.5: Robot Gripper. Has three settings for gripping three object types. Setting 1: Minimum grip for gripping type A. Setting 2: Medium grip for gripping type B. Setting 3: Maximum grip for gripping type C. Settings are produced by controller motor outputs MO-5, MO-6, and MO-7 (section 6.2.7). One of these outputs is then fed to gripper motors MO-A and MO-B.

6.2.7 Artificial Neural Network Controller

Each robot uses a recurrent ANN controller [44], where 36 sensory input neurons are fully connected to 10 hidden layer neurons (figure 6.6). Sensory input neurons [SI-0, SI-7] accept input from each of the eight object/obstacle detection sensors. Inputs [SI-8, SI-15] accept input from each of the eight robot detection sensors. Inputs [SI-16, SI-22] accept input from each of the home area sensors. Inputs [SI-23, SI-25] accept input from each of the object demand sensors. Inputs [SI-26, SI-33] accept input from the previous activation state of each of the hidden layer neurons. The eight motor output neurons [MO-0, MO-7] are fully connected to the hidden layer neurons. The neurons comprising the hidden and output layers are sigmoidal units [70]. Also, within each of the hidden and output neurons, a scalar bias is included. A value of one is added to each neuron's weight product [93]. The inclusion of the bias in the calculation of each hidden and motor output neuron's weight product was found to be advantageous in the GACC collective behavior case study.

Action Selection

At each iteration, one of seven actions is executed by a robot, where the motor output with the highest value is the action executed.

1. *MO-0:* Activate all object/obstacle detection sensors with setting A.
2. *MO-1:* Activate all object/obstacle detection sensors with setting B.
3. *MO-2:* Activate all object/obstacle detection sensors with setting C.
4. *MO-3, MO-4:* If either MO-3 or MO-4 contains the highest value, then the robot moves in a direction calculated from dx and dy.
5. *MO-5:* Activate gripper with setting A.
6. *MO-6:* Activate gripper with setting B.
7. *MO-7:* Activate gripper with setting C.

6.2.8 Heuristic Behavior

In certain situations a robot will execute a predefined heuristic behavior. These situations and the corresponding behaviors are enumerated in the following description. In all other situations a robot's ANN controller produces its behavior.

Situation 1. If *this* robot is in the home area and is *not* gripping an object, and another robot *gripping* an object is detected in the home area then execute the following.

- Move towards the detected robot and grip the object that is being gripped.

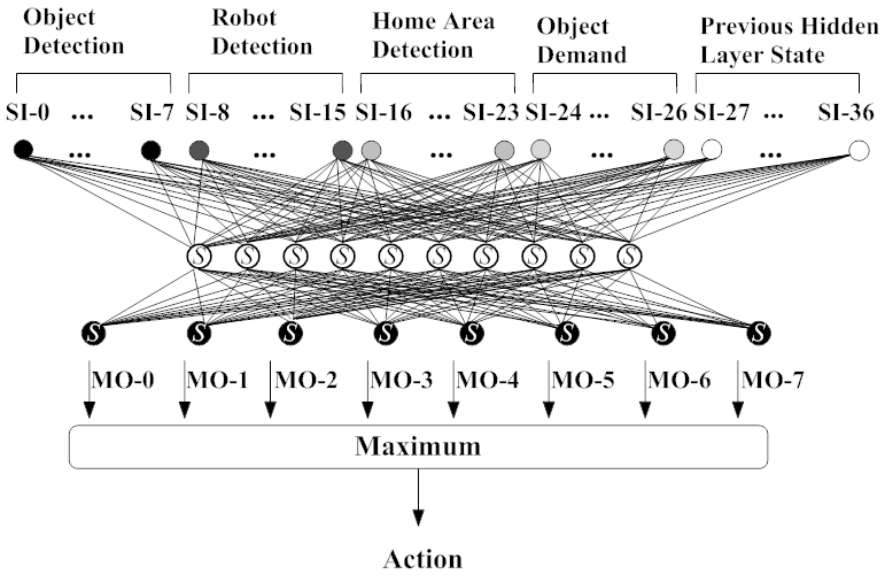

Fig. 6.6: Robot Controller. Recurrent ANN controller. Note that for the purposes of clarity, not all sensory input neurons are illustrated.

- If multiple robots are detected within the home area, each gripping an object, then move towards the closest robot with an object.
- If two or more robots are gripping the same object and not moving the object, and these are the closest robots, then grip this object also.
- If another robot first reaches the closest stationary robot (gripping an object), then move towards the next closest stationary robot.
- If there are no stationary robots (gripping objects) within the home area, then leave the home area and continue searching for objects.

Situation 2. If a robot is in the home area, and is gripping an object, then execute the following.

- Wait for a given time period (*wait with cargo* in section 6.4) for another robot's help (situation 1).
- If the *wait with cargo* time period is exceeded then drop the object, and continue searching for other objects.

Situation 3. If *this* robot is *not* gripping an object, and an object is detected in the home area, then execute the following.

- Move to, and grip the object.

Tab. 6.3: Heuristic Controllers: Probabilistic preferences control action selection. Each controller has either a *specialized* (OA: Object-A, OB: Object-B, OC: Object-C detector, gatherer, or constructor) or *non-specialized* behavior.

Controller Type	**O-A Detect**	**O-B Detect**	**O-C Detect**	**O-A Grip**	**O-B Grip**	**O-C Grip**	**Move**
O-A Detector	0.7	0	0	0	0	0	0.3
O-B Detector	0	0.7	0	0	0	0	0.3
O-C Detector	0	0	0.7	0	0	0	0.3
O-A Gatherer	0	0	0	0.7	0	0	0.3
O-B Gatherer	0	0	0	0	0.7	0	0.3
O-C Gatherer	0	0	0	0	0	0.7	0.3
O-A Constructor	0	0	0	0.7	0	0	0.3
O-B Constructor	0	0	0	0	0.7	0	0.3
O-C Constructor	0	0	0	0	0	0.7	0.3
Non-Specialized	100/7	100/7	100/7	100/7	100/7	100/7	100/7

Situation 4. If at least two robots are gripping an object that cannot be delivered because it is *out-of-order*, then execute the following.

- The robots drop the object, and continue searching for other objects.

Probabilistic Heuristic Controller

For the purposes of controller comparisons and an experimental analysis, robots also use non-adaptive heuristic controllers. Heuristic controllers are used for the experiments described in section 6.3. Heuristic controlled teams are not modified by any adaptive process, but rather the local interactions between robots produce a collective behavior. Seven different heuristic controller types are tested, where each type implements a preset *specialized* or *non-specialized* behavior (table 6.3). Each controller is defined by probabilistic preferences for selection of one of seven actions at each simulation iteration. These actions are the same as those used by the ANN controller. The probabilistic preferences for action selection were derived from a set of exploratory experiments. The preference values for object A, B, and C detectors and *gatherers* were selected since they produced a specialized behavior. That is, these heuristic controllers are specialized since they switch between executing different actions with a low frequency. Similarly, the action selection preference values selected for *non-specialized* heuristic controllers cause these controllers to switch between executing different actions with a high frequency (section 3.2.1). The preference values assigned to the object A, B, and C constructors are the same as those for the gatherers. However, constructors include an additional heuristic which causes constructors to remain within the home area.

6.3 Experimental Design

Experiments test 30 robots with n objects, z obstacles, and a home area in a bounded environment. Experiments measure the impact of a *collective behavior design method* and *environment* upon the *number of complex objects constructed* by the team. The experimental objective is to ascertain which collective behavior design method maximizes team task performance, and to investigate the contribution of emergent behavioral specialization to task performance.

- *Collective Behavior Design Methods*: Each robot controller is adapted with either one of the NE methods.
- *Simulation Environment*: Given a collective behavior design method, *simple* and *complex* environment sets are tested.

6.3.1 Team Fitness Evaluation

A global fitness function (G) is a function of the total *number of objects delivered in the correct sequence* to the construction zone and the number of complex objects constructed by the team. The goal of a team is to maximize G. However, robots do not maximize G directly, instead each robot (η) attempts to maximize its own private fitness function g_η. Also, G does not guide evolution, but rather provides a measure of team performance, based upon the contributions of individual robots. It is g_η that guides the evolution of each robot's controller.

Private Fitness Function

Equation 6.8 presents g_η which calculates the *number of objects delivered in the correct sequence* by η over the course of its lifetime. When an object is delivered to the construction zone each robot receives the same fitness reward.

$$g_v = \sum_{0 \leq t \leq T} \sum_{j \in J} ov_{j,t} \tag{6.8}$$

Where, v is a robot,

$ov_{j,t}$ is the number of objects j delivered in the correct sequence at time t,

J is the set of all objects within the environment.

Global Fitness Function

G calculates the *sum of objects delivered in the correct sequence* by the team. For any given experiment, an average *number of objects delivered in the correct sequence* is calculated over all epochs of all robot lifetime's. The highest *number of atomic objects delivered in the correct sequence* is then selected from each of the robot lifetime's in order to calculate G (equation 6.9).

$$G = \sum_{v \in V} g_v \tag{6.9}$$

Where V is the set of all robots.

6.3.2 Simulation and Neuro-Evolution Parameters

Tables 6.4 and 6.5 present the simulation and NE parameter settings, respectively. NE parameter settings are those used by the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE methods. Simulation settings are those used by the simulator. Any given experiment consists of 250 generations. Each generation corresponds to the *lifetime* of each robot in the team. Each robot lifetime lasts for 10 epochs, where each epoch consists of 3000 simulation iterations. Each epoch represents a task scenario that tests different robot starting positions, and object locations in the environment. A team's task performance is calculated over 20 experimental runs. Parameter values presented in tables 6.4 and 6.5 were derived in a set of exploratory experiments, which indicated that minor changes to these values produced similar results. Changing parameters values to within 0.20 of the values given in tables 6.4 and 6.5 resulted in the evolution of teams that yielded a team task performance within 0.09 of the task performance results presented in section 6.3.

6.3.3 Evolution of Collective Behavior

HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE are applied in order to evolve team behavior.

Evolving GACC Behavior with Homogenous CNE

For a team of 30 robots, the population is initialized with 6000 randomly generated genotypes. Each genotype represents the hidden layer connection weights of one controller. Each genotype is encoded as vector of 420 floating point values (36 sensory inputs plus eight motor outputs multiplied by ten hidden layer neurons). A controller is derived via randomly selecting one genotype from the elite portion (table 6.5) of the population. This selected genotype is replicated 30 times in order to create a team of 30 clones.

Evolving GACC Behavior with Heterogenous CNE

For a team of 30 robots, the population is initialized with 6000 randomly generated genotypes. Each genotype represents the hidden layer connection weights of one controller. Each genotype is encoded as vector of 420 floating point values (36 sensory inputs plus eight motor outputs multiplied by ten hidden layer neurons). A controller is derived via randomly selecting 30 genotypes from the elite portion of the population, such that no genotype is selected more than once. The selected genotypes are decoded into controllers in order to create a team of 30 robots.

Tab. 6.4: Gathering and Collective Construction (GACC) Simulation Parameters. Parameters used in each GACC simulation.

GACC Simulation Parameters	
Complex Object Communication Range	1.0
Complex Object Communication Type	Broadcast
Wait with cargo time	30
Robot Movement Range	0.001
Robot Movement Cost	0.01
Object Detection Sensor Range	0.05
Object Detection Sensor Cost	Variable (section 6.2.1)
Robot Detection Sensor Range	0.05
Robot Detection Sensor Cost	0.01
Robot Detection Sensor Accuracy	1.0
Robot Initial Energy	1000 units
Initial Robot Positions	Random (Excluding home area)
Home Area Position	Environment's Center
Environment Width	1.0
Environment Height	1.0
Complex Objects to be Constructed	Variable (section 6.3)
Total Objects in Environment	Variable (section 6.3)
Total Type A Objects in Environment	Variable (section 6.3)
Total Type B Objects in Environment	Variable (section 6.3)
Total Type C Objects in Environment	Variable (section 6.3)
Object Distribution (Initial Positions)	Random
Robot Lifetime	3000 Iterations

Tab. 6.5: Neuro-Evolution (NE) Parameter Settings. Used by the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE methods.

Neuro-Evolution Parameter Settings	
Generations	250
Epochs	10
Simulation iterations per epoch (Robot lifetime)	3000
Mutation (per gene) probability	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	15 Generations (CONE)
Fitness stagnation W	10 Generations (CONE)
Genotype Distance (GD)	[0.0, 1.0] (CONE)
Specialization Distance (SD)	[0.0, 1.0] (CONE)
Population elite portion	50%
Weight (gene) range	[-10.0, +10.0]
Crossover	Single point
Sensory input neurons	36
Hidden layer neurons (Initial number)	10
Motor output neurons	8
Genotype	Input-output weights: One neuron (Multi-Agent ESP, CONE), All weights: One ANN controller (HomCNE, HetCNE, CCGA)
Total genotypes	15000
Genotype representation	Floating point value vector
Genotype populations	30 (CONE, Multi-Agent ESP, CCGA), 1 (HomCNE, HetCNE)
Genotype length	42 (CONE, Multi-Agent ESP), 420 (HomCNE, HetCNE, CCGA)
Genotypes per population	500 (CONE, Multi-Agent ESP, CCGA), 15000 (HomCNE, HetCNE)

Evolving GACC Behavior with CCGA

For a team of 30 robots, 30 genotype populations are created. Each population is initialized with 200 randomly generated genotypes. Each genotype is encoded as vector of 420 floating point values (34 sensory inputs plus eight motor outputs multiplied by ten hidden layer neurons). A controller is derived via randomly selecting one genotype from the elite portion of a given population. This process is then repeated 30 times, in order to derive 30 different controllers.

Evolving GACC Behavior with Multi-Agent ESP / CONE

For a team of 30 robots, both Multi-Agent ESP and CONE create 30 genotype populations for deriving 30 different controllers. Population i consists of u sub-populations, where u is the number of hidden layer neurons. Each population is initialized with 200 genotypes. Each genotype is encoded as vector of 42 floating point values(34 sensory inputs plus eight motor outputs). A controller is derived via randomly selecting one genotype from the elite portion of each sub-population for a given population. Selected genotypes then form the hidden layer of a controller. This process is repeated 30 times to derive 30 controllers.

Specific to CONE, the number of generations (V in section 3.7) which fitness progress can stagnate within one or more populations (n controllers) before the GD value is adapted (section 3.1.2) is presented as *fitness stagnation V* in table 6.5. The number of generations (W in section 3.7) which fitness can stagnate in any given population before the number of sub-populations is adapted (section 3.6), is presented as *fitness stagnation W* in table 6.5.

6.4 Task Results

This section describes results yielded from two experiment sets. *Atomic objects* are distributed throughout the environment and are the building blocks for complex objects. *Complex objects* are large objects that are constructed from a set of atomic objects. Objects must be delivered to a home area in a specific sequence in order for a complex object to be constructed.

1. *Experiment Set 1:* Demonstrates that not all environments are appropriate for deriving specialization during controller evolution.

2. *Experiment Set 2:* Establishes that certain environments are appropriate for deriving behavioral specialization over the course of controller evolution (section 6.4.2). Furthermore, this experiment set uses *shaping* [132] techniques together with a set of incrementally complex environments in order to derive incrementally complex team behaviors.

   This experiment set demonstrates that CONE consistently derives a set of *castes* [83] where the interaction of these castes produces a higher team task performance, comparative to that of HomCNE, HetCNE, CCGA, and Multi-Agent ESP evolved teams.

Tab. 6.6: *Simple Environments:* 10 objects are required in order to construct one complex object. *Object -A/-B/-C Number:* Number of type x objects. *Complex Objects:* Number of complex objects. *ENV:* Environment number.

ENV	Object-A Number	Object-B Number	Object-C Number	Complex Objects
1	10	0	0	1
2	0	10	0	1
3	0	0	10	1
4	20	0	0	2
5	0	20	0	2
6	0	0	30	3
7	30	0	0	3
8	0	40	0	4
9	0	0	40	4

One experiment consists of placing a team in a given environment and applying a given NE method to evolve controllers. Each experiment consists of two distinct phases: an *evolution phase* and a *testing phase*.

- *Evolution phase:* The controllers of a team are evolved for 250 generations (table 6.5) using a given NE method and a given environment.
- *Testing phase:* The fittest n controllers (the fittest team) are selected and executed in the same environment for one lifetime. The testing phase does not evolve controllers, so the evolved connection weights of each controller remains static. Task performances are averages calculated over 20 runs of the fittest controllers in a test environment.

In order to compare task performances yielded by two teams a statistical comparison is conducted between two given sets of task performance data. The following procedure is used.

- The Kolmogorov-Smirnov test [48] 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 task performance results of any two teams evolved by given methods, an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ.

Results of all statistical comparisons conducted in this chapter are presented in appendix D.

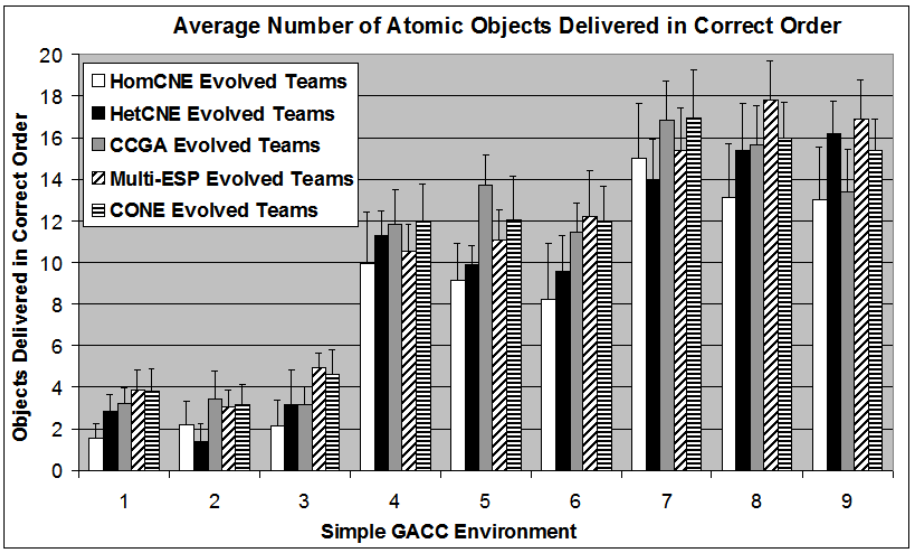

Fig. 6.7: Average Number of Objects Delivered in the Correct Order in each Simple Environment. By teams evolved by each method.

6.4.1 Experiment Set 1: Evolving Teams in Simple Environments

These experiments test the evolution of team behavior in environments that were found *not* to encourage emergent behavioral specialization during controller evolution. This environment set is the *simple environment set*. Each environment contains a different distribution of one object type (either A, B, or C). The object type distribution within each simple environment is presented in table 6.6. The simple environments were derived according to the supposition that if an environment consists of only one object type, then there will be no need for controllers to specialize to different behaviors in order to accomplish the task. Results yielded from team evolution in experiment set 1 elucidate that environments containing only one object type are not appropriate for encouraging emergent specialization. Figure 6.7 presents, for each simple environment, the average *number of objects delivered in the correct order* to the construction zone by teams evolved by each NE method. Appendix D presents, for each simple environment, the behavioral composition of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams.

6.4.2 Experiment Set 2: Shaping of Teams in Complex Environments

This section presents the results of applying HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in a set of shaping experiments. These shaping experiments use environments that were found to encourage emergent specialized behavior during controller evolution. These environments are called *complex*

Tab. 6.7: *Complex Environments:* A complex object is built from a combination of 10 object types. *Object -A/-B/-C Number:* Number of type x objects. *Complex Objects:* Number of complex objects. *ENV:* Environment.

ENV	Object-A Number	Object-B Number	Object-C Number	Complex Objects
1	1	2	7	1
2	2	4	4	1
3	3	6	1	1
4	8	2	10	2
5	10	4	6	2
6	10	16	4	3
7	10	12	8	3
8	16	14	10	4
9	15	20	5	4

environments. Each experiment evolves increasingly complex team behaviors in response to increasingly complex tasks. Shaping was selected as a technique to evolve complex team behaviors, since using direct evolution to solve complex GACC tasks was unsuccessful. Each shaping environment contains a given combination of type A, B, and C objects, and includes incrementally complex requirements for task accomplishment. Team behavior evolved for each task in each environment is then used as the starting point for the evolution of behavior in the next task and the next environment. The object type distribution within each shaping environment is presented in table 6.7. This distribution was derived according to the supposition that if an environment contains multiple object types, then there will be a requirement for controllers to specialize to different behaviors in order to accomplish the task with optimal performance.

Shaping of Team Behavior in Complex Environments

The following describes the decomposition of the GACC task, into eight difficulty levels, and the requirements for task completion at each difficulty level. In order for a complex object to be constructed, objects must be delivered to the construction zone either in any sequence of object types, or in predefined sequence of object types. Complex objects that require the delivery of objects in a predefined sequence of types are called *ordered complex objects*. Complex objects that can be constructed from objects delivered to the construction zone in any sequence of types are called *disordered complex objects*. For environments where more than one complex object is to be constructed, objects can be delivered such complex objects are concurrently constructed. For environments containing obstacles, these obstacles are placed in random locations. The task of the team in each shaping environment (*Env*) is to gather all objects for complex object construction.

Tab. 6.8: *Complex Object Sequence in Shaping Environments:* Up to four complex objects are to be constructed. *Complex Object-x Build Sequence:* The sequence in which type A, B, and C object types must be delivered. *ENV:* Shaping environment number. *NA:* Not Applicable.

ENV	**Complex Object-1 Build Sequence**	**Complex Object-2 Build Sequence**	**Complex Object-3 Build Sequence**	**Complex Object-4 Build Sequence**
1	CCACBCBCCC	*NA*	*NA*	*NA*
2	CAACBBBCBC	*NA*	*NA*	*NA*
3	BABAABCBBB	*NA*	*NA*	*NA*
4	BAABAACAAC	ACCCCCCCCA	*NA*	*NA*
5	BAABAABAAB	CACACACACC	*NA*	*NA*
6	BAAAAAAAAB	CABBBBBBAC	CBBBBBBBBC	*NA*
7	CACACBCBCB	CABABABABB	BABABABACC	*NA*
8	BABACABACB	CACACABABC	BABABABACB	CABACABABC
9	BABACABABB	CACACABABB	BABABABACB	BABABBBABB

Env 1. One *disordered complex object*, no obstacles and 10 objects.

Env 2. One *disordered complex object*, 10 obstacles and 10 objects.

Env 3. One *ordered complex object*, 10 obstacles and 10 objects.

