Neuro-Evolution Methods for Gathering and Collective Construction

Geoff Nitschke

Department of Computer Science, University of Pretoria, South Africa
gnitschke@cs.up.co.za

Abstract. This paper evaluates the *Collective Neuro-Evolution* (CONE) method, comparative to a related controller design method, in a simulated multi-robot system. CONE solves collective behavior tasks, and increases task performance via facilitating behavioral specialization. Emergent specialization is guided by genotype and behavioral specialization difference metrics that regulate genotype recombination. CONE is comparatively evaluated with a similar *Neuro-Evolution* (NE) method in a *Gathering and Collective Construction* (GACC) task. This task requires a multi-robot system to gather objects of various types and then cooperatively build a structure from the gathered objects. This collective behavior task requires that robots adopt complementary and specialized behaviors in order to solve. Results indicate that CONE is appropriate for evolving collective behaviors for the GACC task, given that this task requires behavioral specialization.

1 Introduction

In fields of research such as *multi-robot systems* [11], it is desirable to reproduce the underlying mechanisms that result in replicating the success of biological collective behavior systems. One such mechanism is *emergent behavioral specialization* [10]. In the study of controller design methods that solve various collective behavior tasks emergent specialization is not used as a problem solving mechanism, but rather emerges as an ancillary result of the system accomplishing its given task. Collective behavior tasks are those requiring cooperative behavior.

This paper applies and tests the *Collective Neuro-Evolution* (CONE) method. CONE is a novel controller design method that solves collective behavior tasks via purposefully facilitating emergent behavioral specialization is currently lacking. CONE adapts a set of *Artificial Neural Network* (ANN) controllers for the purpose of solving collective behavior tasks. The advantage of CONE is that it increases collective behavior task performance or attains collective behavior solutions that could not otherwise be attained without specialization.

In line with state of the art methods for controller design [6], this research supports NE as an appropriate approach for controller design within continuous and partially observable collective behavior task environments. NE has been successfully applied to solve a disparate range of collective behavior tasks that include multi-agent computer games [2], pursuit-evasion games [12], and coordinated movement [1]. Such collective behavior tasks require different agents (controllers) to adopt complementary specialized behaviors in order to solve.

The *Gathering and Collective Construction* (GACC) case study presented in this paper is an initial step towards elucidating that specialization that emerges during controller evolution, can be used as part of a collective behavior problem solving process. Experiments elucidate that CONE, comparative to Multi-Agent ESP, is able to effectuate behavioral specialization in a set of ANN controllers, where such specialization increases collective behavior task performance.

Research Goal: To demonstrate that CONE is appropriate for deriving behavioral specialization in a team of simulated robots, where such specialization gives rise to successful collective construction behaviors.

Hypothesis: CONE facilitates emergent behavioral specialization when evolving a team's collective behavior (in tasks that require specialization), where such specialization contributes to a higher task performance, comparative to the task performance of Multi-Agent ESP evolved teams.

GACC Task: This task requires that a team of simulated robots search an environment and gather a set of atomic objects and then use these objects in the cooperative construction of a complex object. Task performance is measured as the number of atomic objects delivered to a *home area* in a given sequence.

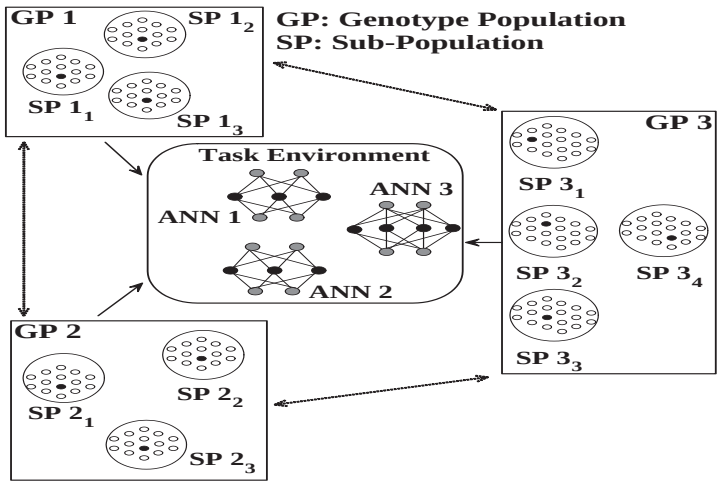

Fig. 1. CONE / Multi-Agent ESP Example. Three ANN controllers are derived from three populations and evaluated in a collective behavior task. CONE: Double ended arrows indicate self regulating recombination occurring *between* populations. Multi-Agent ESP: Recombination only occurs *within* sub-populations.

