Neuro-Evolution for a Gathering and Collective Construction Task

D. W. F. van Krevelen, G.S. Nitschke
Department of Computer Science
Vrije Universiteit, Amsterdam
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
krevelen@cs.vu.nl, nitschke@cs.vu.nl

ABSTRACT

In this paper we apply three Neuro-Evolution (NE) methods as controller design approaches in a collective behavior task. These NE methods are *Enforced Sub-Populations*, *Multi-Agent Enforced Sub-Populations*, and *Collective Neuro- Evolution*. In the collective behavior task, teams of simulated robots search an unexplored area for objects that are to be used in a collective construction task. Results indicate that the *Collective Neuro-Evolution* method, a cooperative co-evolutionary approach that allows for regulated recombination between genotype populations is appropriate for deriving artificial neural network controllers in a set of increasingly difficult collective behavior task scenarios.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Neuro-Evolution, Collective Behavior, Specialization

1. INTRODUCTION

Social biological systems such as termite hills and bee hives are well known for their success in accomplishing goals that could not be achieved by individuals. Decomposing labor into composite specializations and complementary behavioral roles increases global task performance in task environments such as collective gathering and construction [9]. Research in collective behavior systems (both simulated and physical) often attempts to replicate this success, and benefits of emergent specialization have been highlighted in artificial life [21], multi-agent computer games [8] and multi-agent systems [10]. As a means of solving complex collective behavior tasks that require the evolution of complementary and often specialized behavioral roles, co-operative co-evolution has achieved particular success [25], [26], [27]. *Neuro-Evolution* (NE) has been successfully applied to complex collective behavior control tasks for which there is no clear mapping between sensory inputs and motor outputs and that neither pure evolutionary nor pure neural computation methods address effectively [11]. In this research we compare the task performance of three such approaches, *Enforced Sub-Populations* (ESP) [11, 5], *Multi-agent Enforced Sub-Populations* (MESP) [21] and *Collective Neuro-Evolution* (CONE) [16], in a collective behavior task where specialization is beneficial. In the *Gathering and Collective Construction* (GACC) task, simulated robots search for, and carry simple objects towards a drop zone located on a hill. The task requires that (i) objects must be returned in a particular sequence, that is, to construct one complex object, and (ii) it takes at least two robots to transport objects uphill and deliver them to the drop-zone. The number of objects delivered in correct sequence and the total distance that objects travel towards the drop zone determine the team's fitness. The main research objective of this paper is to test the following hypotheses.

1.1 Research Hypotheses

1. Agent (controller) specialization improves collective behavior task performance in the GACC task.
2. CONE yields a higher collective behavior task performance comparative to similar methods used for evolving controllers in to accomplish the GACC task.

1.2 Task Performance Measures

Besides looking at the best fitness and difficulty level during an evolutionary run, the number of neurons in the networks and the number of burst mutations to handle stagnation, we also look at two derived values that measure specialization: *Action Entropy* and *Degree of Specialization*.

1.2.1 Action Entropy

To measure specialization, we apply the measure for information entropy defined by [2]. The *Action Entropy* measure measures an agent's specialization by means of the (un)predictability of its actions during a lifetime. Low information entropy means an agent's choice of action is very predictable as it specializes in a particular action, whereas agents choosing their actions equally often are highly unpredictable (carrying more entropy) and thus not specialized.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