Env 4. One *ordered complex object*, one *disordered complex object*, 10 obstacles and 20 objects.

Env 5. Two *ordered complex objects*, 10 obstacles and 20 objects.

Env 6. Two *ordered complex objects*, one *disordered complex object*, 10 obstacles and 30 objects.

Env 7. Three *ordered complex objects*, 10 obstacles and 30 objects.

Env 8. Three *ordered complex objects*, one *disordered complex object*, 10 obstacles and 40 objects.

Env 9. Four *ordered complex objects*, 10 obstacles and 40 objects.

Table 6.8 presents, for each environment, the sequence of atomic object types that complex objects must be constructed from. Figure 6.8 presents, for each complex environment, the average *number of atomic objects* delivered in the correct order, by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams. Appendix D presents the behavioral composition of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams. Behavioral composition refers to the composite number of robots in a team that are *non-specialized* or *specialized* to activating detection sensors or the gripper with a specific setting.

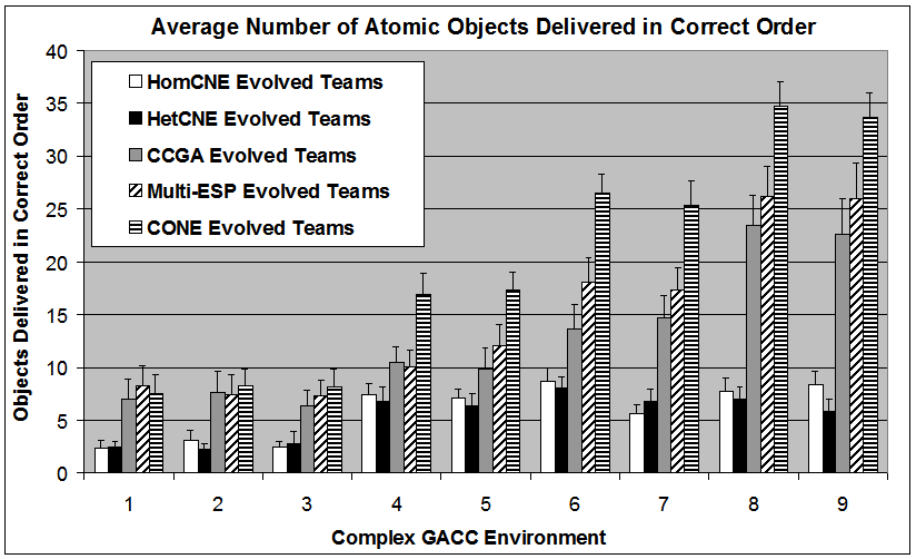

Fig. 6.8: Average Number of Objects Delivered in the Correct Order in each Complex Environment. By teams evolved by each method.

6.4.3 Comparing Task Performances of Teams Evolved in Simple versus Complex Environments

A statistical comparison of results is conducted for teams evolved in *simple* and *complex* environments. Figures 6.7 and 6.8 present the average *number of objects delivered in correct order* by teams evolved by each method, in the simple and complex environments, respectively. The methods used to evolve teams are HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE. Appendix D presents the results of this statistical comparison.

Results Summary: Teams Evolved in Simple and Complex Environments

Task performances yielded by teams evolved in *simple* and *complex* environments, indicate the following results. Average task performance refers to the average *number of objects delivered in correct order*. Optimal task performance is the maximum number of objects that can be delivered in the correct order.

1. For complex environments, results presented in figure 6.8, and the supporting statistical comparison presented in appendix D indicates the average task performance of CONE evolved teams is significantly higher than that of teams evolved by related methods.

2. For all *complex environments*, a comparison of behavioral compositions of the fittest teams evolved by CCGA, Multi-Agent ESP and CONE (appendix D), indicates that complex environments are appropriate for en-

couraging the evolution of teams consisting of multiple complementary *castes*.

3. In each complex environment, the fittest CONE evolved team consists of one non-specialized and multiple specialized castes (appendix D). This result supports the hypothesis that CONE is appropriate for deriving specialization such that teams achieve a higher task performance, comparative to that achieved by related methods.

4. For all complex environments, behavioral compositions and task performances of the fittest HomCNE and HetCNE evolved teams indicates that HomCNE and HetCNE are inappropriate for evolving behaviorally heterogenous teams (appendix D).

5. Average task performance yielded by CONE, Multi-Agent ESP, and CCGA evolved teams in the *complex environments* is significantly higher than that yielded by CONE, Multi-Agent ESP, and CCGA evolved teams in the *simple environments* (appendix D).

6. For all *complex environments* the fittest CCGA, Multi-Agent ESP and CONE evolved teams achieved an average of 0.71%, 0.79%, and 0.93% of optimal task performance (appendix D), respectively.

7. For all *simple environments* the fittest CCGA, Multi-Agent ESP and CONE evolved teams achieved 0.51%, 0.54%, and 0.55% of optimal task performance (appendix D), respectively.

8. In both the *simple* and *complex* environments, there is no significant difference between the average task performances of HomCNE and HetCNE evolved teams.

9. In the *complex environments* the fittest HomCNE and HetCNE evolved teams achieved 0.29% and 0.32% of optimal task performance (appendix D), respectively.

10. In the *simple environments* the fittest HomCNE and HetCNE evolved team achieved 0.48% and 0.47% of optimal task performance for all environments (appendix D), respectively.

Further analysis of these results is present in section 6.5.

6.5 Discussion of Results

This section discusses the role of emergent behavioral specialization in enabling CONE evolved teams to achieve a higher task performance comparative to related methods. Within this section, sections 6.5.1, 6.5.4 and 6.5.5 describe an analysis that evaluates the contribution of emergent behavioral specialization to collective behaviors derived during controller evolution. Section 6.5.6 describes an analysis of the function and contributions of the *genotype* and *specialization* difference metrics to emergent specialization in CONE evolved teams.

6.5.1 CCGA, Multi-Agent ESP, and CONE for Behavioral Specialization

The fittest teams evolved by CCGA, Multi-Agent ESP and CONE in the complex environments, consist of multiple complementary castes. Furthermore, according to statistical tests, the task performances of teams evolved by CCGA, Multi-Agent ESP and CONE in the complex environments is significantly higher comparative that of teams evolved in the simple environments. These results indicate that the cooperative co-evolutionary approaches of CCGA, Multi-Agent ESP, and CONE are better suited for attaining collective behavior solutions in the complex environments. That is, emergent castes benefit the performance of teams evolved in complex environments. This is exemplified by the multiple complementary castes derived by CCGA, Multi-Agent ESP, and CONE in the complex environments, and the corresponding task performance of the fittest teams evolved by these methods (appendix D).

6.5.2 CNE for Behavioral Specialization

The derivation of multiple castes does not occur in the fittest HomCNE and HetCNE evolved teams. In the case of the fittest teams evolved by HomCNE and HetCNE in both the simple and complex environments, a single caste is derived. Furthermore, there is no significant difference between the *average task performance* of teams evolved by HomCNE and HetCNE in both the *simple* and *complex* environments (appendix D). This result supports the notion that methods using cooperative co-evolution and a multiple population architecture (CCGA, Multi-Agent ESP and CONE) are better suited comparative to methods using a single population architecture (HomCNE and HetCNE) for evolving high performance teams in the complex environments.

6.5.3 The Role of Castes

For all complex environments, *specialized* (and *non-specialized*) castes emerge in the fittest CCGA, Multi-Agent ESP, and CONE evolved teams (appendix D). This is attributed to the nature of the task and environment.

Consider that at least two robots are required in order to deliver an object to the construction zone. However, before an object is cooperatively delivered, it must first be detected. The detection and gathering sub-tasks can be performed by individual robots. The detection of type A, B, or C objects requires that a robot's detection sensors be activated with setting A, B, or C, respectively. Similarly, the transportation of type A, B, or C objects requires that a robot's gripper be activated with setting A, B, or C, respectively. Hence, in order for a team to deliver an optimal number of objects to the construction zone, different robots are required to adopt specializations to either object detection or transportation. Specialization to either object detection or transportation results in the derivation of a set of complementary castes in a team. The interactions of these castes in turn results in a high performance team. The fittest teams evolved by CCGA, Multi-Agent ESP, and CONE, each consist of a set of specialized and non-specialized castes (appendix D).

Tab. 6.9: *Task Performance of Fittest HomCNE Evolved Teams (with Lesioned Non-Specialized Caste) in Complex Environments:* Results are percentages of original (unlesioned) performance.

Complex Environment Number	**Performance without Non-Specialized Robots**
1	76.78 %
2	80.01 %
3	78.45 %
4	84.50 %
5	82.12 %
6	70.56 %
7	77.90 %
8	82.34 %
9	78.95 %

As in the multi-rover case study (chapter 5), non-specialized castes are theorized to have emerged in response to task and environment requirements. Non-specialized robots are suitable for complementing specialized robots, since by definition, non-specialized robots do not dedicate a majority of their lifetime to a single behavior. The contribution of non-specialized and specialized castes to the task performance of the fittest CCGA, Multi-Agent ESP and CONE evolved teams is analyzed as part of the caste lesion study (section 6.5.4).

6.5.4 Caste Lesion Study

To investigate the role of emergent behavioral specialization in evolved teams a *caste lesion study* is conducted. The caste lesion study operates via removing sets of specialized or non-specialized controllers. These removed controllers are then replaced with heuristic controllers that implement a *hard-wired* specialized or non-specialized behavior (section 6.2.8). The task performance of these *lesioned* team is then evaluated in 20 new experimental runs for each complex environment, and an average task performance is calculated. The goal of the caste lesion study is to ascertain the contribution of specialized and non-specialized castes to the overall task performance of the fittest teams. The procedure used to lesion each of the fittest teams is described in appendix D. Tables 6.9, 6.10 6.11, 6.12 and 6.13 present the task performances of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE, respectively (in complex environments), with lesioned *non-specialized*, and object type A, B, and C detector, gatherer and constructor castes.

Results presented in tables 6.11, 6.12, and 6.13 indicate that task performances of the fittest teams CCGA, Multi-Agent ESP and CONE evolved teams depends upon the behaviors fulfilled by each of the castes as well as the interaction of these castes. Furthermore, these results indicate that the fittest CONE

Tab. 6.10: *Task Performance of Fittest HetCNE Evolved Teams (with Lesioned Non-Specialized Caste) in Complex Environments:* Results are percentages of original (unlesioned) performance.

Complex Environment Number	**Performance without Non-Specialized Robots**
1	82.81 %
2	81.15 %
3	85.65 %
4	79.58 %
5	80.25 %
6	76.68 %
7	79.97 %
8	72.66 %
9	83.53 %

Tab. 6.11: *Task Performance of Fittest CCGA Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original (unlesioned) performance. *O-A/O-B/O-C Detector:* Robots specialized to detecting. *O-A/O-B/O-C Gatherer:* Robots specialized to moving gripped objects. *O-A/O-B/O-C Constructor:* Robots specialized to placing gripped objects in the construction zone. *NA:* Not Applicable.

	Complex Environment Number								
Specialization	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O-A Detector	*NA*	*NA*	75.2%	65.1%	58.6%	54.6%	56.6%	51.1%	52.2%
O-B Detector	*NA*	45.7%	70.0%	*NA*	79.3%	55.3%	52.6%	67.7%	65.5%
O-C Detector	54.2%	69.1%	*NA*	63.3%	68.5%	72.4%	*NA*	77.4%	79.9%
O-A Gatherer	*NA*	*NA*	55.7%	59.9%	53.9%	56.1%	55.3%	52.1%	50.1%
O-B Gatherer	*NA*	66.2%	67.0%	*NA*	75.8%	61.5%	60.2%	68.2%	57.8%
O-C Gatherer	41.1%	64.2%	*NA*	60.2%	68.3%	70.9%	*NA*	66.3%	78.1%
O-A Constructor	*NA*	*NA*	64.5%	62.0%	58.8%	59.1%	65.0%	68.2%	71.9%
O-B Constructor	*NA*	66.7%	65.7%	*NA*	78.8%	58.9%	67.3%	72.6%	69.7%
O-C Constructor	63.1%	79.8%	*NA*	59.9%	68.9%	*NA*	*NA*	76.4%	*NA*
Non-Specialized	60.2%	58.9%	56.2%	62.1%	59.6%	64.8%	65.5%	87.6%	71.6%

Tab. 6.12: *Task Performance of Fittest MESP Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original (unlesioned) performance. *O-A/O-B/O-C Detector:* Robots specialized to detecting. *O-A/O-B/O-C Gatherer:* Robots specialized to moving gripped objects. *O-A/O-B/O-C Constructor:* Robots specialized to placing gripped objects in the construction zone. *NA:* Not Applicable.

	Complex Environment Number								
Specialization	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O-A Detector	*NA*	*NA*	73.1%	70.2%	68.0%	65.5%	61.0%	58.2%	62.4%
O-B Detector	*NA*	75.7%	58.7%	*NA*	87.1%	62.3%	63.7%	72.4%	67.6%
O-C Detector	55.4%	64.3%	*NA*	68.4%	65.0%	84.5%	*NA*	84.9%	77.8%
O-A Gatherer	*NA*	*NA*	51.3%	66.7%	71.1%	64.3%	66.3%	61.1%	65.4%
O-B Gatherer	*NA*	69.8%	61.9%	*NA*	80.0%	77.9%	65.3%	64.2%	67.9%
O-C Gatherer	49.9%	58.7%	*NA*	68.2%	69.4%	80.9%	*NA*	80.7%	78.2%
O-A Constructor	*NA*	*NA*	70.1%	58.9%	57.7%	66.2%	65.3%	64.0%	61.3%
O-B Constructor	*NA*	75.8%	73.3%	*NA*	71.0%	67.2%	68.8%	70.7%	60.1%
O-C Constructor	65.8%	67.1%	*NA*	55.4%	63.3%	*NA*	*NA*	74.5%	76.0%
Non-Specialized	52.5%	55.7%	58.6%	53.8%	62.4%	68.8%	67.9%	88.0%	*NA*

Tab. 6.13: *Task Performance of Fittest CONE Evolved Teams (with Lesioned Castes) in Complex Environments:* Results are percentages of original (unlesioned) performance. *O-A/O-B/O-C Detector:* Robots specialized to detecting. *O-A/O-B/O-C Gatherer:* Robots specialized to moving gripped objects. *O-A/O-B/O-C Constructor:* Robots specialized to placing gripped objects in the construction zone. *NA:* Not Applicable.

	Complex Environment Number								
Specialization	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O-A Detector	*NA*	*NA*	70.3%	64.5%	60.0%	63.2%	59.6%	57.3%	51.1%
O-B Detector	*NA*	58.2%	62.2%	*NA*	68.1%	65.6%	60.0%	57.7%	56.8%
O-C Detector	45.3%	58.7%	*NA*	60.3%	61.8%	70.2%	*NA*	72.4%	65.8%
O-A Gatherer	*NA*	*NA*	63.8%	62.6%	66.8%	55.8%	53.3%	54.8%	55.6%
O-B Gatherer	*NA*	60.2%	56.8%	*NA*	71.5%	61.2%	57.5%	58.1%	56.0%
O-C Gatherer	40.2%	54.8%	*NA*	45.6%	58.4%	69.2%	*NA*	65.7%	57.2%
O-A Constructor	*NA*	*NA*	72.2%	70.1%	68.6%	65.7%	62.4%	65.0%	48.8%
O-B Constructor	*NA*	65.8%	67.7%	*NA*	64.8%	70.0%	63.3%	61.0%	49.6%
O-C Constructor	60.8%	64.2%	*NA*	69.9%	65.3%	*NA*	*NA*	66.5%	52.7%
Non-Specialized	41.3%	48.6%	45.7%	52.2%	62.1%	61.9%	60.6%	75.7%	*NA*

evolved teams (for all complex environments) have a greater reliance upon constituent castes, comparative to the fittest CCGA and Multi-Agent ESP evolved teams. However, the task performance of the fittest CCGA and Multi-Agent ESP evolved teams is more robust when castes are removed and replaced with heuristic controllers. This indicates that there is less of an inter-dependency between constituent castes in the fittest CCGA and Multi-Agent ESP teams, comparative to the fittest CONE evolved teams. The fittest CCGA and Multi-Agent ESP evolved teams tend to rely less upon the interactions between different castes. An analysis of the inter-dependency between non-specialized and specialized castes is presented as part of the behavioral validation study in section 6.5.5. The smallest difference between original task performance and the task performance of lesioned teams was measured for the fittest HomCNE and HetCNE evolved teams (tables 6.9 and 6.10, respectively). This supports previous results indicating that teams comprised of multiple complementary specialized and non-specialized castes are required in order for the team to attain a near optimal task performance (section 6.4.3).

6.5.5 Validating the Role of Behavioral Specialization

In order to evaluate the contribution of specialized and non-specialized behaviors to a team's task performance, a set of behavioral validation experiments are executed. These behavioral validation experiments investigate the contribution of specialized and non-specialized castes to team task performance, as well as the inter-dependency between castes. Each behavioral validation experiment executes a team where each robot uses a fittest specialized controller. The *team label* and *behavioral specialization of each controller* of such teams, for each behavioral validation experiment (*Exp*), is delineated in the following.

1. *Exp 1:* Object-A Detector, Type A object detection.
2. *Exp 2:* Object-B Detector, Type B object detection.
3. *Exp 3:* Object-C Detector, Type C object detection.
4. *Exp 4:* Object-A Gatherer, Type A object gathering.
5. *Exp 5:* Object-B Gatherer, Type B object gathering.
6. *Exp 6:* Object-C Gatherer, Type C object gathering.
7. *Exp 7:* Object-A Constructor, Type A object construction.
8. *Exp 8:* Object-B Constructor, Type B object construction.
9. *Exp 9:* Object-C Constructor, Type C object construction.
10. *Exp 10:* Non-Specialized, *Non-specialized*.

The setting of an identical behavior for each controller in a team is done via initializing each robot with the fittest controller selected from either the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP or CONE evolved team (for each complex environment). The selected controller is the fittest that exhibits either a specialized detector, gatherer or constructor behavior, or a non-specialized behavior. Such controllers are always selected from CONE evolved teams, since CONE evolved teams yielded the highest task performance in all complex environments (figure 6.8). For each behavioral validation experiment, each selected controller is cloned 20 times in order to create a behaviorally homogenous team. For clarity, teams constructed for the behavioral validation experiments are herein referred to as *cloned teams*. Each clones team is executed in each complex environment for 20 experimental runs. For each behavioral validation experiment, robots and obstacles are placed in random locations, and executed for one *lifetime*. These experiments do not employ any adaptive methods, so task performances achieved by cloned teams result from the interactions of individual robots over the course of a lifetime.

Task performances achieved by cloned teams consisting of controllers specialized to detecting, gathering, and constructing behaviors are presented in appendix D. Teams consisting entirely of robots specialized to the *move* behavior are not tested given that the move action does not directly contribute to the detection or transportation of objects to the construction zone. The move behavior is represented by gatherer teams.

Results of Behavioral Validation Experiments

This section presents the results of statistical tests that compare the task performances of cloned teams (constructed for the behavioral validation experiments) with that of HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams. The results of statistical comparisons between cloned and evolved teams are presented in table appendix D. The average task performance of all the cloned teams is significantly lower comparative to that of evolved teams (appendix D). These results indicate that behaviorally homogenous teams are insufficient for achieving task performances comparable to that achieved by evolved teams.

These behavioral validation experiments support previous results that indicate that a near optimal task performance mandates teams comprised of multiple complementary castes (section 6.4.3). These experiments also support previous results that indicate that the high task performance yielded by the fittest CCGA, Multi-Agent ESP, and CONE evolved teams is a consequence of the interaction and inter-dependency between non-specialized and specialized castes. This is supported by the lack of significant difference between the task performances of the non-specialized cloned teams (section 6.5.5) and teams evolved by HomCNE and HetCNE (section 6.4.2).

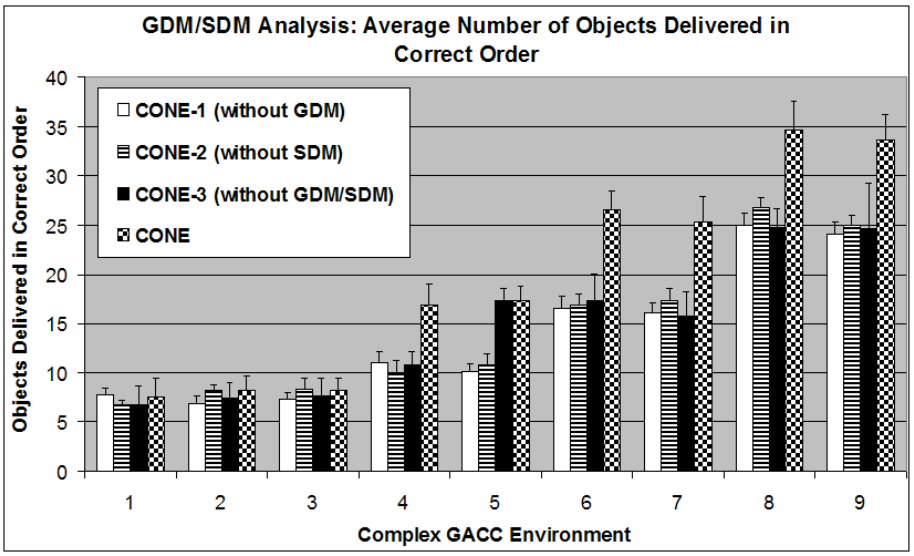

Fig. 6.9: Number of Objects Delivered in Correct Order by Teams evolved by CONE and CONE Variants. CONE, CONE-1, CONE-2, CONE-3.

6.5.6 The Role of Difference Metrics in CONE

This section examines the role of the *Genotype Difference Metric* (GDM) and *Specialization Difference Metric* (SDM) for facilitating behavioral specialization and increasing task performance in CONE evolved teams. As part of this analysis, three variants of CONE are executed.

1. *CONE without GDM (CONE-1):* Teams are evolved using CONE without the GDM. The SDM remains active.
2. *CONE without SDM (CONE-2):* Teams are evolved using CONE without the SDM. The GDM remains active.
3. *CONE without GDM and SDM (CONE-3):* Teams are evolved using CONE without both the GDM and SDM.

Figure 6.9 presents the average task performance of teams evolved using CONE, without the GDM, SDM, and both the GDM and SDM, for all complex environments. These results are averaged over 20 experimental runs for each CONE variant in each environment. For comparison, figure 6.9 also presents results previously attained by CONE evolved teams using the original experimental setup. The results of a statistical comparison between CONE and its variants are presented in appendix D. This results comparison indicates that, for all complex environments, there is a significant difference between the task

performance of teams evolved by CONE and CONE-1, CONE-2, and CONE-3. That is, teams evolved by CONE yield a performance advantage over the CONE variants. This result supports the hypothesis that both the GDM and SDM are 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 caste lesion study (section 6.5.4) supports the hypothesis that the GDM and SDM enables CONE to derive specializations appropriate for achieving a higher task performance, comparative to related methods.