2 Neuro-Evolution Methods

2.1 Multi-Agent ESP: Multi-Agent Enforced Sub-Populations

Multi-Agent ESP is the application of the ESP NE method [8] 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 then collectively evaluated in a task environment. Figure 1 illustrates an example of Multi-Agent ESP using three populations (controllers). Multi-Agent ESP is comprehensively described in related work [12].

2.2 CONE: Collective Neuro-Evolution

Due to space constraints, this section only presents an overview of the novel contributions of *Collective Neuro-Evolution* (CONE). For a comprehensive description of CONE, refer to related work [10]. CONE is a cooperative co-evolution method that adapts a group of ANN controllers. Given n genotype populations, CONE evolves one controller from each population, where controllers must cooperate to solve collective behavior tasks. Controllers are collectively evaluated in a task environment according to how well they solve the given collective behavior task. Each controller is a feed-forward ANN with one hidden layer that is fully connected to the input and output layers. Each hidden layer neuron of each controller is encoded as one genotype. CONE evolves the connection weights of hidden layer neurons, and then combines these neurons into complete controllers. An example of CONE using three controllers (and thus three genotype populations) is presented in figure 1. Unlike related methods such as Multi-Agent ESP, CONE uses *genotype* and *behavioral specialization* (GDM and SDM, respectively) difference metrics to regulate genotype *between* and *within* populations. CONE is an extension of Multi-Agent ESP that includes the following.

1. **GDM:** Adaptively regulates genotype recombination *between* populations, based on neuron (genotype) connection weight similarities [10].
2. **SDM:** Adaptively regulates recombination based on behavioral specialization (CONE uses a specialization metric described in related work [7]) similarities exhibited by controllers [10].
3. **Controller size adaptation:** Adapting the number of hidden layer neurons in each controller facilitates of the evolution of behavioral specialization by CONE, via allowing different controllers to evolve to different sizes. That is, controllers of varying sizes and complexity are often appropriate for solving a set of sub-tasks of varying complexities [10].

2.3 Common Methods

The following describes the procedure used to construct controllers, and evaluate, recombine and mutate genotypes in Multi-Agent ESP and CONE.

- **Constructing Controllers:** Both CONE and Multi-Agent ESP initialize n populations. Population P_i ($i \in \{1, \ldots, n\}$ contains u_i sub-populations. Each sub-population (P_{ij}) contains m genotypes. P_{ij} contains genotypes encoding neurons (strings of floating point values) assigned to position j in the hidden layer of ANN_i. ANN_i is derived from P_i, where $j \leq u_i$.
- **Evaluate all Genotypes:** Systematically select each genotype g in each sub-population of each population, and evaluate g in the context of a complete controller. This controller (containing g) is evaluated together with n-1 other controllers. Other controllers are constructed via randomly selecting a neuron from each sub-population of each of the other populations. The evaluation results in a fitness being assigned to g.

- **Multi-Agent ESP Recombination:** After all neurons (genotypes) have been assigned a fitness [12], each neuron in the *elite portion* (table 1) is recombined with another neuron (randomly selected from the elite portion). Offspring genotypes completely replace each sub-population.
- **CONE Recombination:** The SDM is applied to each pair of controllers. For every pair of controllers within a *Specialization Distance* (SD in table 1) the populations from which these controllers were derived become candidates for recombination. The GDM is then applied to each pair of sub-populations *between* each pair of behaviorally similar populations (controllers). For each pair of sub-populations within a given *Genetic Distance* (GD in table 1) the elite portions of the sub-populations are recombined (using one-point crossover [3]). For every population that is not within the *SD* of another, or each sub-population that is not within the *GD* of another (in another population within the SD),recombination occurs within each sub-population.
- **Mutation:** After recombination, *burst mutation* with a *Cauchy* distribution [8] is applied to each genotype's gene with a given probability (table 1).