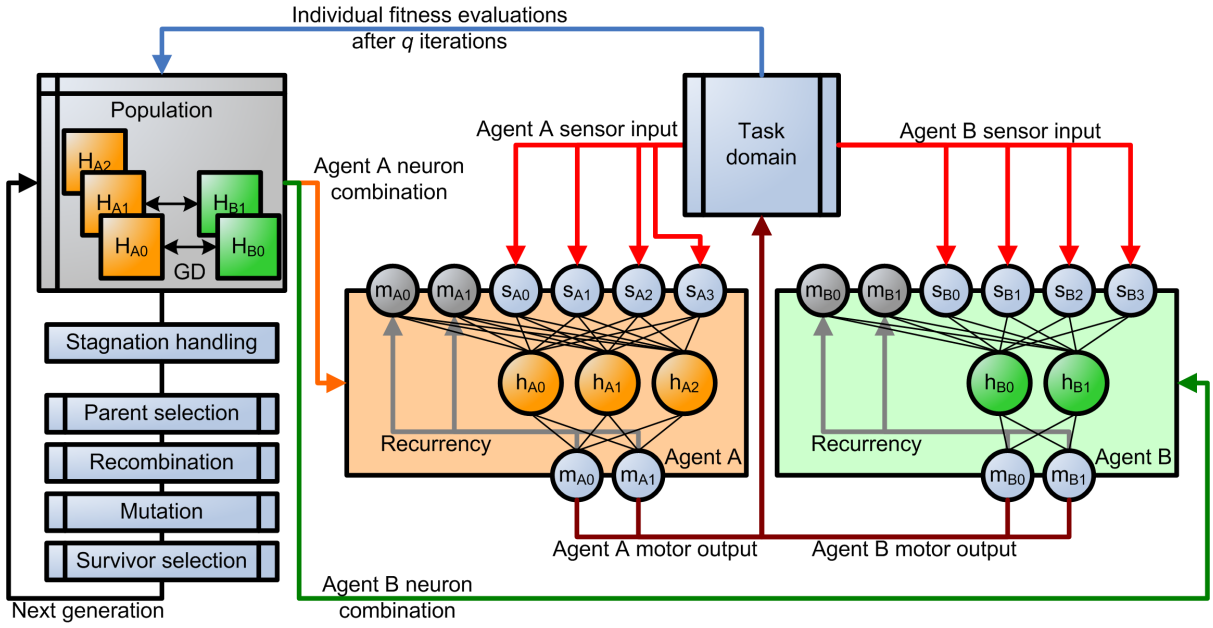

Figure 1: Multi-agent Neuro-Evolution with Enforced Sub-Populations. The evolutionary process (left loop) optimizes individual neurons h_{ij} within designated subpopulations H_{ij}. Different neuron combinations form the agents' semi-recurrent controllers to perform in the task domain for q iterations (right loop) and so evaluate their fitness to direct the search. Only CONE employs the Genetic Distance (GD) metric between subpopulations for inter-population recombination.

Given a collection of robot controllers $r \in \mathcal{R}$ and a set of (mutually exclusive) specialist actions $a \in \mathcal{A}$, we define the *action entropy* E averaged for a particular set of evaluations $v \in \mathcal{V}$ as:

$$E = \frac{1}{|\mathcal{V}|}\frac{1}{|\mathcal{R}|}\sum_{v\in\mathcal{V}}\sum_{r\in\mathcal{R}} -\sum_{a\in\mathcal{A}} p_{v,r,a}\cdot \log_{|\mathcal{A}|} p_{v,r,a} \qquad (1)$$

Action Entropy is averaged over all agents and all evaluations per generation, so a drop in action entropy means that all teams evaluated over several generations contained more specialists overall.

1.2.2 Degree of Specialization

A bit more complex than *Action Entropy* is the *Degree of Specialization* measure which focuses on the action an agent performed most and divides its duration by the amount the agent switched between actions. We define the *degree of specialization* S as:

$$S = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \frac{\max_m(t_{r,m})}{\max(1, s_{r,m})} \qquad (2)$$

where $t_{r,m}$ is the average frequency of the specialty action that agent i performed most often in all evaluations of generation m and $s_{r,m}$ is the average number of times agent r switched between specialty actions in all evaluations of generation m. Intuitively, $S \in [0, 1]$ denotes the degree of specialization. If agents spend more time on particular specialist action without switching a lot, the S value is close to one, whereas agents that share time between different actions and they regularly switch back and forth, the S value is closer to zero.