6.6 Conclusions

This chapter investigated the application of CONE and related methods for evolving collective gathering and construction behaviors in a simulated robot team. This task mandated behavioral specialization, since different object types required that robots specialize to gathering different object types in order for a team to maximize the number of structures built. Teams evolved with CONE were found to yield a higher task performance comparative to related methods. An experimental analysis revealed that CONE derived a set of behavioral specializations that were found to be necessary in order to achieve a task performance that surpassed that achieved by related methods.

7. DISCUSSION AND FUTURE DIRECTIONS

This thesis presented and evaluated *Collective Neuro-Evolution* (CONE), which is a method for the automated controller design in artificial collective behavior systems. In this chapter, the contributions of CONE, and the implications of these contributions are discussed. Future research directions are also presented and discussed in the context of controller design methods that utilize emergent phenomena in order to solve collective behavior tasks.

7.1 Evolving Controllers in Collective Behavior Systems

CONE is a principled method for automated controller design in collective behavior systems. CONE works via designing (evolving) sets of ANN controllers that such that they cooperate in order to accomplish a given collective behavior task. CONE uses a cooperative co-evolution process in order to evolve, evaluate, and propagate controller behaviors via simultaneously exploring different niches of the genotype space. To demonstrate the efficacy of CONE as a controller design method it was applied in three collective behavior case studies. The collective behavior case studies investigated were the *pursuit-evasion* (chapter 4), the *multi-rover* (chapter 5), and the *gathering and collective construction* task (chapter 6). Each of these was a complex non-linear control task that required at least two controllers with minimal *a priori* knowledge, to work cooperatively. Also, in each case study, task accomplishment mandated that controllers adopt complementary behavioral specializations that worked well together.

7.1.1 Contributions of CONE

An analysis of experimental results yielded from each of the collective behavior case studies elucidated the following contributions of CONE.

1. The cooperative co-evolution process of CONE requires *both* behavioral specialization *and* genotype difference metrics in order to evolve specialized controllers. The behaviors of these controllers complement each other for the purpose of producing a collective behavior solution. This was illustrated in the experimental analysis of each of the case studies.

2. CONE dynamically identifies the degree of behavioral specialization required by a given collective behavior task. CONE then facilitates the behavioral specialization appropriate for controllers to optimally (or near

optimally) solve the task. This is accomplished by complex interaction of the behavioral specialization and genotype difference metrics, which adaptively directs a search for beneficial specialized behaviors.

3. Given that CONE effectuates behavioral specialization in controllers, CONE is most appropriately applied to collective behavior tasks that require behaviorally specialized controllers, where such controllers must cooperate in order to produce a collective behavior solution.

7.1.2 Emergent Specialization and CONE

The collective behavior case studies indicated that if a task does not require, or does not benefit from specialization, then CONE will not facilitate emergent behavioral specialization. CONE uses a multiple population cooperative coevolution architecture that implements behavioral specialization and genotype difference metrics. The function of these metrics is described in the following.

1. *Genotype difference metric:* Operates via encouraging the recombination of similar genotypes in different populations. Similarities of genotypes are measured according to average connection weight differences of the neurons that they encode. The genotype metric encourages the recombination of genotypes that are specialized to similar functions, where such functions beneficially contribute to controller performance. However, similar genotypes in different populations may encode very different functionalities, so recombination may produce genotypes (neurons) that do not work in a controller. The specialization metric addresses this problem.

2. *Behavioral specialization difference metric:* Operates via encouraging the recombination of controllers that are specialized to similar behaviors, where such specializations benefit task performance. If the degree of behavioral specialization measured for each controller is too high for producing increasingly fit controllers, then the specialization metric allows for the recombination of controllers that exhibit less behavioral similarities. If the degree of specialization measured for each controller is too low for increasing controller fitness, then the specialization metric restricts controller recombination to those that exhibit more behavioral similarities. Regulation recombination between populations (controllers) has the affect of propagating behaviorally specialized controllers that beneficially contribute to collective behavior task performance.

An experimental analysis conducted for each case study indicated two important results. First, without either the GDM or SDM, CONE evolved teams lose their advantage of a significantly higher task performance. Second, genotypes in different populations (of behaviorally similar controllers) needed to be very similar in order to be recombined. 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. Hence, 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. Also, the interaction of the SDM and GDM regulates recombination between populations such that the chances of producing deleterious offspring is minimal. This aids in directing a search of the genotype space such that beneficial niches in the solution space (high performance controllers) are identified and propagated.

7.1.3 Neuro-Evolution as a Controller Design Method

In the collective behavior case studies, CONE was comparatively evaluated with related controller design methods and was demonstrated as yielding a significantly higher task performance. This is a result which supports related research [49], [175], [13], [21] that has demonstrated NE as an effective approach to controller design in artificial collective behavior systems. Applying NE for controller design is especially prevalent in multi-robot systems, where the goal is to evolve ANN controllers for simulated multi-robot systems, and then to transfer evolved controllers to a counter-part physical multi-robot system [124].

7.2 Future Directions

7.2.1 A General Specialization Metric

The specialization metric (section 3.2.2) used by CONE, makes certain assumptions regarding a controller's behavior. For example, that a controller's behavior is determined by v distinct motor outputs, and that behavioral specialization exhibited by a controller is definable by the frequency with which the controller switches between executing each of its motor outputs. In order to broaden the applicability of CONE to a diverse range of collective behavior systems, a general definition of specialization that does not place constraints on controller architecture, or impose a designer specified threshold for when a controller is *specialized*, would be required. Currently, there is no canonical definition of behavioral specialization, and consequently there exists a disparate range of methods for measuring behavioral specialization in single or multi-agent systems. Section 2.1.3 overviews collective behavior specialization metrics.

A general definition of specialization that accounts for any type of emergent specialization, such as behavioral and morphological, that operates in collective behavior systems, would significantly contribute to research in automated controller design methods, and could also contribute to research in biological fields. Such a generalized specialization definition would allow the development of methods that dynamically evolve collective behavior solutions via using emergent phenomena as a problem solver, as well as potentially elucidating the mechanisms that lead to specialization in nature. The research presented in this thesis was a first step in deriving an automated controller design method

that evolves collective behavior solutions via effectuating and using emergent phenomena as part of the problem solving process.

7.2.2 Introducing Plasticity into the CONE Architecture

In the current CONE architecture the number of genotype populations equals the number of controllers in the collective behavior system. The number of controllers is specified *a priori* by the system designer. A more flexible approach would be to begin with a single population of genotypes, and have the CONE process adapt the number of populations (that is, *species*) in response to task and environment constraints. The goal of such an approach would be to have species dynamically emerge in response to identified subtasks in a given collective behavior task. Ideally, each species would be suited for solving a specific subtask, and together the species would produce a collective behavior solution. A self adapting population that is able to dynamically split into multiple species as a function of identified subtasks would afford a collective behavior controller design method with a high degree of flexibility, adaptability, and applicability. Such an approach was proposed by Potter [136] and applied by Stanley [158] in order to solve various single agent (non collective behavior) tasks [160], [159].

CONE uses direct encoding in order to map genotypes to controllers. The number of sensory inputs and motor outputs is static and needs to be specified by the system designer according to collective behavior task requirements. A beneficial extension to this architecture would be for CONE to employ a developmental approach that derives a number of sensory input and motor output neurons in response to task and environment constraints. The implementation of developmental mechanisms for evolving controller topology, as employed by NEAT [158] and HyperNEAT [162], would allow each controller to more efficiently and effectively adapt to, and solve subtasks that constitute any given collective behavior task. For example, in a given collective behavior task, certain subtasks may require a complex sensory input to motor output mapping in order to accomplish, whilst other subtasks may require relatively few sensory inputs and simple motor outputs. A developmental approach to solving collective behavior tasks would alleviate the need for a system designer to specify the architecture and number of controllers *a priori*. Such an approach would also have important consequences for real world collective behavior applications, since a developmental controller design method would be highly adaptive and require minimal knowledge of the environment or how to solve a given task.

Appendices

APPENDIX A: STATISTICAL COMPARISONS IN THE PURSUIT-EVASION TASK

This appendix presents statistical comparisons of task performances of evolved teams, and an analysis of the role of the genotype and specialization difference metrics in the task performances of evolved teams.

Prey Capture Time Comparisons

Figure 4.10 presents average prey capture times (calculated over 20 experimental runs) yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved groups for all group types. Figure 4.10 indicates that CONE evolved groups yield comparatively higher performances for group types 4, 5, 8, 9, and 10, and comparable performances for groups types 1, 2, 3, 6, and 7.

In order to draw conclusions from this comparative study, a set of statistical tests are performed in order to gauge task performance differences between respective method results. Data sets representing results of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE methods are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48]. Specifically, P = [0.42, 0.36, 0.32, 0.79] is calculated for the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions, respectively. To determine if there is a statistically significant difference between method results presented in figure 4.10 an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ. Table A-1 presents the P values for t-tests conducted between *average prey capture times* yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved groups for all group types. In table A-1, a value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

The Role of Difference Metrics in CONE

This analysis supports the efficacy of the *Genotype Difference Metric* (GDM) and the *Specialization Difference Metric* (SDM) for facilitating behavioral spe-

Tab. A-1: *Statistical Comparison of Prey Capture Times:* T-test values for HomCNE, HetCNE, CCGA, MESP, and CONE evolved groups. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

	Group Type									
Group	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE vs CCGA	0.0001	*0.11*	*0.37*	0.0001	0.0001	*0.89*	*0.62*	0.0001	0.0001	0.077
HomCNE vs MESP	0.0001	0.044	0.0078	0.0001	0.0001	*0.75*	*0.70*	0.0001	0.0001	*0.54*
HomCNE vs CONE	0.0001	0.0015	*0.84*	0.0001	0.0001	*0.12*	*0.073*	0.0001	0.0001	0.0001
HomCNE vs HetCNE	*0.19*	*0.15*	*0.84*	*0.21*	*0.13*	*0.12*	*0.17*	*0.26*	*0.45*	*0.13*
HetCNE vs CCGA	0.0001	*0.11*	*0.37*	0.0001	0.0001	*0.89*	*0.62*	0.0001	0.0001	0.077
HetCNE vs MESP	0.0001	0.044	0.0078	0.0001	0.0001	*0.75*	*0.70*	0.0001	0.0001	*0.54*
HetCNE vs CONE	0.0001	0.0015	*0.84*	0.0001	0.0001	*0.12*	*0.073*	0.0001	0.0001	0.0001
CCGA vs MESP	*0.47*	*0.42*	*0.24*	*0.75*	*0.069*	*0.75*	*0.071*	*0.25*	*0.79*	*0.27*
CCGA vs CONE	*0.16*	*0.081*	*0.11*	0.0001	0.0001	*0.28*	*0.28*	0.0001	0.0001	0.0001
MESP vs CONE	*0.46*	*0.093*	*0.055*	0.0001	0.0001	*0.14*	0.11	0.0001	0.0001	0.0001

Tab. A-2: The role of GDM and SDM in CONE: Comparative t-test results of teams evolved with CONE and CONE variants. without the GDM (Genotype Difference Metric), SDM (Specialization Difference Metric), GDM and SDM. *CONE:* Team evolved with original CONE setup (both GDM and SDM active). *CONE-1:* Team evolved by CONE without GDM. *CONE-2:* Team evolved by CONE without SDM. *CONE-3:* Team evolved by CONE without GDM and SDM.

		Group Type									
Predator Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
CONE-1	*vs*	*0.17*	*0.19*	*0.22*	*0.23*	*0.55*	*0.52*	*0.51*	*0.57*	*0.42*	*0.45*
CONE											
CONE-2	*vs*	*0.29*	*0.32*	*0.33*	*0.40*	*0.38*	*0.43*	*0.29*	*0.11*	*0.16*	*0.23*
CONE											
CONE-3	*vs*	*0.44*	*0.29*	*0.32*	*0.41*	*0.42*	*0.55*	*0.42*	*0.42*	*0.55*	*0.58*
CONE											

cialization, and increasing task performance in CONE evolved teams. As part of this analysis, CONE is re-executed with the following experimental setups.

1. *CONE without GDM (CONE-1 in table A-2:* Predator teams are evolved by CONE without the GDM. The SDM for inter-population genotype recombination remains active.

2. *CONE without SDM (CONE-2 in table A-2:* Teams are evolved by CONE without the SDM. The GDM remains active.

3. *CONE without GDM and SDM (CONE-3 in table A-2:* Teams are evolved by CONE without the GDM and SDM.

Figure 4.11 presents prey-capture times that result from applying each of the variants to the original CONE experimental setup (CONE-1, CONE-2, and CONE-3) for team types 1 to 10. Prey-capture results are averaged over 20 experimental runs. For comparison, results previously attained by CONE evolved teams are also presented in figure 4.11. A statistical comparison of results presented in figure 4.11 indicate that teams evolved by CONE without the GDM (CONE-1), SDM (CONE-2), and both the GDM and SDM (CONE-3), yielded a significantly lower task performance comparable to CONE evolved teams (for all team types except 1, 6 and 7). Table A-2 in appendix A, presents t-test values resulting from this statistical comparison. That is, for a majority of the team types tested, both the GDM and SDM are beneficial in terms of increasing the task performance of CONE evolved predator teams.

APPENDIX B: EMERGENT BEHAVIORS IN THE PURSUIT-EVASION TASK

This appendix presents ancillary information and results for the pursuit-evasion collective behavior case study presented in chapter 4. In particular, the methods used to define and reproduce individual and collective behaviors observed in the pursuit-evasion simulations are elaborated upon. Furthermore, ranges of sensor activation values and motor output activation values corresponding to defined individual and collective behaviors are presented.

In order to reproduce individual predator behaviors, and collective prey-capture behaviors, the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams are executed in a new set of pursuit-evasion experiments. For each of the fittest teams, 20 new experimental runs are executed. Re-executing the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams in a new set of simulations allows for the measurement of predator light and proximity sensor readings.

For each predator, these sensor readings are correlated with specific motor outputs, where distinct ranges of motor outputs correspond to distinct individual behaviors. In certain simulation instances, the interaction of such individual behaviors produces a collective prey-capture behavior.

Defining Individual Predator Behaviors

For each experiment that re-executes a fittest team, 20 instances of individual behaviors are randomly sampled. Sensor values are then measured for the duration of an individual behavior. The duration of an individual behavior is defined by the time that a given predator is within proximity sensor range of at least one other predator, and within light sensor range of a prey.

Sensor values are measured for each light and proximity input neuron. Averages values are then calculated (over all predators in a given fittest team) for each proximity and light input neuron. These average values for each sensory input neuron are used to reproduce individual predator behaviors. Individual predator behaviors are reproduced by manually setting the sensory input neurons of a predator's ANN controller with the average activation value calculated for each sensory input neuron.

Given the measurement of different sensor value activation ranges, five distinct individual predator behaviors are defined. These individual behaviors are labeled as follows: *Pursuer*, *Blocker*, *Flanker*, *Knocker*, and *Idle*. The value

range of light and proximity sensor values is divided into ten segments in the range: [0, 1]. The portion of a given behavior's duration that the light or proximity sensors are active is then measured for each of the ten segments.

Defining Collective Prey-Capture Behaviors

The interaction of at least two individual predator behaviors produce a prey-capture behavior. The interaction of individual behaviors is defined as two predators being within proximity sensor range of each other and within light sensor range of a prey.

Average proximity and light sensor values for each of the prey-capture behaviors are not presented, since such values are simply an average of at least two individual predators measured over the course of a prey-capture behavior. Where, the course of a prey-capture behavior is defined as the time for which at least two predators are within proximity sensor range of each other and light sensor range of a prey.

Furthermore, prey-capture behaviors are reproduced by first reproducing at least two individual predator behaviors. Various combinations of individual behaviors are then combined until the prey-capture behaviors observed for each the fittest teams have been reproduced.

Given this scheme for reproducing prey-capture behaviors, five prey-capture behaviors observed in experiments that re-executed the fittest teams are reproduced. These prey-capture behaviors are labeled as follows: *Entrapment*, *Pursuer-blocker*, *Role-switcher* and *Spiders-fly*.

Calculating Sensor Values for Individual Behaviors

For a given fittest team, the average proximity and light sensor activation values for individual predator behaviors are calculated in the following manner.

1. For the duration that a given predator is within proximity sensor range of at least one other predator and within light sensor range of a prey, the activation values of all infrared proximity and light sensors are recorded. This duration constitutes the occurrence of an individual predator behavior.

2. The average value of each proximity and light sensor input neuron is calculated for the duration of the individual behavior.

3. The average of all proximity and light sensor input neurons is then calculated for the duration of this individual behavior.

4. An average proximity and light sensor value is then calculated over all of the 20 sampled individual predator behaviors.

This process is repeated for each predator in a given fittest team.

Figure B-1 presents the average light sensor values measured over 20 samples of the *pursuer* individual predator behavior. Figure B-2 presents the average

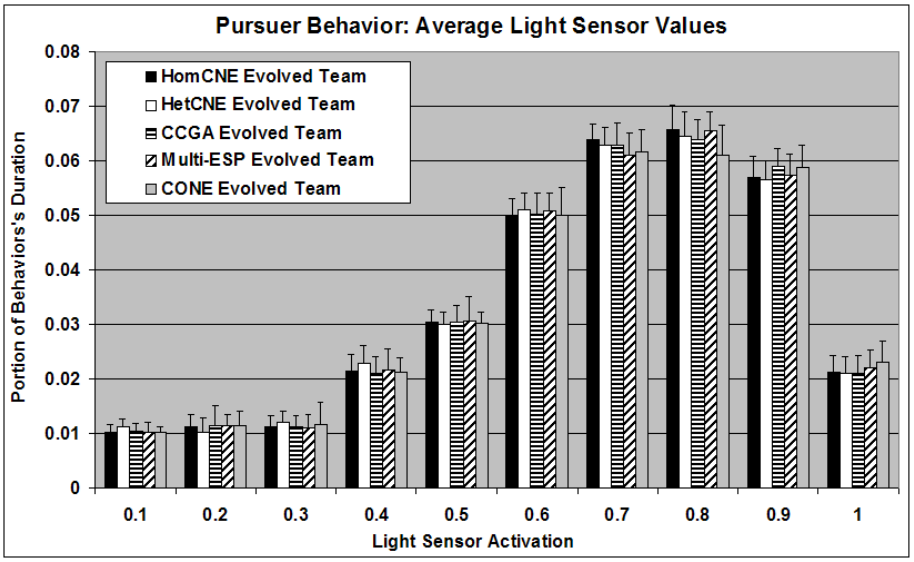

Fig. B-1: Pursuer Behavior: Average light sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

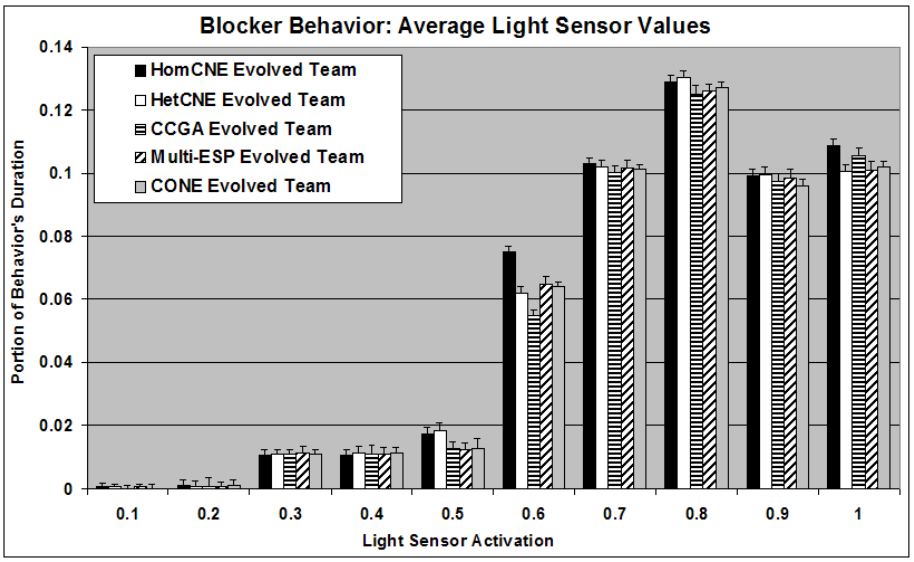

Fig. B-2: Blocker Behavior: Average light sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

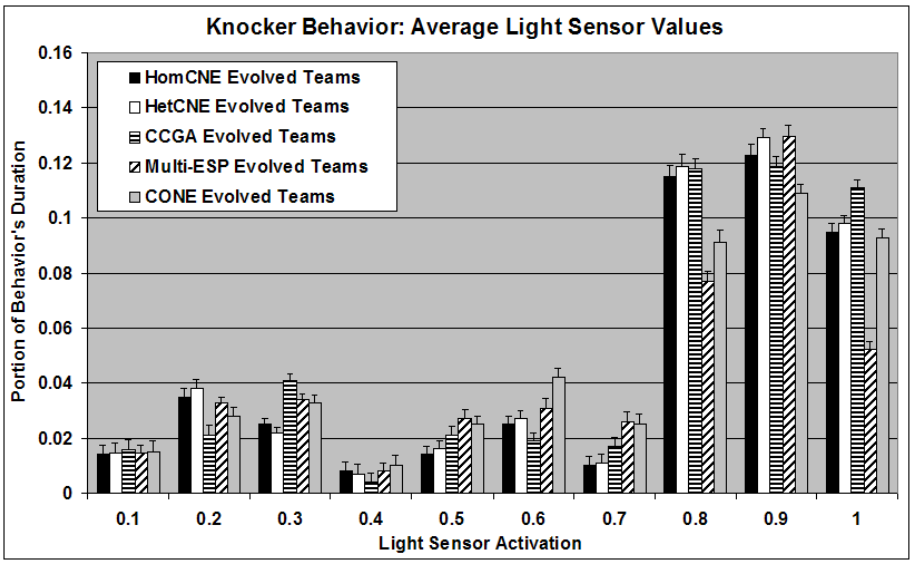

Fig. B-3: Knocker Behavior: Average light sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