3 GACC Task: Experimental Design

Experiments test 30 robots with N building blocks (n type A, p type B, and q type C atomic objects) and a *home area* in a bounded two dimensional continuous environment. The home area (located at the environment's center) is where gathered objects are delivered and where the *complex object* is constructed. A complex object is the structure to be built from N atomic objects. The complex object can only be constructed if robots cooperate place objects of a given type in a predefined sequence. Two, three, and four robots are required to use type A, B, and C objects to construct the complex object. Experiments measure the impact of the Multi-Agent ESP or CONE method and an *environment* upon the number of *atomic objects delivered* in the correct sequence to the home area by the team. The experimental objective test the task performance and the contribution of specialization to performance in teams evolved by each method.

Team Fitness Evaluation: Team fitness (G) equals the total *number of atomic objects delivered in the correct sequence* to the home area. Individual fitness (g_v) equals the *number of atomic objects delivered in the correct sequence* by robot η over the course of its lifetime. The goal of the team is to maximize G. Robots do not maximize G directly, instead each robot η attempts to maximize its own private fitness function g_η, where g_η guides controller evolution.

Simulation: Table 1 presents the simulation and NE parameter settings. These parameter values were determined experimentally. Minor changes to these values produced similar results for both Multi-Agent ESP and CONE. Each experiment consists of 250 generations. One generation is a robot team's *lifetime*. Each robot lifetime lasts for 10 epochs. One epoch is 3000 simulation iterations, and represents a task scenario that tests different robot starting positions, and object locations in an environment. Team task performance is calculated as an average taken over all epochs of a team's lifetime. The best task performance is then selected for each run, and an average is calculated over 20 runs.

Table 1. GACC Simulation and Neuro-Evolution Parameters.

Simulation and Neuro-Evolution Parameters	
Number of robots / Genotype populations	30
Robot movement range / cost	0.001 / 0.01
Object/Robot/Home area detection sensor range	0.05
Object/Robot/Home area detection sensor accuracy	1.0
Object/Robot/Home area detection sensor cost	0.01
Robot initial energy	1000 units
Initial robot positions	Random (Excluding home area)
Environment width / height	1.0
Total number of type A, B, and C objects	Variable (table 2)
Atomic Object distribution (Initial positions)	Random (Excluding home area)
Generations / Epochs	250 / 10
Iterations per epoch (Robot team lifetime)	3000
Mutation (per gene) probability / Mutation range	0.05 / [-1.0, +1.0]
Genotype / Specialization distance (CONE)	[0.0, 1.0]
Population elite portion	50%
Weight (gene) range	[-10.0, +10.0]
Genotype length (Number of connection weights)	42
Genotypes per population	500

4 Robot Sensors, Actuators, and Controller

Detection Sensors: Each robot has eight *object* ([S-0, S-7]), eight *robot* ([S-8, S-15]), eight *home area* ([S-16, S-23]) detection sensors, and three *object demand* ([S-24, S-26]) sensors (figure 2). Each of the eight detection sensors covers one quadrant in a robot's 360 degree sensory Field Of View (FOV). Table 1 presents sensor range, accuracy, and cost.

- **Object Detection Sensors:** Object detection sensors need to be explicitly activated with one of three settings (A, B, C), for detecting type A, B, and C objects, respectively. This constitutes one action and consumes one simulation iteration. Sensor q returns the closest object type (for the current sensor setting) in quadrant q, divided by the squared distance to the robot.
- **Robot Detection Sensors:** The function of these constantly active sensors is to prevent collisions, and provide each robot with an indication of the current *state* of other robots within *this* robot's sensory FOV. *State* refers to if another robot is carrying an object and the *type* of the object being carried. Sensor q returns a value equal to the object type carried by the closest robot (A:1, B:2, C:3), divided by the squared distance to *this* robot.
- **Home Area Detection Sensors:** Sensor q returns a value inversely proportional to the distance to the home area, divided by the squared distance to *this* robot. Home area sensors are constantly active.
- **Object Demand Sensors:** These constantly active sensors indicate the current demand for object types A, B, C. During the construction process, the complex object broadcasts a signal that is received by each robot's object demand sensors, indicating the next required object type.