2. CONTROLLER NEURO-EVOLUTION

We are concerned with NE methods that evolve connection weights in *Artificial Neural Network* (ANN) controllers. The NE methods used for controller design that are compared in this research descend from the *Symbiotic Adaptive Neuro-Evolution* (SANE) method [15]. SANE evolves a population of neurons from which an ANN with a single hidden layer is constructed. However, assigning each neuron its own genotype subpopulation as shown in Figure 1 is beneficial and such an NE method has many applications [11, 21, 5]. Our comparison includes the *Enforced Sub-Populations* or ESP method [11, 5] that first applied this approach of separating genotype subpopulations. The second, multi-agent variant we call MESP [21] appeared later and *co-evolves* a team of networks so they learn to cooperate. Cooperative co-evolution is shown to outperform parallel evolution, producing more complex collective behaviors such that ANN controllers complement rather than obstruct each other [1, 21]. Going one step further, the recently developed *Collective Neuro-Evolution* (CONE) method [24] introduces gene exchange between neuron subpopulations belonging to different ANN controllers. CONE allows controllers to share beneficial and similar genotypes and facilitates the specialization of complementary behavioral roles.The ESP, MESP, and CONE methods share a common genotype representation and a common fitness function. Furthermore, the methods share some parts of their stagnation handling and breeding process in order to facilitate a fair comparison. Common features are listed below[1]:

[1]Implementations use the *Evolutionary Computation for Java* or ECJ toolkit [13].

Generations (G)	250	GDM-adapt interval* (V)	5	Iterations (q)	4000
Evaluation rounds	10	GDM-adapt fraction*	.1	Environment size	80.0^2
Evaluation trials	2	Initial GDM range*	.3	Scan\grab range	24.0\3.0
Subpopulation size (P)	10	Lesion interval (W)	10	Slope\drop range	12.0\6.0
Elite fraction	.25	Lesion threshold	.9	Scan slices (c)	8
Parent fraction (f_p)	.25	Fall-back interval* (Z)	20	Look-ahead (l)	3
Gene/weight range (w_{max})	$\pm$10.0	Fall-back rate	.4	Objects ($\mathcal{O}$) per type ($\mathcal{T}$)	5\4\3
Mutation rate (p_m)	.05	Agent count (n)	3	Agent type preferences	0\1\2
Mutation range	$\pm$1.0	Initial neuron count (m)	10	Preference rate (p_a)	.7
Cauchy scale	.3	Neuron weights (w)	53	ANN learning rate, momentum	{.5; .0}

$\star$ = *CONE-specific*

Table 1: ESP, CONE and GACC task parameters.

Genotype Encoding. Strings of real numbers form genotypes that correspond to a hidden layer neuron's input and output weights. In this study, genes may vary within a *fixed gene/weight range* (see Table 1) rather than have a free weight range as is the case in [5].

Fitness Evaluation. Each generation the sub-populations are shuffled so each neuron participates in an evaluated ANN, rather than just picking random neurons as in [5] which could leave some neurons unevaluated. This is repeated a given number of rounds so as neurons are evaluated in several combinations and each ANN is evaluated a given number of trials (see Table 1). A neuron's final fitness is the average over all trials' performances of each combination in which the neuron was combined.

Breeding Process. Shown in Figure 1 (left loop) the methods follow a general procedure for evolutionary algorithms [3]. After initializing the population parent genotypes are selected and recombined to produce offspring, which are then mutated before they become part of the next generation together with any other selected survivors. The compared methods share the following.

- A common *mutation operator*, adding noise from a *Cauchy-Lorentz* distribution in a fixed *noise range* with a fixed *mutation probability* (see Table 1). This follows methods described by [5], with the exception that mutation operates with given probability on each gene rather than on exactly one gene.
- *Survivor selection*, where all methods keep the same top or elite fraction (see Table 1) of each sub-population in the next generation.