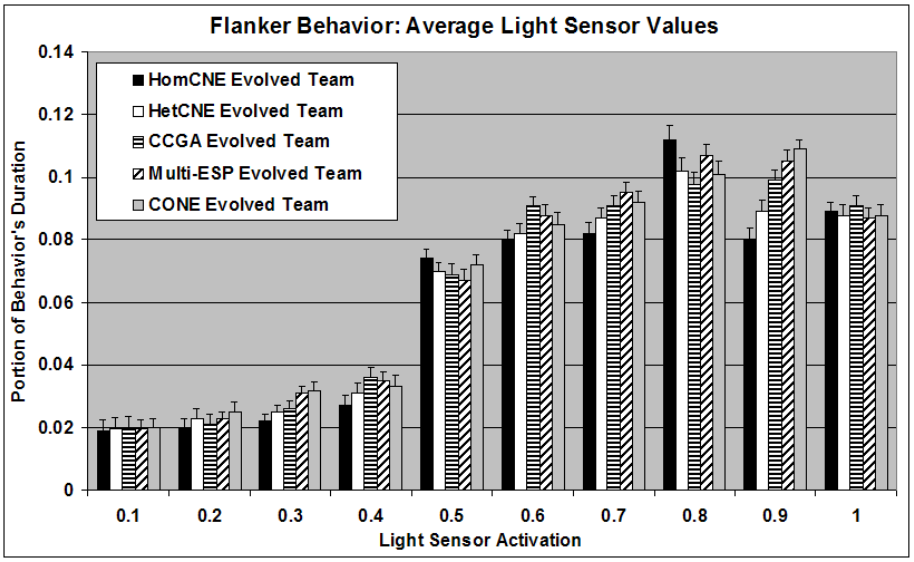

Fig. B-4: Flanker Behavior: Average light sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

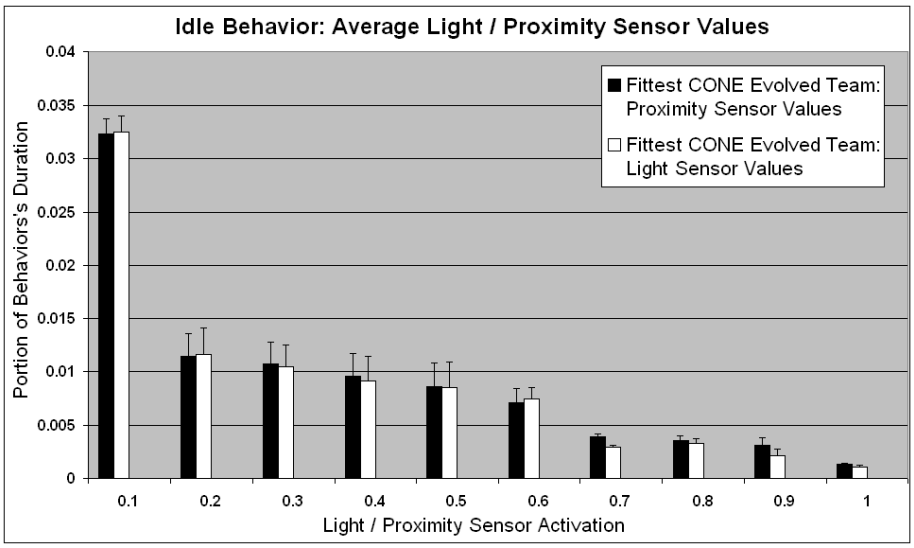

Fig. B-5: Idle Behavior: Average light and proximity sensor values for each range segment, for all predators in the fittest CONE evolved team.

light sensor values measured over 20 samples of the *blocker* individual predator behavior. Figure B-3 presents the average light sensor values measured over 20 samples of the *knocker* individual predator behavior. Figure B-4 presents the average light sensor values measured over 20 samples of the *flanker* individual predator behavior. Figure B-5 presents the average light and proximity sensor values measured over 20 samples of the *idle* individual predator behavior.

Figure B-6 presents the average proximity sensor values measured over 20 samples of the *pursuer* individual predator behavior. Figure B-7 presents the average proximity sensor values measured over 20 samples of the *blocker* individual predator behavior. Figure B-8 presents the average proximity sensor values measured over 20 samples of the *knocker* individual predator behavior. Figure B-9 presents the average proximity sensor values measured over 20 samples of the *flanker* individual predator behavior.

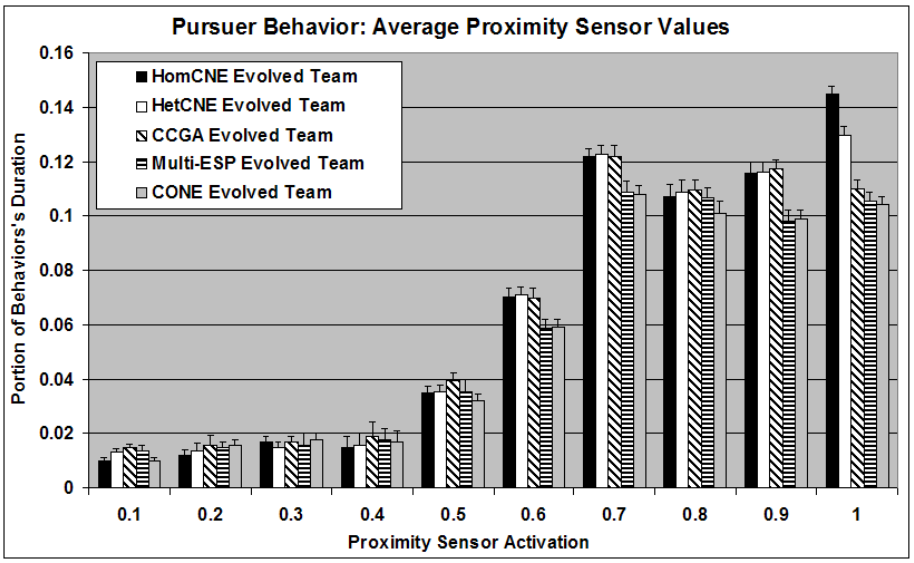

Fig. B-6: Pursuer Behavior: Average proximity sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

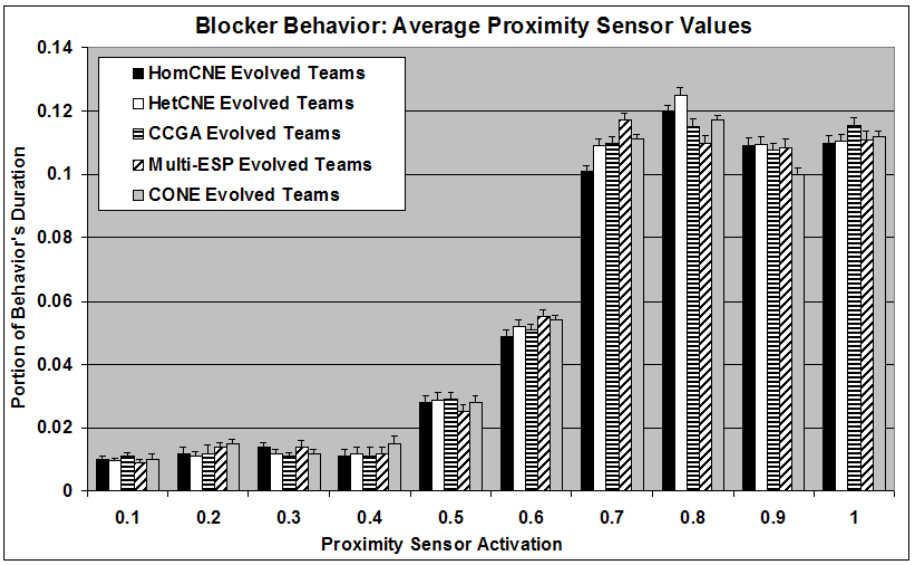

Fig. B-7: Blocker Behavior: Average proximity sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

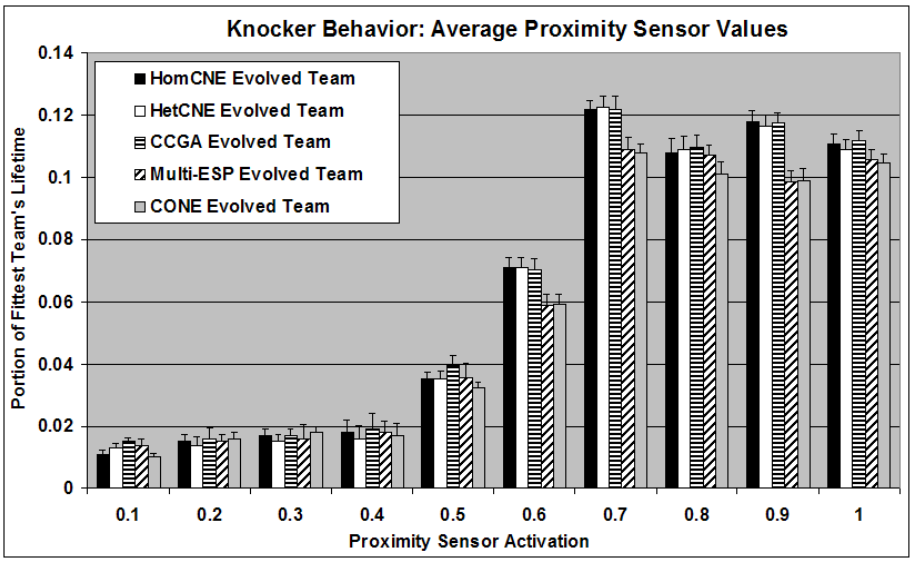

Fig. B-8: Knocker Behavior: Average proximity sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

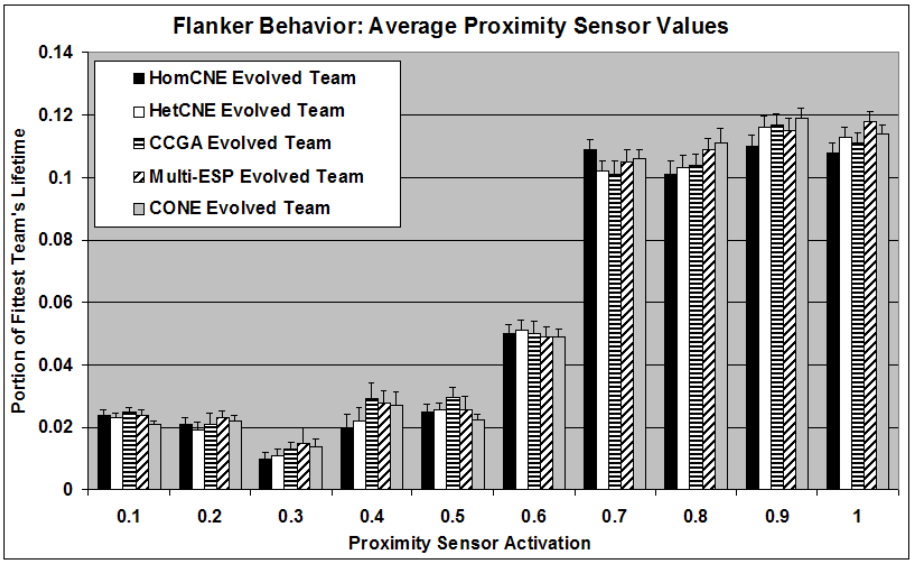

Fig. B-9: Flanker Behavior: Average proximity sensor values for each range segment, for all predators in the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams.

APPENDIX C: STATISTICAL COMPARISONS IN THE MULTI-ROVER TASK

This appendix presents a statistical comparison of task performance results and an analysis of emergent collective behaviors exhibited by evolved teams.

Behavioral Compositions of Fittest Evolved Rover Teams in Simple Environments

Tables C-1, C-2, C-3, C-4, and C-5 present the behavioral composition of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams, respectively, where these teams were evolved in the simple environments. In these tables, *red rock distribution* (RRD) refers to one of the ten environments within the simple environment set. Red rock distributions are described in section 5.1.4. The behavioral compositions of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE are presented for each of the *simple environments*. Behavioral composition refers to the composite number of rovers in each team that are *non-specialized* or *specialized* to activating red rock detection sensors with either the *low-res*, *med-res*, or *hi-res* settings (section 5.2.6).

Behavioral Compositions of Fittest Evolved Rover Teams in Complex Environments

Tables C-6, C-7, C-8, C-9, and C-10 present the behavioral compositions of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE in the complex environments. In these tables, *Red Rock Distribution* (RRD) refers to one of the ten environments within the complex environment set, where each environment corresponds to a given red rock distribution (section 5.1.4). Behavioral compositions of evolved teams are defined according to the number of rovers that adopt either the *low-res*, *med-res*, or *hi-res* detector specialization, or no behavioral specialization (section 5.2.6).

Tab. C-1: *Behavioral Compositions of Fittest HomCNE Evolved Teams in Simple Environments:* For teams of 20 rovers. *HomCNE:* Homogenous Conventional NeuroEvolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	20	0	0	0
2	0	20	0	0	0
3	0	20	0	0	0
4	0	20	0	0	0
5	0	20	0	0	0
6	0	20	0	0	0
7	20	0	0	0	0
8	20	0	0	0	0
9	20	0	0	0	0
10	20	0	0	0	0

Tab. C-2: *Behavioral Compositions of Fittest HetCNE Evolved Teams in Simple Environments:* For teams of 20 rovers. *HetCNE:* Heterogenous Conventional NeuroEvolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	20	0	0	0
2	0	20	0	0	0
3	0	20	0	0	0
4	0	20	0	0	0
5	0	20	0	0	0
6	0	20	0	0	0
7	0	20	0	0	0
8	0	20	0	0	0
9	0	20	0	0	0
10	0	20	0	0	0

Tab. C-3: *Behavioral Compositions of Fittest CCGA Evolved Teams in Simple Environments:* For teams of 20 rovers. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	2	9	0	9
2	0	3	7	0	10
3	0	2	11	0	7
4	0	1	10	0	9
5	0	3	9	0	8
6	0	2	10	0	8
7	0	5	5	0	10
8	0	3	7	0	10
9	0	4	5	0	11
10	0	4	4	0	12

Tab. C-4: *Behavioral Compositions of Fittest MESP Evolved Teams in Simple Environments:* For teams of 20 rovers. *MESP:* Multi-Agent Enforced Sub-Populations. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rovers. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	5	10	0	5
2	0	4	12	0	4
3	0	4	9	0	7
4	0	3	12	0	5
5	0	4	11	0	5
6	0	3	10	0	7
7	4	2	5	0	9
8	2	4	6	0	8
9	3	4	5	0	8
10	2	6	5	0	7

Tab. C-5: *Behavioral Composition of Fittest CONE Evolved Teams in Simple Environments:* For teams of 20 rovers. *CONE:* Collective Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	7	8	0	5
2	3	6	7	0	4
3	4	5	8	0	3
4	3	5	9	0	3
5	2	4	9	0	5
6	3	5	8	0	4
7	4	4	7	0	5
8	5	4	6	0	5
9	3	5	8	0	4
10	5	4	4	0	7

Tab. C-6: *Fittest HomCNE Evolved Team Compositions in Complex Environments:* For teams of 20 rovers. *HomCNE:* Homogenous Conventional Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	20	0	0	0
2	0	20	0	0	0
3	0	20	0	0	0
4	0	20	0	0	0
5	0	20	0	0	0
6	0	20	0	0	0
7	0	0	20	0	0
8	0	0	20	0	0
9	0	0	20	0	0
10	0	0	20	0	0

Tab. C-7: *Fittest HetCNE Evolved Team Compositions in Complex Environments:* For teams of 20 rovers. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	0	20	0	0
2	0	0	20	0	0
3	0	0	20	0	0
4	0	0	20	0	0
5	0	0	20	0	0
6	0	0	20	0	0
7	0	0	20	0	0
8	0	0	20	0	0
9	0	0	20	0	0
10	0	0	20	0	0

Tab. C-8: *Fittest CCGA Evolved Team Compositions in Complex Environments:* For teams of 20 rovers. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rover. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
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

Tab. C-9: *Fittest MESP Evolved Team Compositions in Complex Environments:* For teams of 20 rovers. *MESP:* Multi-Agent Enforced Sub-Populations. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rovers. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
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

Tab. C-10: *Fittest CONE Evolved Team Compositions in Complex Environments:* For teams of 20 rovers. *CONE:* Collective Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
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

Comparing Task Performance Results of Teams Evolved in Simple and Complex Environments

Data sets of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams presented in figures 5.8, 5.9, 5.10, and 5.11, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.8, the respective P values: [0.60, 0.64, 0.72, 0.83, 0.48] are calculated.
- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.9, the respective P values: [0.67, 0.76, 0.69, 0.70, 0.75] are calculated.
- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.10, the respective P values: [0.55, 0.60, 0.78, 0.78, 0.65] are calculated.
- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.11, the respective P values: [0.74, 0.87, 0.79, 0.67, 0.77] are calculated.

Table C-11 presents the P values for t-tests conducted between the *average red rock value detected* by teams evolved in the *simple* and *complex* environments. Table C-12 presents the P values for t-tests conducted between the *average area covered* by teams evolved in the *simple* and *complex* environments. For both tables C-11 and C-12, a value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

Comparing Teams Evolved in the Complex Environments

Data sets of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams presented in figures 5.10, and 5.11, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- For HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE, red rock value detected data distributions represented in figure 5.10, the respective P values: [0.62, 0.66, 0.82, 0.71, 0.58] are calculated.
- For HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE, area covered data distributions represented in figure 5.11, the respective P values: [0.55, 0.64, 0.68, 0.76, 0.67] are calculated.

Tab. C-11: *Statistical Comparison of Red Rock Value Detected by Teams Evolved in Simple and Complex Environments:* T-test values for HomCNE, HetCNE, CCGA, MESP, and CONE evolved teams. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Simple and Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HomCNE											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HetCNE											
CCGA	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.12*	*0.28*	0.0005	0.0001
CCGA											
MESP	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.42*	*0.83*	0.0005	0.0003
MESP											
CONE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	0.0006	0.0002
CONE											

Tab. C-12: *Statistical Comparison of Area Covered by Teams Evolved in Simple and Complex Environments:* T-test values for HomCNE, HetCNE, CCGA, MESP, and CONE evolved teams. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Simple and Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs*	*0.057*	*0.18*	*0.076*	*0.20*	*0.22*	*0.31*	*0.088*	*0.057*	0.0067	0.0028
HomCNE											
HetCNE	*vs*	*0.095*	*0.12*	*0.065*	*0.28*	*0.12*	*0.16*	*0.098*	*0.097*	*0.13*	*0.19*
HetCNE											
CCGA	*vs*	*0.29*	*0.060*	*0.093*	*0.057*	*0.25*	*0.11*	0.0075	*0.053*	0.011	0.0008
CCGA											
MESP	*vs*	*0.064*	*0.35*	0.024	0.044	*0.22*	*0.19*	0.028	*0.067*	*0.14*	*0.15*
MESP											
CONE	*vs*	*0.28*	*0.14*	*0.45*	*0.076*	0.047	*0.10*	*0.56*	*0.78*	*0.15*	*0.68*
CONE											

Tab. C-13: *Statistical Comparison of Red Rock Value Detected by Teams in Complex Environments:* T-test values for teams evolved by HomCNE, HetCNE, CCGA, MESP, and CONE. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CCGA											
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MESP											
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CCGA											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MESP											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
CCGA	*vs*	*0.65*	*0.30*	*0.37*	*0.34*	*0.29*	*0.33*	*0.38*	*0.45*	*0.082*	*0.30*
MESP											
CCGA	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0003	0.0001	0.0001	0.0001
CONE											
MESP	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											

Table C-13 presents the P values for t-tests conducted between the *average red rock value detected* by teams evolved in the *complex* environment set. Table C-14 presents the P values for t-tests conducted between the *average area covered* by teams evolved in the *complex* environment set. For both tables C-13 and C-14, a value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

Behavioral Compositions of Fittest Teams Evolved in Extended Complex Environments

Tables C-15, C-16 C-17, C-18, and C-19 present the behavioral composition of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in each of the extended complex environments. Behavioral composition refers to the composite number of rovers in each team that are *non-specialized*

Tab. C-14: *Statistical Comparison of Area Covered by Teams in Complex Environments:* T-test values for teams evolved by HomCNE, HetCNE, CCGA, MESP, and CONE. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.84*	0.044	*0.093*	0.035
CCGA											
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.40*	*0.12*	*0.083*	*0.37*
MESP											
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.70*	*0.13*	0.0001	0.0001
CCGA											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.18*	0.0005	*0.15*	0.0009
MESP											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
CCGA	*vs*	*0.35*	*0.69*	*0.55*	*0.59*	*0.54*	*0.40*	*0.17*	*0.31*	*0.93*	0.0053
MESP											
CCGA	*vs*	0.0001	0.0011	0.0001	0.0001	0.0002	0.0019	0.0001	0.0001	0.0001	0.0001
CONE											
MESP	*vs*	0.0001	0.0007	0.0001	0.0001	0.0007	0.03	0.0001	0.0001	0.0001	0.0001
CONE											

Tab. C-15: *Fittest HomCNE Evolved Team Compositions in the Extended Complex Environment Set:* For teams of 20 rovers. *HomCNE:* Homogenous Conventional Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to the low-res, med-res, and hi-res detection behavior, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	20	0	0	0	0
2	0	0	0	0	20
3	0	0	0	0	20
4	0	0	20	0	0
5	20	0	0	0	0
6	0	0	0	0	20
7	0	0	0	0	20
8	0	0	20	0	0
9	20	0	0	0	0
10	0	0	20	0	0

or *specialized* to activating red rock detection sensors with either the *low-res*, *med-res*, or *hi-res* settings. In tables C-15, C-16, C-17, C-18, and C-19, *red rock distribution* (RRD) refers to one of the ten environments within the extended complex environment set, where each environment corresponds to a given red rock distribution (section 5.1.4).

Task Performance of Fittest Teams Evolved in Extended Complex Environments

Tables C-20 and C-21 present the *red rock value detected* and *area covered*, respectively, by the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE in each of the extended complex environments.

Comparing Results of Teams Evolved in Extended Complex Environments

Data sets of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams presented in figures 5.12 and 5.13, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.12, the respective P values: [0.52, 0.49, 0.70, 0.73, 0.68] are calculated.