Movement Actuators: Two wheel motors control each robot's heading at a constant speed. Movement actuators need to be explicitly activated (motor outputs MO-4 and MO-5 in figure 2). This is one action which takes one simulation

iteration. A robot's heading is determined by normalizing and scaling motor output values (vectors dx and dy) by the maximum distance a robot can traverse in one iteration (d_{max}). That is: $dx = d_{max}(o_1 - 0.5)$, and $dy = d_{max}(o_2 - 0.5)$. Where: o_1 and o_2 are values of motor outputs MO-4 and MO-5, respectively.

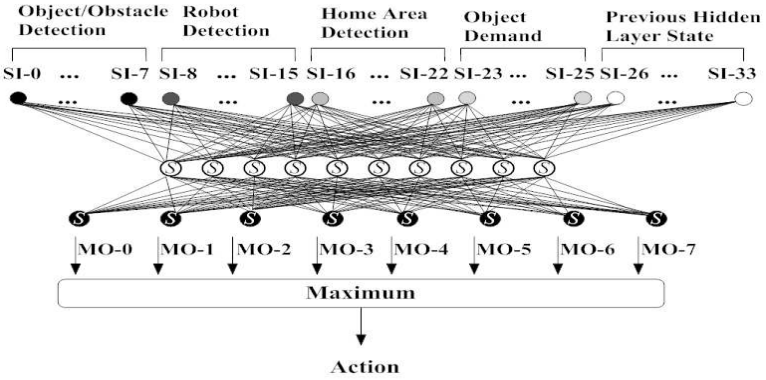

Fig. 2. Robot ANN Controller. For clarity, not all sensory input neurons are illustrated.

Object Gripper: Each robot is equipped with a gripper turret for gripping and transporting objects to the home area. The gripper has three actuator settings (A, B, C) for gripping and transporting type A, B, and C objects, respectively. The gripper needs to be explicitly activated which takes one iteration.

ANN Controller: Each robot uses a recurrent ANN controller [4], which fully connects 34 sensory input neurons to 10 hidden layer neurons to eight motor output neurons (figure 2). Hidden and output neurons are sigmoidal [9] units. Sensory input neurons [SI-26, SI-33] accept input as the previous activation state of the hidden layer. At each iteration, one of seven actions is executed by a robot. 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:** Calculate direction from motor outputs dx, dy.
5. **MO-5:** Activate gripper with setting A (figure 2).
6. **MO-6:** Activate gripper with setting B (figure 2).
7. **MO-7:** Activate gripper with setting C (figure 2).

5 Results and Discussion

Two experiment sets were run. In experiment set 1, robot teams were evolved in nine *simple environments*. Each simple environment contained a distribution only type A objects, and there was no predefined sequence for object delivery to the home area. In experiment set 2, robot teams were evolved in nine*complex environments*. Each complex environment contained a distribution of type A, B, and C objects. Table 2 presents the distribution of each object type for each

Table 2. Distribution of objects for each complex environment. Env: Environment.

Env	Object-A Number	Object-B Number	Object-C Number	Complex Object Build Sequence
1	1	2	7	CCACBCBCCC
2	2	4	4	CAACBBBCBC
3	3	6	1	BABAABCBBB
4	16	2	2	BAAAABAAAACAAAAAAAAC
5	4	14	2	BBBABABBBBABABBBCBCB
6	7	2	11	BACACACACACACACCCCCB
7	12	13	5	CBAABCAACAABBBAACABABBBAACBBBB
8	13	14	3	BABABABAAACABABAABBBBCAACABBBB
9	4	15	11	CBBBCBCACBBBCACCACBBBCACBBBBBC

environment, and the required sequence that object types must be delivered to the home area in order for the complex object to be constructed. These distributions were derived according to the supposition that if an environment contains multiple object types, there will be a requirement for controllers to specialize to different behaviors in order to efficiently accomplish the task.