Stagnation Handling. Shown in Figure 1 (left loop) the selected methods all extend the general breeding procedure [3] applying following stagnation handling measures with intervals listed in Table 1:

- A common delta-coding *burst-mutation operator* that resets a sub-population and generates new individuals randomly around the current best solution based on a *Cauchy-Lorentz* distribution as in [7].
- A common *lesion mechanism* similar to [**?**] that removes neurons (together with their designated subpopulations) if removal does not affect the best team's performance to below a certain threshold, or add a neuron if none were removed.
- A *fall-back rate* to reduce the score-to-beat when stagnation handling fails. In this study the fall-back rate is in the range $[0, 1]$. The purpose is to keep the best solution found thus far until the reduced score is improved. In [5] the scores to beat are reset, often causing the propagation of inferior champion genotypes.

Although the selected methods share common ancestry, each has distinct features regarding fitness evaluation, stagnation handling, and the breeding process.

Cooperative Co-evolution. Unlike in ESP, the fitness evaluation in MESP and CONE is based on team performance rather than individual performance. This increases the search space exponentially. For instance, finding the best network of 10 neurons connecting to 53 input/output weights requires searching $10 \times 53 = 530$ dimensions. Evaluating 3 controllers in parallel as ESP does, reduces search time for a single successful controller, while finding a successful team of 3 such controllers that work cooperatively as MESP and CONE do, means searching in $3 \times 530 = 1590$ dimensions thus increasing search time.

Crossover Operator. Whilst ESP and MESP employ single-point crossover [3], CONE treats the input weights separately from the output weights by applying triple-point crossover with the middle point fixed between input and output weights [16].

Parent Selection. The implementations of ESP and MESP select parents randomly from the top fraction (parent portion) of a sub-population in order to replace the non-elite that do not survive. This deviates from [5] where each candidate in the top quarter is combined with a random better candidate such that their offspring replace the bottom half of the subpopulation.

In CONE a *Genetic Distance* (GD) metric is used to determine if sub-populations for neurons of the same type but in different ANN controllers are close enough to combine their parent portions for parent selection. GD measures similarity in the potential parents' genes as a ratio $[0, 1]$ of the absolute weight range, where 0 is exactly similar and 1 is very dissimilar. Furthermore, a self-adapting *GD* value regulates the recombination between corresponding subpopulations in different controllers so as to facilitate specialized behavior. The GD metric and the CONE method are further described in [24] and are not elaborated upon here given space limitations.

Stagnation Handling. Following [5], ESP and MESP share a common strategy to handle stagnation:

- If for W generations an agent's performance did not improve, burst mutation is performed. This resets its sub-populations around the best corresponding neurons found so far. Burst-mutated sub-populations skip recombination as their fitness is not yet evaluated.
- After $2{\cdot}W$ generations of stagnation (i) ANN controller size is adapted by adding/removing neurons and their corresponding sub-populations using a lesion mechanism [**?**], (ii) burst-mutate the controller's subpopulations upon neuron removal, and (iii) multiply the best scores found by the *fall-back* rate.

CONE however adopts another approach:

- After V generations of stagnation the *GD range* adapts to stabilize inter-population merging of parent portions;
- After $W > V$ generations the lesion mechanism is applied (as in ESP the controller's sub-populations burst-mutate upon removal).
- After $Z > W$ generations all controllers are burst-mutated and the best scores multiplied by the *fall-back* rate.

3. COLLECTIVE BEHAVIOR TASK

We test the performance of ESP, MESP and CONE in a collective behavior task[2] that requires collective behavior to complete. Robots are placed in a square environment and start around a central drop point. Their task is to search and collect objects $\mathcal{O}$ of different types $\mathcal{A}$ placed at random in an unexplored environment before returning and dropping them again within a certain range of the drop point. As shown in Figure 2, the circular drop-zone resides on a hill which robots can navigate when empty. However, carrying objects uphill and onto the drop-zone requires the cooperation of another robot to push or pull them up the slope.