Tab. C-16: *Fittest HetCNE Evolved Team Compositions in the Extended Complex Environment Set:* For teams of 20 rovers. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to the low-res, med-res, and hi-res detection behavior, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	0	0	0	0	20
2	0	0	0	0	20
3	0	0	0	0	20
4	0	0	0	0	20
5	0	0	0	0	20
6	0	0	0	0	20
7	0	0	0	0	20
8	0	0	0	0	20
9	0	0	0	0	20
10	0	0	0	0	20

Tab. C-17: *Fittest CCGA Evolved Team Compositions in the Extended Complex Environment Set:* For teams of 20 rovers. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to the low-res, med-res, and hi-res detection behavior, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	11	2	0	0	7
2	4	10	0	0	6
3	1	5	5	0	9
4	0	7	6	0	7
5	11	3	0	0	6
6	5	7	0	0	6
7	0	8	2	0	10
8	0	7	7	0	6
9	8	7	0	0	5
10	0	8	6	0	6

Tab. C-18: *Fittest MESP Evolved Team Compositions in Extended Complex Environments:* For teams of 20 rovers. *MESP:* Multi-Agent Enforced Sub-Populations. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection, respectively. *Mover:* Rovers specialized to moving. *Non-Specialized:* Non-specialized rovers. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	12	3	0	0	5
2	7	6	0	0	7
3	0	10	3	0	7
4	0	6	6	0	8
5	10	3	0	0	7
6	2	8	2	0	8
7	4	5	5	0	6
8	0	4	8	0	8
9	10	7	0	0	3
10	0	9	7	0	4

Tab. C-19: *Fittest CONE Evolved Team Compositions in Extended Complex Environments:* For teams of 20 rovers. *CONE:* Collective Neuro-Evolution. *Low-/Med-/Hi-Res Detectors:* Rovers specialized to low-res, med-res, and hi-res detection. *Mover:* Rovers specialized to moving. *Non-Specialized:* Rovers not specialized to any behavior. *RRD:* Red Rock Distribution.

RRD	*Low-Res Detectors*	*Med-Res Detectors*	*Hi-Res Detectors*	*Mover*	*Non-Specialized*
1	10	5	0	0	5
2	8	7	0	0	5
3	0	8	6	0	6
4	0	8	8	0	4
5	8	4	3	0	5
6	6	7	3	0	4
7	6	6	5	0	3
8	4	5	8	0	3
9	11	6	0	0	3
10	0	10	6	0	4

Tab. C-20: *Red Rock Value Detected by Fittest Teams in Extended Complex Environments:* For HomCNE, HetCNE, CCGA, MESP and CONE evolved teams. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution. *RRD:* Red Rock Distribution.

RRD	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	2975	2664	3938	3896	4591
2	2680	2745	3750	3915	4226
3	2891	2770	3875	3875	4331
4	2594	2675	3949	3930	4465
5	2970	2924	3881	3860	4426
6	2789	2814	3775	3782	4529
7	2694	2899	3977	3910	4394
8	2949	2855	4002	4067	4519
9	2814	2969	4061	3958	4433
10	2916	2809	4161	4003	4371

Tab. C-21: *Area Covered by the Fittest Teams Evolved in Extended Complex Environments:* Teams evolved by HomCNE, HetCNE, CCGA, MESP and CONE. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution. *RRD:* Red Rock Distribution.

RRD	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	0.70	0.69	0.79	0.78	0.92
2	0.79	0.74	0.82	0.80	0.87
3	0.77	0.75	0.79	0.81	0.88
4	0.69	0.72	0.82	0.83	0.90
5	0.76	0.73	0.86	0.86	0.93
6	0.79	0.79	0.79	0.82	0.89
7	0.80	0.77	0.81	0.81	0.88
8	0.65	0.66	0.84	0.79	0.90
9	0.68	0.65	0.80	0.82	0.91
10	0.64	0.67	0.82	0.84	0.88

Tab. C-22: *Statistical Comparison of Red Rock Value Detected by Teams Evolved in Extended Complex Environments:* T-test values for teams evolved by HomCNE, HetCNE, CCGA, MESP, and CONE. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Extended Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs* *CCGA*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HomCNE	*vs* *MESP*	0.0001	0.0010	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HomCNE	*vs* *CONE*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HetCNE	*vs* *CCGA*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HetCNE	*vs* *MESP*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HetCNE	*vs* *CONE*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CCGA	*vs* *MESP*	*0.46*	*0.35*	*0.39*	*0.083*	*0.22*	*0.17*	*0.070*	*0.56*	*0.081*	0.021
CCGA	*vs* *CONE*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MESP	*vs* *CONE*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 5.13, the respective P values: [0.65, 0.71, 0.79, 0.67, 0.60] are calculated.

Comparisons for Behavioral Validation

This section presents the results of a set of statistical tests that compare the task performances of non-specialized, low-res, med-res, and hi-res detector teams with task performances yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams. For simplicity, the non-specialized, low-res, med-res, and hi-res detector teams (section 5.5.4) are referred to as *cloned teams*, and teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE are referred to as *evolved teams*. Data sets of the cloned teams are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- P = [0.80, 0.94, 0.64, 0.72, 0.83] is calculated for the red rock value detected result distributions for non-specialized, low-res, med-res, and hi-res detector teams, respectively.

Tab. C-23: *Statistical Comparison of Area Covered by Teams Evolved in Extended Complex Environments:* T-test values for teams evolved by HomCNE, HetCNE, CCGA, MESP, and CONE. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Extended Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
HomCNE	*vs*	0.0001	0.0001	*0.081*	0.0001	0.0001	*0.30*	*0.32*	0.0001	0.0001	0.0001
CCGA											
HomCNE	*vs*	0.0001	0.0001	*0.057*	0.0001	0.0001	0.0001	*0.057*	0.0001	0.0001	0.0001
MESP											
HomCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	*0.40*	*0.45*	0.0001	0.0001	0.0001
CCGA											
HetCNE	*vs*	0.0001	0.010	0.0001	0.0001	0.0001	0.0001	*0.15*	*0.18*	0.0001	0.0001
MESP											
HetCNE	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
CCGA	*vs*	*0.27*	0.0001	*0.40*	0.024	*0.17*	0.0003	*0.14*	0.0001	0.030	*0.67*
MESP											
CCGA	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											
MESP	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE											

- P = [0.66, 0.78, 0.84, 0.82, 0.76] is calculated for the area covered result distributions for non-specialized, low-res, med-res, and hi-res detector teams, respectively.

To determine if there is a statistically significant difference between task performance results of cloned and evolved teams, an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ.

The results of statistical task performance comparisons between cloned and evolved teams are presented in tables C-24 and C-25. A value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

Genotype and Specialization Difference Metric Analysis

This analysis supports the efficacy of the *Genotype Difference Metric* (GDM), and *Specialization Difference Metric* (SDM) for facilitating behavioral specialization, and increasing task performance in CONE evolved rover teams. To support this the CONE method is executed with the following three variations of the original experimental set up for CONE experiments.

1. *CONE without GDM (CONE-1 in tables C-26 and C-27):* Teams are evolved using CONE without the GDM (section 3.1.2), meaning that genotype recombination only occurs *within* sub-populations of a given population. This is also the case for Multi-Agent ESP (section 2.3.3). The SDM for inter-population genotype recombination remains active.

2. *CONE without SDM (CONE-2 in tables C-26 and C-27):* Teams are evolved using CONE without the SDM (section 3.2.2). The GDM remains active.

3. *CONE without GDM and SDM (CONE-3 in tables C-26 and C-27):* Teams are evolved using CONE without both the GDM and SDM.

Figure 5.16 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. Figure 5.17 presents the *average area covered* 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 figures 5.16 and 5.17.

To determine if there is a statistically significant difference between task performance results of HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE

Tab. C-24: *Statistical Comparison of Red Rock Value Detected by Cloned and Evolved Teams in Complex Environments:* HomCNE, HetCNE, CCGA, MESP, or CONE is used to evolve teams. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution. Cloned teams are labeled: *LR:* Low-Res Detector Team. *MR:* Med-Res Detector Team *HR:* Hi-Res Detector Team *NS:* Non Specialized Team.

	Complex Environment									
Team	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
LR vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0026	*0.55*	*0.23*	*0.25*	*0.19*
MR vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HR vs HomCNE	*0.098*	*0.055*	*0.20*	*0.11*	*0.14*	*0.052*	0.0001	0.0001	0.0001	0.0001
HR vs HetCNE	*0.10*	*0.13*	*0.095*	*0.15*	*0.12*	*0.21*	*0.20*	*0.15*	*0.19*	*0.15*
HR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0069	0.0098
HR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0005	0.0010
HR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0012	0.0055
NS vs HomCNE	*0.09*	*0.17*	*0.30*	*0.41*	*0.17*	*0.32*	0.0001	0.0001	0.0001	0.0001
NS vs HetCNE	*0.13*	*0.12*	*0.14*	*0.15*	*0.20*	*0.14*	0.0001	0.0001	0.0001	0.0001
NS vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Tab. C-25: *Statistical Comparison of Area Covered by Cloned and Evolved Teams in Complex Environments:* HomCNE, HetCNE, CCGA, MESP, or CONE is used to evolve teams. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogenous Conventional Neuro-Evolution. *CCGA:* Cooperative Coevolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution. Cloned teams are labeled: *LR:* Low-Res Detector Team. *MR:* Med-Res Detector Team *HR:* Hi-Res Detector Team *NS:* Non Specialized Team.

	Complex Environment									
Team	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
LR vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
LR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.15*	*0.48*	*0.63*	*0.70*
MR vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0012	0.0039
MR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0017	0.0003	0.0045
MR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0007
HR vs HomCNE	*0.12*	*0.38*	*0.13*	*0.12*	*0.10*	*0.1*	0.0001	0.0001	0.0001	0.0001
HR vs HetCNE	*0.18*	*0.23*	*0.24*	*0.11*	*0.19*	*0.30*	*0.27*	*0.29*	*0.28*	*0.37*
HR vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0009	*0.18*	*0.24*	*0.15*	*0.16*
HR vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	*0.21*	*0.32*	*0.35*	*0.25*
HR vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	*0.12*	*0.19*	*0.22*	*0.24*
NS vs HomCNE	*0.41*	*0.55*	*0.50*	*0.62*	*0.47*	*0.65*	0.0001	0.0001	0.0001	0.0001
NS vs HetCNE	*0.23*	*0.29*	*0.23*	*0.35*	*0.30*	*0.32*	0.0001	0.0001	0.0001	0.0001
NS vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

evolved teams, and teams evolved using each variation of the CONE experimental setup (without SDM, GDM or both SDM and GDM), an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ. Data sets of teams evolved using CONE without GDM, CONE without SDM, and CONE without GDM and SDM, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- P = [0.72, 0.64, 0.80, 0.77, 0.78] is calculated for the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions, respectively.

For the comparison of the *average red rock value detected*, the resulting P values are presented in table C-26. For the comparison of the *average area covered*, the resulting P values are presented in table C-27. A value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

The comparison of results presented in figures 5.16 and 5.17 indicate that, for all complex environments, there is a significant difference between the task performance of teams evolved by CONE, and the task performance of teams evolved by CONE-1, CONE-2, and CONE-3. That is, the teams evolved by the CONE method yield a performance advantage over the CONE variants. This result supports the research hypothesis (section 1.2) that both the GDM and SDM are beneficial in terms of increasing task performance in CONE evolved teams. Specifically, without either the GDM or SDM, the CONE evolved teams lose their advantage of a significantly higher task performance. Furthermore, the rover caste lesion study (section 5.5.3) supports the hypothesis that the GDM and SDM enables CONE to derive a degree of behavioral specialization (in this case, a set of complementary and interacting rover castes) that is appropriate for achieving a higher task performance, comparative to related methods.

Tab. C-26: *Statistical Comparison of Red Rock Value Detected by Evolved Teams (by CONE variants) in Complex Environments:* Teams evolved by CONE without GDM, SDM (or both). *CONE:* Team evolved by Multi-Agent Enforced Sub-Populations. *CONE-1:* CONE evolved team (no GDM). *CONE-2:* CONE evolved team (no SDM). *CONE-3:* CONE evolved team (no GDM and SDM).

		Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
CONE-1	*vs*	*0.27*	*0.34*	*0.36*	*0.43*	*0.50*	*0.55*	*0.36*	*0.47*	*0.66*	*0.53*
CONE											
CONE-2	*vs*	*0.17*	*0.25*	*0.22*	*0.32*	*0.46*	*0.35*	*0.29*	*0.11*	*0.16*	*0.23*
CONE											
CONE-3	*vs*	*0.20*	*0.15*	*0.12*	*0.24*	*0.52*	*0.42*	*0.33*	*0.29*	*0.24*	*0.19*
CONE											

Tab. C-27: *Statistical Comparison of Area Covered by Evolved Teams (by CONE variants) in Complex Environments:* Teams evolved by CONE without GDM, SDM (or both). *CONE:* Team evolved by Multi-Agent Enforced Sub-Populations. *CONE-1:* CONE evolved team (no GDM). *CONE-2:* CONE evolved team (no SDM). *CONE-3:* CONE evolved team (no GDM and SDM).

		Complex Environment									
Team		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
CONE-1	*vs*	*0.37*	*0.41*	*0.52*	*0.63*	*0.45*	*0.42*	*0.43*	*0.49*	*0.57*	*0.46*
CONE											
CONE-2	*vs*	*0.26*	*0.15*	*0.12*	*0.15*	*0.16*	*0.50*	*0.32*	*0.40*	*0.36*	*0.55*
CONE											
CONE-3	*vs*	*0.28*	*0.25*	*0.42*	*0.36*	*0.48*	*0.34*	*0.47*	*0.53*	*0.44*	*0.58*
CONE											

APPENDIX D: STATISTICAL COMPARISONS IN THE GACC TASK

This appendix presents statistical comparisons of task performance results, an analysis of these results, and of the behavioral compositions of evolved teams.

Behavioral Compositions of Evolved Teams

Tables D-1, D-2 D-3, D-4, and D-5 present, for each simple GACC environment, the behavioral composition of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams, respectively. Behavioral composition refers to the composite number of robots in a team that are *non-specialized* or *specialized* to activating object detection sensors or the gripper gripper with a given setting (section 6.1.1). Tables D-6, D-7 D-8, D-9, and D-10 present, for each complex GACC environment, the behavioral composition of the fittest HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams, respectively. Behavioral composition refers to the composite number of robots in a team that are *non-specialized* or *specialized* to activating its object detection sensors or gripper (section 6.1.1).

Task Performances of Evolved Teams

An analysis of emergent team behaviors requires a statistical comparison of task performance results (*average number of atomic objects delivered in correct order*) yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams. The task performance of the fittest teams evolved in simple and complex environment sets are presented in table D-11 and table D-13, respectively. TableD-12 and table D-14 present the percentage of optimal task performance achieved by the fittest teams evolved in the simple and complex environments, respectively.

Comparing Task Performances of Teams Evolved in Simple versus Complex Environments

A statistical comparison of task performance results yielded by teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE in the *simple* and *complex* environment sets, is conducted. A comparison is conducted between the *average number of objects delivered in correct order* by teams evolved

Tab. D-1: *Behavioral Compositions of Fittest Homogenous CNE Evolved Teams in Simple Environments:* For teams of 30 robots evolved by HomCNE in each of the simple environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Simple Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	0	0	0	0	0	0	0
O-B Detector	0	0	0	0	0	0	0	0	0
O-C Detector	0	0	0	0	0	0	0	0	0
O-A Gatherer	0	0	0	0	0	0	0	0	0
O-B Gatherer	0	0	0	0	0	0	0	0	0
O-C Gatherer	0	0	0	0	0	0	0	0	0
O-A Constructor	0	0	0	0	0	0	0	0	0
O-B Constructor	0	0	0	0	0	0	0	0	0
O-C Constructor	0	0	0	0	0	0	0	0	0
Non-Specialized	30	30	30	30	30	30	30	30	30

Tab. D-2: *Behavioral Compositions of Fittest Heterogenous CNE Evolved Teams in Simple Environments:* For teams of 30 robots evolved by HetCNE in each of the simple environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Simple Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	0	0	0	0	0	0	0
O-B Detector	0	0	0	0	0	0	0	0	0
O-C Detector	0	0	0	0	0	0	0	0	0
O-A Gatherer	0	0	0	0	0	0	0	0	0
O-B Gatherer	0	0	0	0	0	0	0	0	0
O-C Gatherer	0	0	0	0	0	0	0	0	0
O-A Constructor	0	0	0	0	0	0	0	0	0
O-B Constructor	0	0	0	0	0	0	0	0	0
O-C Constructor	0	0	0	0	0	0	0	0	0
Non-Specialized	30	30	30	30	30	30	30	30	30

Tab. D-3: *Behavioral Compositions of Fittest CCGA Evolved Teams in Simple Environments:* For teams of 30 robots evolved by CCGA in each of the simple environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Simple Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	9	0	8	10	0	0	9	0	0
O-B Detector	0	8	0	0	8	0	0	8	0
O-C Detector	0	0	0	0	0	8	0	0	10
O-A Gatherer	14	0	0	9	0	0	13	0	0
O-B Gatherer	0	12	10	0	12	0	0	12	0
O-C Gatherer	0	0	0	0	0	11	0	0	9
O-A Constructor	4	0	5	6	0	0	4	0	0
O-B Constructor	0	5	0	0	4	0	0	3	0
O-C Constructor	0	0	0	0	0	5	0	0	4
Non-Specialized	3	5	7	5	6	6	4	5	7

Tab. D-4: *Behavioral Compositions of Fittest MESP Evolved Teams in Simple Environments:* For teams of 30 robots evolved by MESP in each of the simple environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Simple Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	7	0	8	11	0	0	10	0	0
O-B Detector	0	9	0	0	9	0	0	10	0
O-C Detector	0	0	0	0	0	7	0	0	8
O-A Gatherer	13	0	0	10	0	0	11	0	0
O-B Gatherer	0	11	10	0	9	0	0	10	0
O-C Gatherer	0	0	0	0	0	10	0	0	12
O-A Constructor	5	0	6	4	0	0	5	0	0
O-B Constructor	0	4	0	0	5	0	0	4	0
O-C Constructor	0	0	0	0	0	5	0	0	5
Non-Specialized	5	6	6	5	7	8	4	6	5

Tab. D-5: *Behavioral Compositions of Fittest CONE Evolved Teams in Simple Environments:* For teams of 30 robots evolved by CONE in each of the simple environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Simple Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	9	0	9	10	0	0	10	0	0
O-B Detector	0	11	0	0	11	0	0	9	0
O-C Detector	0	0	0	0	0	10	0	0	7
O-A Gatherer	11	0	0	10	0	0	9	0	0
O-B Gatherer	0	8	9	0	11	0	0	11	0
O-C Gatherer	0	0	0	0	0	11	0	0	10
O-A Constructor	4	0	7	6	0	0	5	0	0
O-B Constructor	0	5	0	0	3	0	0	6	0
O-C Constructor	0	0	0	0	0	6	0	0	5
Non-Specialized	6	6	5	4	5	3	6	4	8

Tab. D-6: *Behavioral Compositions of Fittest Homogenous CNE Evolved Teams in Complex Environments:* For teams of 30 robots evolved by HomCNE in each of the complex environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Complex Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	0	0	0	0	0	0	0
O-B Detector	0	0	0	0	0	0	0	0	0
O-C Detector	0	0	0	0	0	0	0	0	0
O-A Gatherer	0	0	0	0	0	0	0	0	0
O-B Gatherer	0	0	0	0	0	0	0	0	0
O-C Gatherer	0	0	0	0	0	0	0	0	0
O-A Constructor	0	0	0	0	0	0	0	0	0
O-B Constructor	0	0	0	0	0	0	0	0	0
O-C Constructor	0	0	0	0	0	0	0	0	0
Non-Specialized	30	30	30	30	30	30	30	30	30

Tab. D-7: *Behavioral Compositions of Fittest Heterogenous CNE Evolved Teams in Complex Environments:* For teams of 30 robots evolved by HetCNE in each of the complex environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Complex Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	0	0	0	0	0	0	0
O-B Detector	0	0	0	0	0	0	0	0	0
O-C Detector	0	0	0	0	0	0	0	0	0
O-A Gatherer	0	0	0	0	0	0	0	0	0
O-B Gatherer	0	0	0	0	0	0	0	0	0
O-C Gatherer	0	0	0	0	0	0	0	0	0
O-A Constructor	0	0	0	0	0	0	0	0	0
O-B Constructor	0	0	0	0	0	0	0	0	0
O-C Constructor	0	0	0	0	0	0	0	0	0
Non-Specialized	30	30	30	30	30	30	30	30	30

Tab. D-8: *Behavioral Compositions of Fittest CCGA Evolved Teams in Complex Environments:* For teams of 30 robots evolved by CCGA in each of the complex environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Complex Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	1	4	5	4	4	5	4
O-B Detector	0	3	6	0	1	5	6	3	5
O-C Detector	8	3	0	5	3	1	0	2	1
O-A Gatherer	0	0	5	4	5	5	5	6	5
O-B Gatherer	0	6	6	0	1	6	7	5	7
O-C Gatherer	12	5	0	5	4	1	0	3	1
O-A Constructor	0	0	1	2	3	2	2	2	2
O-B Constructor	0	2	2	0	1	3	2	2	2
O-C Constructor	2	1	0	3	2	0	0	1	0
Non-Specialized	8	10	9	7	6	3	4	1	3

Tab. D-9: *Behavioral Compositions of Fittest MESP Evolved Teams in Complex Environments:* For teams of 30 robots evolved by MESP in each of the complex environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

Complex Environment									
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	2	3	4	4	5	6	5
O-B Detector	0	3	7	0	1	5	5	4	5
O-C Detector	9	4	0	4	4	1	0	1	2
O-A Gatherer	0	0	6	4	5	6	5	6	5
O-B Gatherer	0	5	7	0	1	4	6	6	6
O-C Gatherer	10	7	0	6	5	1	0	2	2
O-A Constructor	0	0	1	3	3	2	2	2	2
O-B Constructor	0	2	1	0	2	3	3	1	2
O-C Constructor	2	1	0	3	1	0	0	1	1
Non-Specialized	9	8	6	7	4	4	4	1	0

Tab. D-10: *Behavioral Compositions of Fittest CONE Evolved Teams in Complex Environments:* For teams of 30 robots evolved by CONE in each of the complex environments. *O-A/O-B/O-C Detector:* Robots specialized to detecting type A, B, and C objects, respectively. *O-A/O-B/O-C Gatherer:* Robots specialized to moving whilst gripping type A, B, and C objects, respectively. *O-A/O-B/O-C Constructor:* Robots specialized to gripping and placing type A, B, and C objects in the construction zone, respectively. *Non-Specialized:* Robots not specialized to any behavior.