Simple Environment Experiment Set: Both Multi-Agent ESP and CONE evolved teams that yielded comparable performance for all simple environments. This result was supported by an independent t-test [5] (P values are not presented due to space constraints). This indicates that environments containing only one object type are not appropriate for encouraging the evolution of behavioral specialization, and that an optimal team behavior in this experiment set is for all controllers to adopt a non-specialized behavior. That is, in the simple environment experiment set, there is no requirement for different controllers to converge to complementary behavioral specializations in order to accomplish the task. This result supports the hypothesis that CONE only evolves and uses behavioral specialization in tasks that require specialization.

Complex Environment Experiment Set: Figure 3 presents, for each complex environment, the average *number of atomic objects* delivered to the construction zone in the correct order, by Multi-Agent ESP and CONE evolved teams. Task performance results presented in figure 3, indicates that CONE evolved teams yield a significantly higher average task performance comparative to Multi-Agent ESP evolved teams in six out of the nine environments. This is supported by an independent t-test. In each complex environment, the fittest CONE evolved team consisted of multiple castes (robot sub-groups specialized to gathering and construction with either type A, B, or C objects). This result supports the hypothesis that CONE is appropriate for deriving behavioral specialization which leads to a higher collective behavior task performance (comparative to Multi-Agent ESP evolved teams) is achieved.

6 Conclusions

This paper described the application of the Multi-Agent ESP and CONE neuroevolution methods for the purpose of automating controller design in a team of simulated robots. CONE effectively facilitated specialization in the behaviors of the robots which (comparative to Multi-Agent ESP teams) lead to a higher task performance in a process in a gathering and collective construction task. This research suggests that the controller design process used by CONE is able to

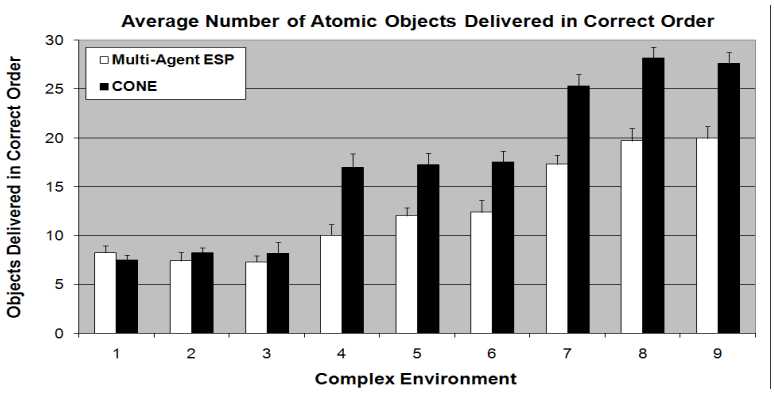

Fig. 3. Average Number of Objects Delivered in Correct Order to Home Area in each Complex Environment by Multi-Agent ESP and CONE evolved teams.

leverage and use emergent specialization in tasks that benefit from a behaviorally specialized problem solving approach.

References

1. G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behavior. *Artificial Life*, 9(1):255–267, 2003.
2. B. Bryant. *Evolving Visibly Intelligent Behavior for Embedded Game Agents. PhD thesis.* Computer Science Department. University of Texas, Austin, USA, 2006.
3. A. Eiben and J. Smith. *Introduction to Evolutionary Computing.* Springer-Verlag, Berlin, Germany, 2003.
4. J. Elman. Finding structure in time. *Cognitive Science*, 14(1):179 211, 1990.
5. B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, 1986.
6. D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
7. J. Gautrais, G. Theraulaz, J. Deneubourg, and C. Anderson. Emergent polyethism as a consequence of increased colony size in insect societies. *Journal of Theoretical Biology*, 215(1):363–373, 2002.
8. F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(1):317–342, 1997.
9. J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.
10. G. Nitschke. *Neuro-Evolution for Emergent Specialization in Collective Behavior Systems. PhD thesis.* Computer Science Department, Vrije Universiteit., Amsterdam, Netherlands, 2009.
11. C. Schultz and L. Parker. *Multi-robot Systems: From Swarms to Intelligent Automata.* Kluwer Academic Publishers, Washington DC, USA, 2002.
12. C. Yong and R. Miikkulainen. *Coevolution of Role-Based Cooperation in Multi-Agent Systems. Technical Report AI07-338.* Department of Computer Sciences. The University of Texas, Austin, USA, 2007.