	Complex Environment Number								
Specialization	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
O-A Detector	0	0	3	4	5	4	5	5	6
O-B Detector	0	4	6	0	2	3	4	5	5
O-C Detector	8	5	0	4	4	2	0	1	3
O-A Gatherer	0	0	5	5	4	6	6	4	6
O-B Gatherer	0	5	6	0	2	5	6	5	5
O-C Gatherer	8	5	0	7	4	2	0	3	2
O-A Constructor	0	0	1	2	2	2	2	2	1
O-B Constructor	0	2	1	0	2	1	2	2	1
O-C Constructor	3	2	0	2	2	0	0	2	1
Non-Specialized	11	7	8	6	3	5	5	1	0

Tab. D-11: *Number of Atomic Objects Delivered in Correct Order by Fittest Teams Evolved in Simple Environments:* For teams evolved by HomCNE, HetCNE, CCGA, MESP and CONE. *ENV:* Environment Number.

ENV	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	3	4	4	5	5
2	4	2	5	4	4
3	4	5	4	6	6
4	14	13	14	12	14
5	12	11	15	13	15
6	14	13	13	15	14
7	18	17	19	18	20
8	19	20	18	21	18
9	16	19	15	19	17

Tab. D-12: *Percentage of Optimal Performance Achieved by Fittest Teams Evolved in Simple Environments:* For teams evolved by HomCNE, HetCNE, CCGA, MESP and CONE. *ENV:* Environment Number.

ENV	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	0.3	0.4	0.4	0.5	0.5
2	0.4	0.2	0.5	0.4	0.4
3	0.4	0.5	0.4	0.6	0.6
4	0.7	0.65	0.7	0.6	0.7
5	0.6	0.55	0.75	0.65	0.75
6	0.46	0.43	0.43	0.5	0.47
7	0.6	0.56	0.63	0.6	0.67
8	0.47	0.5	0.45	0.52	0.45
9	0.4	0.47	0.37	0.47	0.42
AVG	0.48	0.47	0.51	0.54	0.55

Tab. D-13: *Number of Atomic Objects Delivered in Correct Order by Fittest Teams Evolved in Complex Environments:* For teams evolved by HomCNE, HetCNE, CCGA, MESP and CONE. *ENV:* Environment Number.

ENV	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	3	3	9	10	10
2	3	4	10	10	10
3	4	3	8	9	10
4	8	9	13	12	20
5	7	8	12	15	20
6	9	10	16	21	29
7	8	7	19	22	28
8	8	9	27	29	38
9	7	10	26	31	37

Tab. D-14: *Percentage of Optimal Performance Achieved by Fittest Teams Evolved in Complex Environments:* For teams evolved by HomCNE, HetCNE, CCGA, MESP and CONE. *ENV:* Environment Number. *AVG:* Average value calculated for all environments.

ENV	*HomCNE*	*HetCNE*	*CCGA*	*MESP*	*CONE*
1	0.3	0.3	0.9	1.0	1.0
2	0.3	0.4	1.0	1.0	1.0
3	0.4	0.3	0.8	0.9	1.0
4	0.4	0.45	0.65	0.6	1.0
5	0.35	0.4	0.6	0.75	1.0
6	0.3	0.33	0.53	0.7	0.96
7	0.26	0.23	0.63	0.73	0.93
8	0.2	0.22	0.67	0.725	0.95
9	0.17	0.25	0.65	0.775	0.925
AVG	0.29	0.32	0.71	0.79	0.93

Tab. D-15: *Statistical Comparison of Number of Objects Delivered in Correct Order (Complex Objects Constructed) by Teams in Simple and Complex Environments:* T-test values for HomCNE, HetCNE, CCGA, MESP, and CONE evolved teams. *CNE:* Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution.

		Simple and Complex Environment								
Method		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
HomCNE *HomCNE*	*vs*	*0.47*	*0.48*	*0.27*	*0.39*	*0.44*	*0.35*	*0.48*	*0.25*	*0.16*
HetCNE *HetCNE*	*vs*	*0.30*	*0.26*	*0.22*	*0.18*	*0.21*	*0.11*	*0.37*	*0.41*	*0.19*
CCGA *CCGA*	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
MESP *MESP*	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0003	0.0001
CONE *CONE*	*vs*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

with respective methods in the *simple* (figure 6.7) and *complex* (figure 6.8) environment sets. The respective methods are HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE. Data sets representing results of the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE evolved teams presented in figures 6.7, and 6.8, are found to conform to normal distributions via applying the Kolmogorov-Smirnov test.

- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 6.7, the respective P values: [0.62, 0.70, 0.66, 0.63] are calculated.
- For the HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE data distributions represented in figure 6.8, the respective P values: [0.61, 0.65, 0.69, 0.62, 0.60] are calculated.

To determine if there is a statistically significant difference between task performance results of teams evolved by respective methods, an independent t-test is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ. Table D-15 presents the P values for t-tests conducted between average task performances yielded by teams evolved in the *simple* and *complex* environments. For table D-15, a value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

Tab. D-16: *Statistical Comparison of Number of Atomic Objects Delivered in Correct Order by Teams Evolved in Complex Environments:* Results of teams evolved by Collective Neuro-Evolution (CONE) versus other methods in the complex environments. *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogeneous Conventional Neuro-Evolution. *CCGA:* Team evolved by Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Team evolved by Multi-Agent Enforced Sub-Populations.

	Complex Environment Number								
Team	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
CONE vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE vs CCGA	*0.18*	*0.10*	*0.072*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CONE vs MESP	*0.23*	*0.14*	*0.098*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Comparisons for Behavioral Validation

This section presents the results of statistical tests that compare the task performances of teams constructed entirely from non-specialized or specialized controllers, with task performances yielded by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams (section 6.4.2). For clarity in describing comparisons, the non-specialized, object-A, object-B, and object-C detector and gatherer teams (section 6.5.5) are referred to as *cloned GACC teams*, and teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE (section 6.4.2) are referred to as *evolved GACC teams*. The task performances attained by each of the cloned teams are detailed in the following.

- *Object-A Detector Teams:* The task performance achieved by object-A detector teams is presented in figure D-1.
- *Object-B Detector Teams:* The task performance achieved by object-B detector teams is presented in figure D-1.
- *Object-C Detector Teams:* The task performance achieved by object-C detector teams is presented in figure D-1.
- *Object-A Gatherer Teams:* The task performance achieved by object-A gatherer teams is presented in figure D-2.
- *Object-B Gatherer Teams:* The task performance achieved by object-B gatherer teams is presented in figure D-2.
- *Object-C Gatherer Teams:* The task performance achieved by object-C gatherer teams is presented in figure D-2.

- *Object-A Constructor Teams:* The task performance achieved by object-A gatherer teams is presented in figure D-3.
- *Object-B Constructor Teams:* The task performance achieved by object-B gatherer teams is presented in figure D-3.
- *Object-C Constructor Teams:* The task performance achieved by object-C gatherer teams is presented in figure D-3.
- *Non-Specialized Teams:* For the purposes of comparison with homogeneous teams of detectors, gatherers and constructors, the task performance of non-specialized teams is presented in figures D-1, D-2, and D-3.

Data sets representing results of the cloned teams are found to conform to normal distributions via applying the Kolmogorov-Smirnov test.

- For non-specialized teams, P = [0.85] is calculated for the task performance result distributions.
- For object-A, object-B, object-C detector teams, P = [0.66, 0.75, 0.58] is calculated for the task performance result distributions, respectively.
- For object-A, object-B, object-C gatherer teams, P = [0.77, 0.69, 0.68] is calculated for the task performance result distributions, respectively.
- For object-A, object-B, object-C constructor teams, P = [0.80, 0.59, 0.57] is calculated for the task performance result distributions, respectively.

To determine if there is a statistically significant difference between task performance results of cloned and evolved teams, an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ. The results of statistical task performance comparisons between cloned and evolved GACC teams are presented in table D-17. A value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results.

Caste Lesion Study

In order to investigate the role of emergent behavioral specialization in teams evolved by HomCNE, HetCNE, CCGA, MESP, and CONE, a *caste lesion study* is conducted. The caste lesion study operates via removing sets of controllers specialized to a given behavior or non-specialized controllers. The behaviors performed by these controllers are then replaced with behaviors yielded by hard-wired heuristic controllers. The task performance of the *lesioned* team is then

Tab. D-17: *Statistical Comparison of Objects delivered in Correct Order by Cloned/Evolved Teams in Complex Environments.* Evolved teams: *HomCNE:* Homogenous Conventional Neuro-Evolution. *HetCNE:* Heterogeneous Conventional Neuro-Evolution. *CCGA:* Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Multi-Agent Enforced Sub-Populations. *CONE:* Collective Neuro-Evolution. Cloned teams: *AD:* Object-A Detector Team. *BD:* Object-B Detector Team. *CD:* Object-C Detector Team. *AG:* Object-A Gatherer Team. *BG:* Object-B Gatherer Team. *CG:* Object-C Gatherer Team. *AC:* Object-A Constructor Team. *BC:* Object-B Constructor Team. *CC:* Object-C Constructor Team. *NS:* Non-Specialized Team.

	Complex Environment Number								
Team	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
AD vs HomCNE	0.0001	0.0047	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AD vs HetCNE	0.0001	0.0001	0.0001	0.0011	0.0001	0.0001	0.0001	0.0001	0.0001
AD vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AD vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AD vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BD vs HomCNE	0.0001	0.0003	0.0001	0.0001	0.0001	0.0026	0.0001	0.0001	0.0001
BD vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0015	0.0001	0.0001	0.0001
BD vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BD vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BD vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CD vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CD vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CD vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CD vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0025
CD vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AG vs HomCNE	0.0001	0.0043	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AG vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AG vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AG vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AG vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BG vs HomCNE	0.0001	0.0001	0.0067	0.0001	0.0001	0.0002	0.0001	0.0001	0.0001
BG vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BG vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BG vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BG vs CONE	0.0006	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CG vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CG vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CG vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
CG vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CG vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AC vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AC vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AC vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AC vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AC vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BC vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0026	0.0001	0.0001	0.0001
BC vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0026	0.0001	0.0001	0.0003
BC vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BC vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
BC vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CC vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CC vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CC vs CCGA	0.0001	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0069
CC vs MESP	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0005
CC vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0012
NS vs HomCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs HetCNE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs CCGA	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs MESP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
NS vs CONE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

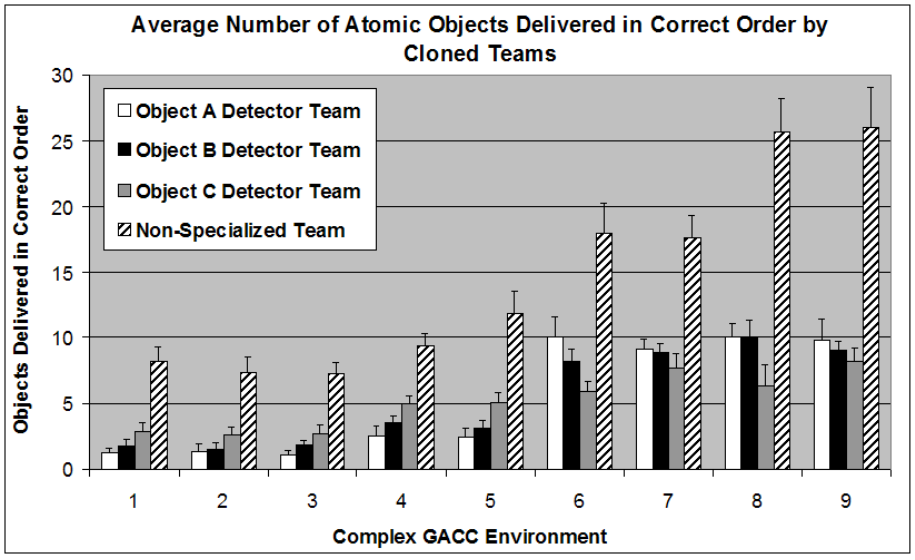

Fig. D-1: Average Number of Objects Delivered in Correct Order by Teams of Clones in the Complex Environment Set. Teams are constructed that consist entirely of an (evolved) non-specialized or specialized controller. Here, the specialized controller is a specialized detector behavior.

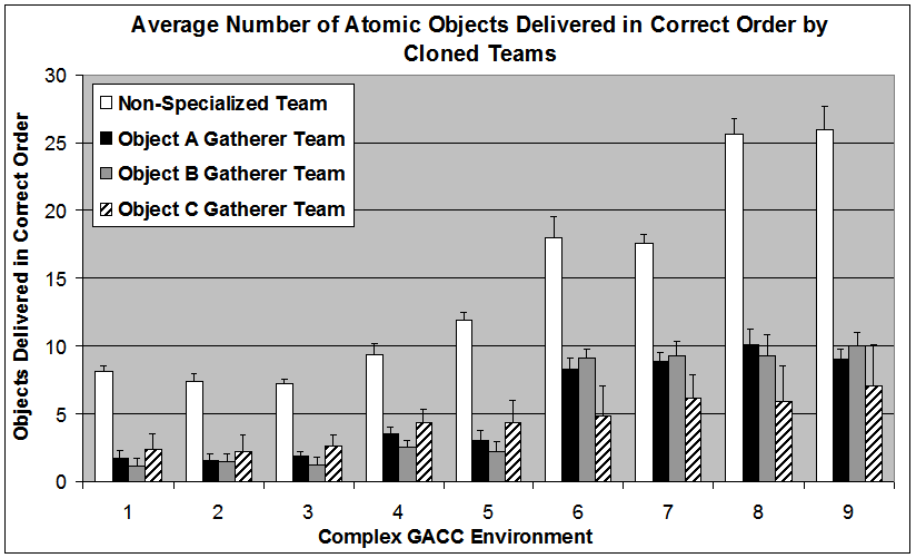

Fig. D-2: Average Number of Objects Delivered in Correct Order by Teams of Clones in the Complex Environment Set. Teams are constructed that consist entirely of an (evolved) non-specialized or specialized controller. Here, the specialized controller is a specialized gatherer behavior.

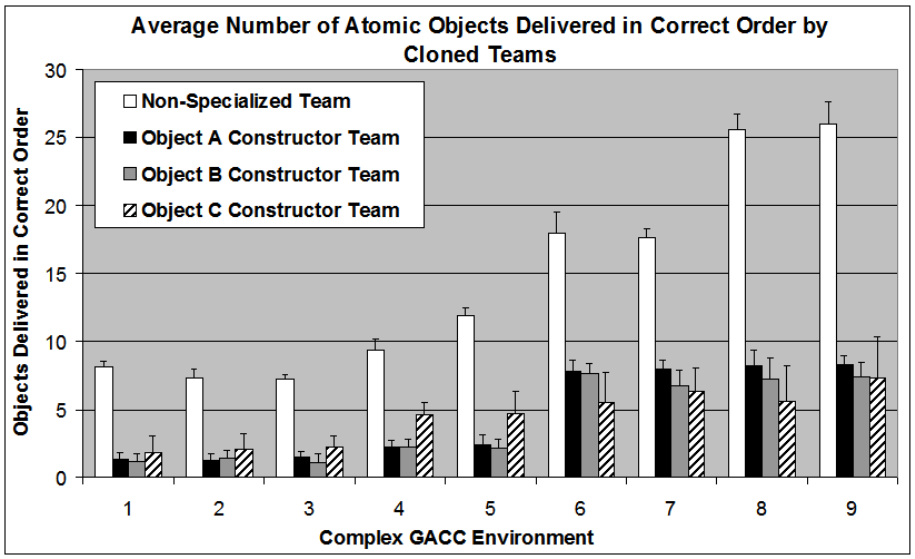

Fig. D-3: Average Number of Objects Delivered in Correct Order by Teams of Clones in the Complex Environment Set. Teams are constructed that consist entirely of an (evolved) non-specialized or specialized controller. Here, the specialized controller is a specialized construction behavior.

re-evaluated. The goal of the caste lesion study is to ascertain the contribution of specialized and non-specialized castes to the overall task performance of the fittest teams. For each of the fittest teams evolved by HomCNE, HetCNE, CCGA, Multi-Agent ESP, and CONE the specialized and non-specialized castes are systematically removed and replaced with heuristic controllers that implement a *hard-wired* specialized or non-specialized behavior (section 6.2.8). Each lesioned team is then executed in 20 new experimental runs for each complex environment, and an average task performance is calculated for each run.

Procedure for lesioning fittest teams: The following procedure is used for each of the fittest teams evolved by each NE method in the complex environments. 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. Thus castes within a fittest team are often re-evaluated in a subset of the complex environment set.

- *Fittest Teams Evolved by HomCNE:* In environments [1, 9] (table D-6) all 30 *non-specialized* robots are replaced with 30 robots using *non-specialized* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 9].

- *Fittest Teams Evolved by HetCNE:* In environments [1, 9] (table D-7) all 30 *non-specialized* robots are replaced with 30 robots using *non-specialized* heuristic controllers (table 6.3). These lesioned teams are then executed

in environments [1, 9].

- *Fittest Teams Evolved by CCGA:*
  - In environments [3, 9] (table D-8), the *O-A Detector* caste is replaced with robots using *Object A Detector* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 9].
  - In environments [2, 3, 5, 6, 7, 8, 9] (table D-8), the *O-B Detector* caste is replaced with robots using *Object B Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
  - In environments [1, 2, 4, 5, 6, 8, 9] (table D-8), the *O-C Detector* caste is replaced with robots using *Object C Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
  - In environments [3, 4, 5, 6, 7, 8, 9] (table D-8), the *O-A Gatherer* caste is replaced with robots using *Object A Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [3, 4, 5, 6, 7, 8, 9].
  - In environments [2, 3, 5, 6, 7, 8, 9] (table D-8), the *O-B Gatherer* caste is replaced with robots using *Object B Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [3, 4, 5, 6, 7, 8, 9].
  - In environments [1, 2, 4, 5, 6, 8, 9] (table D-8), the *O-C Gatherer* caste is replaced with robots using *Object C Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
  - In environments [3, 9] (table D-8), the *O-A Constructor* caste is replaced with robots using *Object A Constructor* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 9].
  - In environments [2, 3, 5, 6, 7, 8, 9] (table D-8), the *O-B Constructor* caste is replaced with robots using *Object B Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
  - In environments [1, 2, 4, 5, 8, 9] (table D-8), the *O-C Constructor* caste is replaced with robots using *Object C Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 8, 9].
  - In environments [1, 9], the *non-specialized* caste is replaced with robots using *non-specialized* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 9].
- *Fittest Teams Evolved by Multi-Agent ESP:*

- In environments [3, 9] (table D-9) the *O-A Detector* caste is replaced with robots using *Object A Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [3, 9].
- In environments [2, 3, 5, 6, 7, 8, 9] (table D-9) the *O-B Detector* caste is replaced with robots using *Object B Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 6, 8, 9] (table D-9) the *O-C Detector* caste is replaced with robots using *Object C Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
- In environments [3, 9] (table D-9) the *O-A Gatherer* caste is replaced with robots using *Object A Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [3, 9].
- In environments [2, 3, 5, 6, 7, 8, 9] (table D-9) the *O-B Gatherer* caste is replaced with robots using *Object B Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 6, 8, 9] (table D-9) the *O-C Gatherer* caste is replaced with robots using *Object C Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
- In environments [3, 9] (table D-9) the *O-A Constructor* caste is replaced with robots using *Object A Constructor* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 9].
- In environments [2, 3, 5, 6, 7, 8, 9] (table D-9) the *O-B Constructor* caste is replaced with robots using *Object B Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 8, 9] (table D-9) the *O-C Constructor* caste is replaced with robots using *Object C Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 8, 9].
- For environments [1, 8], the *non-specialized* caste is replaced with robots using *non-specialized* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 8].

- *Fittest Teams Evolved by CONE:*
  - In environments [3, 8] (table D-10), the *O-A Detector* caste is replaced with robots using *Object A Detector* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 8].

- In environments [2, 3, 5, 6, 7, 8, 9] (table D-10), the *O-B Detector* caste is replaced with robots using *Object B Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 6, 8, 9] (table D-10), the *O-C Detector* caste is replaced with robots using *Object C Detector* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
- In environments [3, 9] (table D-10), the *O-A Gatherer* caste is replaced with robots using *Object A Gatherer* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 9].
- In environments [2, 3, 5, 6, 7, 8, 9] (table D-10), the *O-B Gatherer* caste is replaced with robots using *Object B Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 6, 8, 9] (table D-10), the *O-C Gatherer* caste is replaced with robots using *Object C Gatherer* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 6, 8, 9].
- In environments [3, 9] (table D-10), the *O-A Constructor* caste is replaced with robots using *Object A Constructor* heuristic controllers (table 6.3). Lesioned teams are then executed in environments [3, 9].
- In environments [2, 3, 5, 6, 7, 8, 9] (table D-10), the *O-B Constructor* caste is replaced with robots using *Object B Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [2, 3, 5, 6, 7, 8, 9].
- In environments [1, 2, 4, 5, 8, 9] (table D-10), the *O-C Constructor* caste is replaced with robots using *Object C Constructor* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 2, 4, 5, 8, 9].
- In environments [1, 8], the evolved *non-specialized* caste is replaced with robots using *non-specialized* heuristic controllers (table 6.3). These lesioned teams are then executed in environments [1, 8].

The Role of Difference Metrics in CONE

This section examines the role of the *Genotype Difference Metric* (GDM), and *Specialization Difference Metric* (SDM) for facilitating behavioral specialization, and increasing task performance in CONE evolved teams. As part of this analysis, CONE is executed with the following three experimental variations.

1. *CONE without GDM (CONE-1 in table D-18):* Teams are evolved using CONE without the GDM, so genotype recombination only occurs *within*

Tab. D-18: Statistical Comparison of Number of Atomic Objects Delivered in Correct Order by Teams Evolved (by CONE variants) in Complex Environments: Results of teams evolved by Collective Neuro-Evolution (CONE) without the GDM (Genotype Difference Metric), SDM (Specialization Difference Metric), GDM and SDM. *CCGA:* Team evolved by Cooperative Co-evolutionary Genetic Algorithm. *MESP:* Team evolved by Multi-Agent Enforced Sub-Populations. *CONE-1:* Team evolved by CONE without GDM. *CONE-2:* Team evolved by CONE without SDM. *CONE-3:* Team evolved by CONE without GDM and SDM.

	Complex Environment Number								
Team	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
CONE-1 vs CONE	*0.27*	*0.24*	*0.32*	*0.23*	*0.29*	*0.15*	*0.26*	*0.17*	*0.22*
CONE-2 vs CONE	*0.17*	*0.25*	*0.22*	*0.22*	*0.26*	*0.068*	*0.10*	*0.10*	*0.21*
CONE-3 vs CONE	*0.098*	*0.15*	*0.12*	*0.24*	*0.22*	*0.072*	*0.093*	*0.12*	*0.26*

sub-populations of a given population. The SDM for inter-population genotype recombination remains active.

2. *CONE without SDM (CONE-2 in table D-18):* Teams are evolved using CONE without the SDM. The GDM remains active.

3. *CONE without GDM and SDM (CONE-3 in table D-18):* Teams are evolved using CONE without both the GDM and SDM.

Figure 6.9 presents the average task performance of 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 figure 6.9. To determine if there is a statistically significant difference between task performance results of HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE evolved teams, and teams evolved using each variation of the CONE experimental setup (without SDM, GDM or both SDM and GDM), an independent t-test [48] is applied. The threshold for statistical significance is 0.05, and the null hypothesis is that data sets do not significantly differ. Data sets representing results of teams evolved using CONE-1, CONE-2, and CONE-3, were found to conform to normal distributions via applying the Kolmogorov-Smirnov test [48].

- For the HomCNE, HetCNE, CCGA, ESP, and CONE data distributions, P = [0.80, 0.77, 0.78] is calculated, respectively.

The P values resulting from a statistical comparison of the average task performances are presented in table D-18. A value of 0.0001 indicates that a value of less than 0.0001 is calculated by the t-test. A value typed in italics indicates

that the null hypothesis is rejected for the given t-test, and that there is no significant difference between the given task performance results. Values not in italics indicate that the null hypothesis is accepted and that there is a significant difference between the given task performance results. The comparison of results indicates that, for all complex environments, the task performance of teams evolved by the variations of CONE (CONE-1: without the GDM, CONE-2: SDM, and CONE-3: without both GDM and SDM), is significant lower comparative to the task performance yielded by CONE (original setup) evolved teams. This result indicates that both the GDM and SDM are beneficial for the purpose of increasing the task performance of teams evolved by CONE in each of the complex environments.

APPENDIX E: EXPERIMENTS AND NEURO-EVOLUTION PARAMETERS

This appendix describes parameters specific to the *Collective Neuro-Evolution* (CONE) method, and the simulation environments used to implement the pursuit-evasion, multi-rover, and gathering and collective construction case studies. The task environments and methods compared in this dissertation were implemented using the open-source *Neuro-Evolution Simulation Toolkit* (NEST) based on the platform-independent Java programming language. NEST is largely based upon the *Multi-Agent Simulation* (MASON) toolkit [92], and includes a 2D physics engine for collision handling. Source code, libraries, and documentation used for the experiments can be found at:

http://gforge.cs.vu.nl/projects/nest/

Computer systems used for the experiments were the 72-node Intel Pentium-III Distributed ASCI Supercomputer 2 (DAS-2) cluster at the Vrije Universiteit's Computer Science department, the 20 node 2.0 Gigahertz AMD NEWTIES cluster at the Vrije Universiteit's Computer Science department, and the 30 node 3.2 Gigahertz AMD Hydra2 cluster at George Mason University's Computer Science department. Each node used 2 Gigabytes of RAM.

The parameter settings used depended upon the collective behavior case study. CONE parameter values specific to each case study are described in chapters 4, 5 and 6. For each collective behavior case study, appropriate parameter values were found experimentally, and via extensive experimentation, moderate changes to these parameter values were found to yield minimal variations in collective behavior task performance.

1. *Iterations per epoch (lifetime):* The number of *simulation iterations* that each epoch consists of. At the end of a lifetime for a given agent, the agent begins a new epoch.

2. *Crossover (Recombination):* For consistency, single-point crossover [43] is applied in all experiments.

3. *Epochs:* The number of *task scenarios* that each generation consists of. Each epoch tests different agent and environment parameter values, such as different agent starting positions, orientations, and distributions of environmental resources.

4. *Fitness Stagnation V:* If the fitness of at least one of the fittest n controllers has not progressed in V generations, then adapt the the *Special-*

ization Similarity Threshold (SST) value. Section 3.5 presents a complete description of the method used to adapt the SST value.

5. *Fitness Stagnation W:* If the fitness of at least one of the fittest n controllers has not progressed in $V + W$ generations, then adapt the the *Genetic Similarity Threshold* (GST) value. Section 3.5 presents a complete description of the method used to adapt the GST value.

6. *Fitness Stagnation Y:* If fitness of at least one of the n controllers has not progressed in $V + W + Y$ generations, then the number of sub-populations (controller size) for each of the controllers with stagnating fitness, is adapted. Section 3.6 comprehensively describes the method used to adapt controller size.

7. *Generations:* Number of generations of a neuro-evolution process.

8. *Genotype Distance (GD) Value:* A normalized value within the range [0.0, 1.0], that corresponds to the *Genetic Distance* between two genotypes g_a and g_b. Calculation of the GD value is described in section 3.1.2.

9. *Genotype Representation:* The encoding of any given ANN controller. Each genotype is represented as a floating point value vector and is direct one-to-one encoding of an ANN controller. That is, the value of each sensory input weight connecting each input neuron to each hidden layer neuron, plus the value of each motor output weight connecting each output neuron to each hidden layer neuron. Within each genotype, each gene represents a single connection weight.

10. *Hidden Layer Neurons:* The number of hidden layer neurons assigned to any given ANN controller, and thus the number of *sub-populations* within a genotype population, from which the ANN controller is derived. The number of hidden layer neurons is determined by the experimenter *a priori*, where the appropriate number is task dependent, and is determined via a set of exploratory experiments. The number of hidden layer neurons (and hence the number of sub-populations) is adapted over the course of the CONE evolutionary process as a function of fitness progress.

11. *Input Neurons:* The number of sensory input neurons assigned to any given ANN controller. The number of sensory input neurons is determined by the experimenter *a priori*, where the appropriate number is task dependent. The number of sensory input neurons remains static for any given NE experiment, and determines genotype length.

12. *Mutation probability:* The degree of probability *per gene* in a given genotype, that the gene is mutated within a given range. Within any given genotype, each gene represents a ANN controller connection weight.

13. *Mutation Type:* For consistency, burst mutation with a Cauchy distribution [57] is applied in all experiments.

14. *Output Neurons:* The number of motor output neurons assigned to any given ANN controller. The number of motor output neurons is determined by the experimenter *a priori*, where the appropriate number is task dependent. The number of motor output neurons remains static for any given NE experiment, and determines genotype length.

15. *Population Elite Portion:* A fittest portion of each genotype population. This portion is a part of the genotype selection process for the HomCNE, HetCNE, CCGA, Multi-Agent ESP and CONE methods.

16. *Population Size:* The number of genotypes (ANN controllers) in a given population, where there are initially n populations.

17. *Specialization Distance (SD) Value:* A normalized value within the range [0.0, 1.0] that corresponds to the difference in behavioral specialization exhibited by two controllers: ANN_i and ANN_j. Calculation of the SD value is described in section 3.2.2.

18. *Weight Mutation Range:* The range of change in the value of an ANN controller weight connection effectuated by the mutation operator.

19. *Weight (Gene) Range:* The range of the value of each connection weight (gene) in any given ANN controller.

APPENDIX F: PREDATOR HEURISTIC CONTROLLER

In the shaping experiments for prey controller evolution, predators use heuristic controllers. A heuristic predator controller mandates that a predator's movement be deterministic when a prey is within light sensor range, and that movement be stochastic when a prey is beyond light sensor range. That is:

if ≥ 1 prey within light sensor range
  then Execute *deterministic movement*
  else Execute *stochastic movement*

Deterministic movement: uses the *max norm* greedy heuristic [82] as a method for selecting the direction of movement (an orientation between 0 and 360 degrees) that places a predator on a heading towards the prey. The greedy max norm heuristic measures the diagonal difference between a predator and a prey, and works via minimizing the distance between each of the predators, as well as each of the predators and a prey.

Stochastic movement moves a predator in the direction of a prey's last position (when last in light sensor range) 60% of the time, and in a random direction 30% of the time. No movement is made 10% of the time.

NOMENCLATURE

Activity: What is being done by one controller or a set of controllers [83].

Agent: Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [147].

Artificial collective behavior system: Systems where simulated (software) or physically embodied (robot) controllers interact in a common task environment in order to solve a collective behavior task.

Artificial Neural Network (ANN): In line with definitions adopted by evolutionary robotics researchers such as Nolfi and Floreano [124], ANN refers to a controller that maps sensory inputs to motor outputs as a function of a process that replicates the neural computation properties of biological neural networks [70].

Atomic Object: A term that is specific to the GACC task (chapter 6). The term *atomic object* is used interchangeably with *object*, and refers to a building block of a *complex object*. In the GACC task, a set of atomic objects delivered to a construction in a specific order constitutes the construction of a complex object.

Behavioral specialization: The predisposition for a controller to adopt a specific behavior, where this behavior is advantageous to task accomplishment. Behavioral specialization is applicable to both simulated and embodied controllers.

Biological collective behavior system: A natural system of organisms co-inhabiting an environment found in nature, where organisms interact at a local level with the result of exhibiting a global behavior at the system level. Examples of biological collective behavior systems include complex ecological communities such as social insect colonies [19], biological neural networks [9], multi-cellular organisms [66], economies of a nation, companies, corporations and other business organizations [1].

Caste: A set of controllers specialized to the same role [83].

Collective behavior: A global behavior that is derived via the local interactions between individual controllers situated in a common collective behavior task environment [111].

Collective behavior system: Either an artificial or biological collective behavior system [111].

Collective behavior task environment: A simulated or physical environment in which a set of simulated or embodied controllers operate with the goal of accomplishing a collective behavior task [111].

Collective behavior task: A task that is accomplished by a collective behavior, and could not otherwise be accomplished by any of the individuals operating in a collective behavior system [111].

Collective Neuro-Evolution (CONE): CONE is a principled method for automated controller design in collective behavior systems. CONE is the main contribution of this book and is described in chapter 3.

Complex adaptive system: Either biological or artificial systems, where the constituent components of the system interact at a local level so as to produce a global system level behavior [113].

Complex Object: A term that is specific to the GACC task (chapter 6). A complex object refers to a structure that is constructed from a set of atomic objects. In the context of the GACC task, a complex object is metaphorical, and represents a physical structure situated in hazardous or uninhabitable environments that would most appropriately be constructed by a robot team. Examples of real world complex objects include underwater laboratories and space stations.

Controller: A simulated (software) or embodied (robotic) entity designed to operate in a given environment and perform a given task. Simulated controllers are labeled as such since they operate within the confines of computer simulation. However, embodied controllers operate in a physical (real world) environment [124].

Conventional Neuro-Evolution (CNE): CNE is an approach that evolves ANN controllers, where a single population of genotypes is specified. Each genotype encodes all the connection weights of one ANN controller [195]. For the purposes of experiments reported in this book, CNE is divided into two methods. First, *Homogenous Conventional Neuro-Evolution* (HomCNE), and second, *Heterogenous Conventional Neuro-Evolution* (HetCNE). For the creation of a team of n ANN controllers, the mechanism used to select genotypes from the population differs between HomCNE and HetCNE. HomCNE selects one of the fittest genotypes from the population, decodes it into an ANN controller, and then clones the controller n times in order to construct a team. HetCNE selects n of the fittest genotypes from the population, and decodes them into a set of n controllers. These methods are described in section 2.3.1.

Cooperative Co-evolution: In terms of evolutionary computation research, cooperative co-evolution is an artificial evolution process that evaluates an

individual based on how well the individual cooperates with the other individuals in the task environment. Individuals succeed when they are able to effectively complement the behaviors of other individuals, and all individuals accomplish a given task [187].

Cooperative Co-evolutionary Genetic Algorithm (CCGA): The *Cooperative Co-evolutionary Genetic Algorithm* is one instantiation of a generalized cooperative co-evolution architecture proposed by Potter [136]. This cooperative co-evolution architecture uses multiple populations. Phenotypes from each population are decoded and evaluated together in a common task environment. In CCGA, genetic algorithms are used as a means of adapting genotypes in each population [136].

Epoch: A task trial executed for a given number of simulation iterations. Each epoch is initialized with different agent and environment conditions, such as agent positions and orientations, and resource locations. A given number of epochs represents one agent lifetime, which is equated with one generation in terms of an evolutionary algorithm [124].

Emergent behavior: Emergent behavior is considered to be any computation that achieves global affects, formally or stochastically, via working within a bounded number of neighbors and without global visibility [47].

Emergent specialization: Is specialization (either behavioral or morphological) that is derived as result of collective behavior system dynamics or in response to task and environment constraints [113].

Evaluation: In terms of evolutionary computation theory, this is the testing of a candidate solution in order to determine its fitness [35].

Enforced Sub-Populations (ESP): ESP is a cooperative co-evolutionary NE method. ESP creates and evolves p genotype populations for an ANN consisting of a single hidden layer of p neurons. For a given population, individual genotypes in the population encode the input-output weights of a neuron assigned to a given hidden layer position. Each generation of the evolutionary process, genotypes selected from each population are decoded into hidden layer neurons and evaluated together in the context of an ANN. Genotypes are assigned a fitness and recombined as a function of the performance of their corresponding neurons in the ANN [56].

Functional specialization: A specific function of an individual neuron operating within an ANN.

Gathering and Collective Construction (GACC) task: A collective behavior case study detailed in chapter 6. In the GACC task, it is the goal of a robot team to maximize the number of complex objects constructed during the team's lifetime.

Island Model: A parallel implementation of a genetic algorithm within a communication structure. The island model uses a fixed number of genotype populations (so called *islands*) which evolve competing solutions. Individual genotypes occasionally migrate from one island to another, so as to induce a gradual mixing of genetic material [43].

Individual: In terms of evolutionary computation, a member of the population of candidate solutions upon which evolutionary algorithms operate [43].

Lamarckian Evolution (Lamarckism): Where characteristics learn't or acquired during one's lifetime are genetically encoded and passed onto to one's offspring [61]. In the context of neuro-evolution, controllers that exploit Lamarckism are those where connection weights adapted due to a lifetime learning process are passed onto the next generation of controllers and thus propagated throughout the evolutionary process [60].

Lifetime: An agent (controller) lifetime lasts for q epochs, where each epoch consists of a number of simulation iterations [124].

Morphological specialization: A particular configuration or structure of an embodied entity, where such a structure is advantageous in terms of the controller behavior that it defines and influences [113].

Multi-Agent Enforced Sub-Populations (Multi-Agent ESP): Multi-Agent ESP is the application of ESP to collective behavior tasks [197]. Multi-Agent ESP creates n populations for evolving n ANN controllers. Each population consists of u sub-populations, where individual ANNs are constructed as in ESP. This process is repeated n times for n ANNs, which are then collectively evaluated in a task environment.

Multi-Rover task: A collective behavior task investigated as one of the case studies in this book. A team of simulated autonomous vehicles (rovers) operating in an unexplored environment attempt to maximize the value of features of interest (red rocks) detected over the course of the team's lifetime. The multi-rover task is described in chapter 5.

Neuro-Evolution (NE): The adaptation of ANN properties such as architecture, connection weights, and learning rate via the use of an evolutionary algorithm [194].

Ontogenetic adaptation: Adaptation due to a learning process [39].

Phylogenetic adaptation: Adaptation due to an evolutionary process [39].

Pursuit-evasion task: One of the collective behavior case studies investigated in this book. Multiple pursuers (predators) are required to interact cooperatively so as to capture (immobilize) one or more evaders (prey). Pursuit-evasion is commonly used within artificial life research to test both non-adaptive (typically game theoretic [87]) and adaptive (typically learning and evolution [68]) methods for controller design.

Red rock: A term that is specific to the multi-rover task (chapter 5). The term *red rock* is adapted from that introduced by Young *et al.* [199] and refers to discrete high-value features of interest on an unexplored terrain.

Role: The task assigned to an individual controller within a set of tasks given to a group of controllers [83].

Singularity: In mathematical terms, a point at which the derivative does not exist for a given function but every neighborhood of which contains points for which the derivative exists [81].

Species: In the context of evolutionary computation, species refers to a population of genotypes. The term species is applicable to the ESP, Multi-Agent ESP, and CCGA methods, since genotype recombination does not occur between populations [138].

Symbiotic Adaptive Neuro-Evolution (SANE): A cooperative co-evolutionary NE method. SANE uses a single genotype population which evolves an ANN. Individual genotypes are encoded as the input-output weights of hidden layer neurons. At each generation of the evolutionary process, genotypes selected from the population are decoded into hidden layer neurons for the purpose of constructing an ANN. This ANN is evaluated in a task, and fitness assigned to the genotypes corresponding to each of the neurons that participated in the ANN. SANE also evolves the hidden layer positions that genotypes (neurons) are assigned to [102].

Task: What has to be done by an individual controller [83].

Task domain: A set of related tasks.

SAMENVATTING

De titel van dit proefschrift is: *Neuro-Evolutie voor Emergente Specialisatie in Collectief Gedrag Systemen.* De belangrijkste bijdrage van het proefschrift is een nieuwe methode: *Collectieve Neuro-Evolutie* (CONE) dat werkt binnen computersimulatie om collectieve gedrag problemen op te lossen. Een collectieve gedrag probleem is een taak die alleen kan worden opgelost door meerdere agenten (computerprogramma's) te laten samenwerken. Het onderwerp van dit proefschrift is gelegen in het veld van neuro-evolutie onderzoek: het snijvlak van evolutionaire en neurale algoritmiek onderzoek. Gegeven een collectief gedrag probleem ontwerpt CONE een multi-agent systeem waarin de agenten samenwerken om een optimale oplossing van het probleem te vinden. CONE werkt via het aanpassen van agent gedrag en agent interacties tijdens een simulatie, en gebruikt specialisatie die ontstaat als agenten interacteren. Emergente specialisatie verwijst naar het gespecialiseerde probleemoplossend gedrag van de agenten. Met behulp van emergente specialisatie als een probleemoplossend mechanisme kan CONE beter presteren dan verwante methoden.

BIBLIOGRAPHY

[1] H. Abdel-Rahman. When do cities specialize in production. *Reg Sci Urban Econ*, 26(1):1–22, 2001.

[2] A. Agogino. *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, 2003.

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

[4] P. Angeline and J. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270, San Mateo, USA, 1993. Morgan Kaufmann.

[5] T. Arai, E. Pagello, and L. Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[6] R. Arkin. *Behavior based Robotics.* MIT Press, Cambridge, USA, 1998.

[7] R. Arkin and T. Balch. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1999.

[8] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, USA, 1984.

[9] K. Baev. *Biological Neural Networks.* Birkuser, Berlin, Germany, 1997.

[10] T. Balch. *Behavioral Diversity in Learning Robot Teams. PhD Thesis.* College of Computing, Georgia Institute of Technology, Altanta, USA, 1998.

[11] T. Balch. Measuring robot group diversity. In *Robot teams: From diversity to polymorphism*, pages 93–135. Peters, Natick, USA, 2002.

[12] T. Balch. Taxonomies of multi-robot task and reward. In *Robot teams: From diversity to polymorphism*, pages 23–35. Peters, Natick, USA, 2002.

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

[14] J. Blumenthal and G. Parker. Co-evolving team capture strategies for dissimilar robots. In *AAAI Artificial Multi-Agent Learning Symposium*, pages 15–23, Arlinton, Virginia, 2004. AAAI Press.

[15] J. Blumenthal and G. Parker. Competing sample sizes for the co-evolution of heterogeneous agents. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1438–1443, Sendai, Japan, 2004. IEEE Press.

[16] J. Blumenthal and G. Parker. Punctuated anytime learning for evolving multi-agent capture strategies. In *Proceedings of the Congress on Evolutionary Computation*, pages 1820–1827, Portland, USA, 2004. IEEE Press.

[17] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, Oxford, England, 1998.

[18] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J. Deneubourg. Adaptive task allocation inspired by a model of division of labour in social insects. In *Bio-Computing and Emergent Computation*, pages 36–45. World Scientific, Singapore, 1997.

[19] E. Bonabeau, G. Theraulaz, and J. Deneubourg. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London B*, 263(1):1565–1569, 1996.

[20] B. Bryant. *Evolving Visibly Intelligent Behavior for Embedded Game Agents. PhD thesis.* Department of Computer Sciences, The University of Texas, Austin, Texas, USA, 2006.

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

[22] M. Bugajska and A. Schultz. Co-evolution of form and function in the design of autonomous agents: Micro air vehicle project. In *Workshop on Evolution of Sensors, GECCO 2000*, pages 240–244, Las Vegas, USA, 2000.

[23] L. Bull. On zcs in multi-agent environments. In *Proceedings of Parallel Problem Solving From Nature*, pages 471–480, Amsterdam, The Netherlands, 1998. Springer.

[24] L. Bull, T. Fogarty, and M. Snaith. 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*, pages 382–388, San Mateo, USA, 1995. Morgan Kauffman.

[25] L. Busoniu, R. Babuska, and B. DeSchutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[26] N. Calderone and R. Page. Genotypic variability in age polyethism and task specialization in the honey bee. *Apis mellifera. Behav. Ecol. Sociobiol*, 22(1):17–25, 1988.

[27] C. Campos, G. Theraulaz, E. Bonabeau, and J. Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2):83–94, 2001.

[28] A. Cavalcanti and R. Freitas. Nanorobotics control design: A collective behavior approach for medicine. *IEEE Trans. NanoBioSci.*, 4(6):133–140, 2005.

[29] K. Charles. *Ecological Methodology*. HarperCollins, New York, USA, 1989.

[30] C. Chen. The leaders and followers among the ants in nest-building. *Physiol. Zool.*, 10(1):437–455, 1937.

[31] C. Chen. Social modification of the activity of ants in nest-building. *Physiol. Zool.*, 10(1):420–436, 1937.

[32] N. Cole, S. Louis, and C. Miles. Using a genetic algorithm to tune first-person shooter bots. In *Proceedings of the 2004 Congress on Evolutionary Computation, vol. 1*, pages 139–145, Piscataway, USA, 2004. IEEE Press.

[33] D. D'Ambrosio and K. Stanley. Generative encoding for multiagent learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Atlanta, USA, 2008. ACM Press.

[34] E. De Jong and L. Steels. A distributed learning algorithm for communication development. *Complex Systems*, 14(4):315–334, 2003.

[35] K. De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, USA, 2006.

[36] J. Deneubourg, S. Goss, J. Pasteels, D. Fresneau, and J. Lachaud. Self-organization mechanisms in ant societies (ii): learning in foraging and division of labor. In *From Individual to Collective Behavior in Social Insects*, pages 177–196. Birkhauser, Basel, Switzerland, 1987.

[37] J. Deneubourg, G. Theraulaz, and R. Beckers. Swarm-made architectures. In *Proceedings of the European Conference on Artificial Life*, pages 123–133, Amsterdam, The Netherlands, 1991. Elsevier Academic Publishers.

[38] J. Denzinger and M. Fuchs. Experiments in learning prototypical situations for variants of the pursuit game. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 48–55, Kyoto, Japan, 1996. MIT Press.

[39] Dictionary. *Dictionary of Biology*. House Books Limited, Oxford University Press, Oxford, England, 2000.

[40] R. Drežewski. A model of co-evolution in multi-agent system. In *Multi-Agent Systems and Applications III*, pages 314–323, Berlin, Germany, 2003. Springer-Verlag.

[41] R. Drežewski. Co-evolutionary multi-agent system with speciation and resource sharing mechanisms. *Computing and Informatics*, 25(4):305–331, 2006.

[42] J. Edmonds. Path, trees, and flowers. *Canadian J. Math.*, 17(1):449–467, 1965.

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

[44] J. Elman. Finding structure in time. *Cognitive Science*, 14(1):179–211, 1990.

[45] R. Eriksson and B. Olsson. Cooperative coevolution in inventory control optimisation. In *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*.

[46] S. Ficici and J. Pollack. Challenges in coevolutionary learning: Armsrace dynamics, openendedness, and mediocre stable states. In *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, Cambridge, USA, 1998. MIT Press.

[47] D. Fisher and H. Lipson. Emergent algorithms - a new method for enhancing survivability in unbounded systems. In *Proceedings of the Thirty second Annual Hawaii International Conference on System Sciences-Volume 7*, pages 7043–7053, Washington, DC, USA, 1999. IEEE Press.

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

[49] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[50] D. Floreano, S. Mitri, S. Magnenat, and L. Keller. Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17(1):514–519, 2007.

[51] E. Folgado, M. Rincon, J. Alvarez, and J. Mira. A multi-robot surveillance system simulated in gazebo. In *Nature Inspired Problem-Solving Methods in Knowledge Engineering*, pages 202–211. Springer Berlin, Heidelberg, Germany, 2007.

[52] P. Funes, B. Orme, and E. Bonabeau. Evolving emergent group behaviors for simple humans agents. In *Proceedings of the Seven European Conference on Artificial Life*, pages 76–89, Berlin, Germany, 2003. Springer-Verlag.

[53] D. Futuyma and M. Slatkin. In D. Futuyma and M. Slatkin, editors, *Coevolution*. Sinauer Associates, Sunderland, Massachusetts, USA, 1983.

[54] G. Gause. *The Struggle for Existence*. Williams and Wilkins, Baltimore, USA, 1934.

[55] 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.

[56] F. Gomez. *Robust Non-Linear Control Through Neuroevolution. PhD thesis*. Department of Computer Sciences, The University of Texas, Austin, Texas, USA, 2003.

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

[58] F. Gomez and R. Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1356–1361, Stockholm, Sweden, 1999. Morgan Kaufmann.

[59] F. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 2084–2095, Chicago, USA, 2003. ACM Press.

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

[61] S. Gould. *The Structure of Evolutionary Theory*. Belknap Press, Cambridge, USA, 2002.

[62] P. Grant. *Ecology and Evolution of Darwin's Finches*. Princeton University Press, Princeton, USA, 1986.

[63] J. Grefenstette. Credit assignment in rule discovery systems. *Machine Learning*, 3(3):225–246, 1995.

[64] R. Gross, E. Tuci, F. Mondada, and M. Dorigo. Object transport by modular robots that self-assemble. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 2558–2564, Los Alamitos, CA, 2006. IEEE Computer Society Press.

[65] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 81–89, Cambridge, USA, 1996. MIT Press.

[66] D. Hawthorne. Genetic linkage of ecological specialization and reproductive isolation in pea aphids. *Nature*, 412(1):904–907, 2001.

[67] T. Haynes and S. Sen. Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations*, 1(4):1–20, 1996.

[68] T. Haynes and S. Sen. Evolving behavioral strategies in predators and prey. In *Adaptation and Learning in Multi-Agent Systems: Lecture Notes in Computer Science*, pages 113–126. Springer-Verlag, Berlin, Germany, 1996.

[69] L. Hercog and T. Fogarty. Social simulation using a multi-agent model based on classifier systems: The emergence of vacillating behaviour in "el farol" bar problem. In *Proceedings of the Fourth International Workshop on Learning Classifier Systems*, pages 362–366, San Francisco, USA, 2001. Springer.

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

[71] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. PhD Thesis*. University of Michigan Press, Ann Arbor, USA, 1975.

[72] J. Holland. Properties of the bucket brigade. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 1–7, Mahwah, USA, 1985. Lawrence Erlbaum Associates, Inc.

[73] J. Holland and J. Reitman. Cognitive systems based on adaptive algorithms. *Pattern Directed Inference Systems*, 7(2):125–149, 1978.

[74] W. Hsu and S. Gustafson. Layered learning in genetic programming for a cooperative robot soccer problem. In *Proceedings of the Fourth European Conference on Genetic Programming*, pages 291–301, Como, Italy, 2001. Springer-Verlag.

[75] A. Ijspeert, A. Martinoli, A. Billard, and L. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots.*, 11(2):149–171, 2001.

[76] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence*, 4(1):237–285, 1996.

[77] S. Kenny and B. Kirby. Complex systems in language evolution: the cultural emergence of compositional structure. *Artificial Life*, 9(4):371–386, 2003.

[78] S. Kirby and J. Hurford. The emergence of linguistic structure: An overview of the iterated learning model. In A. Cangelosi and D. Parisi, editors, *Simulating the Evolution of Language*, pages 121–148. Springer Verlag, London, United Kingdom, 2002.

[79] H. Kitano and M. Asada. The robocup humanoid challenge as the millennium challenge for advanced robotics. *Advanced Robotics.*, 13(1):723–736, 2000.

[80] H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. Robocup: A challenge problem for ai. *AI Magazine*, 18(1):73–85, 1997.

[81] K. Knopp. Singularities. In *Theory of Functions Parts I and II*, pages 117–139. Dover, New York, USA, 1996.

[82] R. Korf. A simple solution to pursuit games. In *Working Papers of the Eleventh International Workshop on DAI*, pages 195–213, Geneva, Switzerland, 1992. Springer-Verlag.

[83] 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(1):65–84, 2000.

[84] C. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems: Special Issue on Biomimetic Robots*, 1(1):20–29, 1999.

[85] R. Kube and H. Zhang. Collective robotics: from social insects to robots. *Adaptive Behaviour*, 2(2):189–218, 1994.

[86] C. Langton. *Artificial Life: An Overview.* MIT Press, Cambridge, USA, 1995.

[87] R. Levy and J. Rosenschein. Game theoretic approach to distributed artificial intelligence and the pursuit problem. In *Decentralized AI III*, pages 129–146. Springer-Verlag, Kaiserslautern, Germany, 1992.

[88] L. Li, A. Martinoli, and Y. Mostafa. Emergent specialization in swarm systems. In *Lecture notes in computer science: Vol. 2412. Intelligent data engineering and automated learning*, pages 261–266. Springer-Verlag, Berlin, Germany, 2002.

[89] L. Li, A. Martinoli, and A. Yaser. Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior.*, 12(3):199–212, 2004.

[90] H. Lipson and J. Pollack. Automatic design and manufacture of robotic life forms. *Nature*, 406(1):974–978, 2000.

[91] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I*, pages 398–411. Springer-Verlag, Berlin, Germany., 1998.

[92] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. MASON: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[93] D. MacKay. *Information Theory, Inference and Learning Algorithms*. University of Cambridge Press, Cambridge, United Kingdom, 2003.

[94] B. MacLennan and G. Burghardt. Synthetic ethology and the evolution of cooperative communication. *Adaptive Behavior*, 2(2):161–188, 1993.

[95] S. Martello and P. Toth. Linear assignment problems. surveys in combinatorial optimization. *Ann Discrete Math*, 31(1):259–282, 1987.

[96] A. Martinoli, Y. Zhang, P. Prakash, E. Antonsson, and R. Olney. Towards evolutionary design of intelligent transportation systems. In *Eleventh International Symposium on New Technologies for Advanced Driver Assistance Systems*, pages 283–290, Siena, Italy., 2002. ATA Press.

[97] M. Mataric. Behavior-based control: examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):62–78, 1997.

[98] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative multi-robot box-pushing. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3*, pages 556–561, Los Alamitos, USA, 1995. IEEE Computer Society Press.

[99] H. Matsubara, I. Noda, and K. Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass and play in soccer. In *Adaptation, Co-evolution, and Learning in Multi-agent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 63–67, Boston, USA, 1996. AAAI Press.

[100] O. Miglino, H. Hautop, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1995.

[101] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In *Proceedings of Third International Symposium on Experimental Robotics*, pages 501–513, Kyoto, Japan., 1993. IEEE Press.

[102] D. Moriarty. *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. PhD thesis*. Department of Computer Sciences, The University of Texas, Austin, Texas, 1997.

[103] D. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(1):373–399, 1997.

[104] A. Murciano and J. Millan. Learning signaling behaviors and specialization in cooperative agents. *Adaptive Behavior*, 5(1):5–28, 1997.

[105] A. Murciano, J. Millan, and J. Zamora. Specialization in multi-agent systems through learning. *Biological Cybernetics*, 76(1):375–382, 1997.

[106] Y. Ng and X. Yang. Specialization, information, and growth: A sequential equilibrium analysis. *Rev Dev Econ*, 1(1):257–274, 1997.

[107] S. Nishimura and T. Ikegami. Emergence of collective strategies in a prey-predator game model. *Artificial Life*, 3(1):243–260, 1997.

[108] S. Nishimura and I. Takashi. Emergence of collective strategies in a predator-prey game model. *Artificial Life*, 3(1):243–260, 1997.

[109] G. Nitschke. Co-evolution of cooperation in a pursuit evasion game. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2037–2042, Las Vagas, USA, 2003. IEEE Press.

[110] G. Nitschke. Designing emergent cooperation: a pursuit-evasion game case study. *Artificial Life and Robotics*, 9(4):222–233, 2005.

[111] G. Nitschke. Emergence of cooperation: State of the art. *Artificial Life*, 11(3):367–396, 2005.

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

[113] G. Nitschke, M. Schut, and A. Eiben. Emergent specialization in biologically inspired collective behavior systems. In *Intelligent Complex Adaptive Systems*, pages 100–140. IGI Publishing, New York, USA, 2007.

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

[115] G. Nitschke and D. van Krevelen. Neuro-evolution for a gathering and collective construction task. In Conor Ryan, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*. ACM Press, 2008.

[116] I. Noda and P. Stone. The robocup soccer server and cmunited clients: Implemented infrastructure for mas research. *Autonomous Agents and Multi-Agent Systems*, 7(1):101–120, 2001.

[117] C. Noirot and J. Pasteels. Ontogenetic development and the evolution of the worker caste in termites. *Experientia.*, 43(1):851–860, 1987.

[118] S. Nolfi. Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. In *Evolutionary Robotics 97 - From Intelligent Robotics to Artificial Life*, pages 243–254. AAI Books, Ottawa, Canada, 1997.

[119] S. Nolfi. Using emergent modularity to develop control system for mobile robots. *Adaptive Behavior*, 5(1):343–363, 1997.

[120] S. Nolfi. *Evorobot 1.1 User Manual. Technical Report.* Institute of Cognitive Sciences, National Research Council, Rome, Italy, 2000.

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

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

[123] S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do arm races arise in artificial evolution. *Artificial Life*, 4(4):311–335, 1999.

[124] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* MIT Press, Cambridge, USA, 2000.

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

[126] S. O'Donnell. Effects of experimental forager removals on division of labour in the primitively eusocial wasp polistes instabilis. *Behaviour*, 135(2):173–193, 1998.

[127] S. ODonnell and R. Jeanne. Forager specialization and the control of nest repair in polybia occidentalis olivier (hymenoptera: Vespidae). *Behavioral Ecology and Sociobiology*, 27(1):359–364, 1990.

[128] M. O'Riain, J. Jarvis, R. Alexander, R. Buffenstein, and C. Peeters. Morphological castes in a vertebrate. *Proceedings of the National Academy of Sciences of the United States of America.*, 97(24):13194–13197, 2000.

[129] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 3(11):387–434, 2005.

[130] G. Parker and P. Nathan. Evolving sensor morphology on a legged robot in niche environments. In *World Automation Congress 2006*, pages 1–10, Budapest, Hungary, 2006. IEEE Press.

[131] A. Perez-Uribe, D. Floreano, and L. Keller. Effects of group composition and level of selection in the evolution of cooperation in artificial ants. In *Advances of Artificial Life: Proceedings of the Seventh European Conference on Artificial Life*, pages 128–137, Dortmund, Germany, 2003. Springer.

[132] G. Peterson. A day of great illumination: B. f. skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior*, 82(3):317–328, 2004.

[133] R. Pfeifer, F. Iida, and G. Gomez. Designing intelligent robots: On the implications of embodiment. *Journal of Robotics Society of Japan*, 24(7):9–16, 2006.

[134] J. Polechovà and N. Barton. Speciation through competition: a critical review. *Evolution*, 59(6):1194–1210, 2005.

[135] J. Polechovà and D. Storch. Ecological niche. In *Encyclopedia of Ecology*. Elsevier, Amsterdam, The Netherlands, 2005.

[136] M. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution.* Department of Computer Science, George Mason University, Fairfax, USA, 1997.

[137] M. Potter and K. De Jong. Evolving neural networks with collaborative species. In *Proceedings of the Summer Computer Simulation Conference*, pages 340–345. The Society of Computer Simulation, 1995.

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

[139] M. Potter, K. De Jong, and J. Grefenstette. A co-evolutionary approach to learning sequential decision rules. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 366–372, San Franciso, USA, 1995. Morgan Kaufmann.

[140] M. Potter, L. Meeden, and A. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1337–1343, Seattle, USA, 2001. AAAI Press.

[141] M. Quinn, L. Smith, G. Mayley, and P. Husbands. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 361(1):2321–2344, 2003.

[142] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds.* MIT Press, Cambridge, USA, 1997.

[143] T. Revello and R. McCartney. Generating war game strategies using a genetic algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1086–1091, Piscataway, USA, 2002. IEEE Press.

[144] C. Reynolds. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–36, 1987.

[145] S. Robson and J. Traniello. Key individuals and the organization of labor in ants. In *Information Processing in Social Insects*, pages 239–259. Springer Verlag, Basel, Switzerland, 1999.

[146] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[147] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach. Second Edition*. University of Michigan Press, Ann Arbor, USA, 2003.

[148] M. Saptharishi, C. Oliver, C. Diehl, K. Bhat, J. Dolan, A. Trebi-Ollennu, and P. Khosla. Distributed surveillance and reconnaissance using multiple autonomous atvs: Cyberscout. *IEEE Transactions on Robotics and Automation*, 18(5):826–836, 2002.

[149] A. Schultz and M. Bugajska. 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*, pages 154–166, Chicago, 2000. AAAI Press.

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

[151] H. Seligmann. Resource partition history and evolutionary specialization of subunits in complex systems. *Biosystems*, 51(1):31–39, 1999.

[152] O. Sigaud and W. Stewart. Learning classifier systems: a survey. *Soft Computing*, 11(11):1065–1078, 2007.

[153] K. Sims. Evolving 3d morphology and behavior by competition. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 28–39, Cambridge, USA, 2004. MIT Press.

[154] Z. Skolicki and K. De Jong. The influence of migration sizes and intervals on island models. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 1295–1302. ACM Press, 2005.

[155] A. Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations. Fifth edition.* Methuen and Co., Ltd. First published: 1776., London, United Kingdom, 1904.

[156] S. Smith. *A Learning System Based on Genetic Adaptive Algorithms. PhD Thesis*. University of Pittsburgh Press.

[157] D. Solow and J. Szmerekovsky. Mathematical models for explaining the emergence of specialization in performing tasks. *Complexity*, 10(1):37–48, 2004.

[158] K. Stanley. *Efficient Evolution of Neural Networks Through Complexification. Ph. D. Dissertation*. Department of Computer Sciences, The University of Texas, Austin, USA, 2004.

[159] K. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines: Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[160] K. Stanley, B. Bryant, and R. Miikkulainen. Evolving neural network agents in the nero video game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, pages 182–189, Piscataway, USA, 2005. IEEE Press.

[161] 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.

[162] K. Stanley, D'Ambrosio, and J. Gauci. Hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, To appear, 2008.

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

[164] L. Steels and F. Kaplan. Aibo's first words: The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32, 2001.

[165] W. Stolzmann. Anticipatory classifier systems. In J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. Fogel, M. Garzon, D. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming*, pages 658–664. Morgan Kaufmann Publishers, San Francisco, USA, 1998.

[166] P. Stone and M. Veloso. Using decision tree confidence factors for multiagent control. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 110–116, Minneapolis, USA, 1998. ACM Press.

[167] H. Sugie, Y. Inagaki, S. Ono, H. Aisu, and T. Unemi. Placing objects with multiple mobile robotsmutual help using intention inference. In *Proc. of the 1995 IEEE Int. Conf. on Robotics and Automation, vol. 2*, pages 2181–2186, Los Alamitos, USA, 1995. IEEE Computer Society Press.

[168] R. Sutton and A. Barto. *An Introduction to Reinforcement Learning.* John Wiley and Sons, Cambridge, USA, 1998.

[169] D. Tarapore, D. Floreano, and L. Keller. Influence of the level of polyandry and genetic architecture on division of labour. In *The Tenth International Conference on the Simulation and Synthesis of Living Systems (Alife X)*, pages 358–364, Cambridge, USA, 2006. MIT Press.

[170] G. Theraulaz and E. Bonabeau. Coordination in distributed building. *Science*, 269(1):686–688, 1995.

[171] G. Theraulaz, E. Bonabeau, and J. Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, 60(1):753–807, 1998.

[172] G. Theraulaz, E. Bonabeau, and J. Deneubourg. Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B*, 265(1):327–332, 1998.

[173] G. Theraulaz, J. Gervet, and S. Semenoff. Social regulation of foraging activities in polistes dominulus christ: a systemic approach to behavioural organization. *Behaviour*, 116(1):292–320, 1991.

[174] A. Thompson, I. Harvey, and P. Husbands. Unconstrained evolution and hard consequences. In *Towards Evolvable Hardware: The evolutionary engineering approach, volume 1062 of LNCS.*, pages 135–165, Berlin, Germany, 1996. Springer-Verlag.

[175] V. Trianni, R. Gross, T. Labella, E. Sahin, and M. Dorigo. Evolving aggregation behaviors in a swarm of robots. In *Advances in Artificial Life: Proceedings of the Seventh European Conference on Artificial Life*, pages 865–874. Springer-Verlag, Dortmund, Germany, 2003.

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

[177] R. Watson, S. Ficici, and J. Pollack. Embodied evolution: A response to challenges in evolutionary robotics. In *Eighth European Workshop on Learning Robots*, pages 14–22. Springer Verlag, Lausanne, Switzerland, 1999.

[178] R. Watson, S. Ficici, and J. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *1999 Congress on Evolutionary Computation*, pages 335–342, Washington D.C., USA, 1999. IEEE Press.

[179] R. Watson, S. Ficici, and J. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, 2002.

[180] T. Wenseleers, F. Ratnieks, and J. Billen. Caste fate conflict in swarm-founding social hymenoptera: an inclusive fitness analysis. *Evolutionary Biology.*, 16(1):647–658, 2003.

[181] J. Werfel, Y. Bar-Yam, and R. Nagpal. Building patterned structures with robot swarms. In *International Joint Conference on Artificial Intelligence*, pages 1495–1502, Edinburgh, Scotland, United Kingdom, 2005. AAAI Press.

[182] J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal. Distributed construction by mobile robots with enhanced building blocks. In *IEEE International Conference on Robotics and Automation*, pages 2787–2794, Orlando, Florida, USA, 2006. IEEE Press.

[183] J. Werfel and R. Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21(2):20–28, 2006.

[184] 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, Chicago, 2003. AAAI Press.

[185] Darrell Whitley, Soraya B. Rana, and Robert B. Heckendorn. Island model genetic algorithms and linearly separable problems. In *Evolutionary Computing Workshop*, pages 109–125. Morgan Kaufmann, 1997.

[186] R. Wiegand. Applying diffusion to a cooperative coevolutionary model. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, pages 560–569, London, United Kingdom, 1998. Springer-Verlag.

[187] R. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms. PhD. Thesis.* George Mason University Press, George Mason University, Fairfax, USA, 2004.

[188] S. Wilson. Knowledge growth in an artificial animat. In *Proceedings of the First international Conference on Genetic Algorithms*, pages 16–23, Mahwah, USA, 1985. Lawrence Erlbaum Associates, Inc.

[189] S. Wilson. Zcs: A zeroth-level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.

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

[191] B. Woodward, J. Winn, and F. Fish. Morphological specializations of baleen whales associated with hydrodynamic performance and ecological niche. *Journal of Morphology*, 267(11):1284–1294, 2006.

[192] J. Wu, Z. Di, and Z. Yang. Division of labor as the result of phase transition. *Physica A*, 7(1):323–663, 2003.

[193] G. Yannakakis, J. Levine, and J. Hallam. An evolutionary approach for interactive computer games. In *Proceedings of the 2004 Congress on Evolutionary Computation*, pages 986–993, Piscataway, USA, 2004. IEEE Press.

[194] X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[195] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

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

[197] C. Yong and R. Miikkulainen. *Coevolution of Role-Based Cooperation in Multi-Agent Systems. Technical Report AI01-287.* Department of Computer Sciences, The University of Texas, Austin, USA, 2007.

[198] 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, Big Sky, USA, 2002. IEEE Press.

[199] L. Young, G. Pisanich, and C. Ippolito. Aerial explorers. In *Proceeding of the 43rd AIAA Aerospace Sciences Meeting and Exhibit*, pages 4–12, Reno, Nevada, USA, 2005. AIAA Press.

[200] L. Young, G. Pisanich, C. Ippolito, and R. Alena. Aerial vehicle surveys of other planetary atmospheres and surfaces: imaging, remote-sensing, and autonomy technology requirements. In *Proceedings of the SPIE, Real-Time Imaging IX, Volume 5671*, pages 183–199, Reno, Nevada, USA, 2005. SPIE Press.

[201] N. Zaera, D. Cliff, and J. Bruten. *(Not) Evolving Collective Behaviors in Synthetic Fish (Tech. Rep.).* Hewlett-Packard Laboratories, Bristol, England, 1996.

[202] Y. Zhang, A. Martinoli, and E. Antonsson. Evolutionary design of a collective sensory system. In *The 2003 AAAI Spring Symposium on Computational Synthesis*, pages 283–290, Stanford, USA, 2003. AAAI Press.

INDEX