The objects $o \in \mathcal{O}$ must be gathered in a particular order as defined by the current scenario. The order is a sequence of object types $a \in \mathcal{A}$. Type 0 is distributed around the drop point while all other types > 0 are distributed in the environment's corners. The reward $r \in [0, 1]$ for moving and delivering objects is calculated as the distance objects are transported towards the drop-zone, multiplied by the number of (in order) deliveries $\mathcal{O}_{delivered} \subset \mathcal{O}$. That is:

$$r = \frac{\sum_{o \in \mathcal{O}} d(o)}{\sum_{o \in \mathcal{O}} d_{\max}(o)} \times \frac{|\mathcal{O}_{delivered}| + 1}{|\mathcal{O}| + 1} \qquad (3)$$

Where $d(o)$ denotes the distance between an object and the drop-zone's edge. To prevent rewards outside the $[0, 1]$ range when an object is moved away from the drop-zone its maximum distance $d_{\max}(o)$ is updated when necessary.

Shaping or incremental evolution is applied to the task of deriving controllers that collectively deliver objects in the correct order, since direct evolution of such controllers was unsuccessful. Similar to other NE research [11, 21, 5] the task is decomposed into several difficulty levels which the NE method can switch between as desired, guiding it towards good solutions. The following conditions must be met to complete each level:

Level 0. Each robot gathers one object of any type, in an environment without obstacles.

Level 1. Each robot gathers one object of each type, in an environment without obstacles.

Level 2. Each robot gathers one object of each type, in an environment with obstacles.

Level 3. The team gathers half the object sequence in order, in an environment with obstacles.

Level 4. The team gathers the entire sequence in order, in an environment with obstacles.

Each ANN controller evolves a behavior with the following sensors and motors:

- 1 *Sensor* indicating whether the robot has cargo.
- 2 *Sensors* for the drop zone's relative range and bearing.
- 2 *Sensors* indicating if and how long to wait for help.
- 2 *Sensors* indicating if and where assistance is required.
- $l \times |\mathcal{A}|$ *Sensors* indicating demand per atomic object type with length l look-ahead.
- $2 \times |\mathcal{A}|$ *Sensors* indicating the nearest object's range and bearing for each object type.
- s *Sensors* indicating obstacles per scan direction.
- m *Sensors* for receiving previous motor output.
- 7 *Motors* to move ahead/reverse and left/straight/right or to halt;
- $1 + |\mathcal{A}|$ *Motors* to grab the nearest object (for each type) a or to drop/assist.

For environments with $|\mathcal{A}| = 3$ object types, a look-ahead of $l = 3$ and $s = 8$ scan directions, robots must appropriately activate $m = 7{+}1{+}|\mathcal{A}| = 11$ motor outputs using input from $1{+}2{+}2{+}2{+}l{\times}|\mathcal{A}|{+}2{\times}|\mathcal{A}|{+}s{+}m = 41$ sensors. This means that each hidden neuron's inputs, outputs and bias requires the NE method to learn an extra $41 + 11 + 1 = 53$ weights per neuron. For controllers with 10 hidden-layer neurons this means an NE algorithm has to optimize weight starting points in $53 \times 10 = 530$ dimensions per controller. Co-evolving a team of three such agents to behave cooperatively means optimizing weight starting points in $530 \times 3 = 1590$ dimensions.

To control these sensors and motors, robots use a single-layer ANN with sigmoidal units, illustrated in Figure 1 (right loops). ANN controllers learn during a trial by adapting perceptron weights each iteration using the Back Propagation algorithm[3] [17, 19] with a specific learning rate and momentum (see Table 1) based on the following heuristics.

[2]The GACC task is implemented with the Multi-Agent Simulation (MASON) toolkit [12], including its 2D physics engine for collision handling.

[3]ANN controllers and BackPropagation are implemented with the Weka toolkit for machine learning algorithms [20].

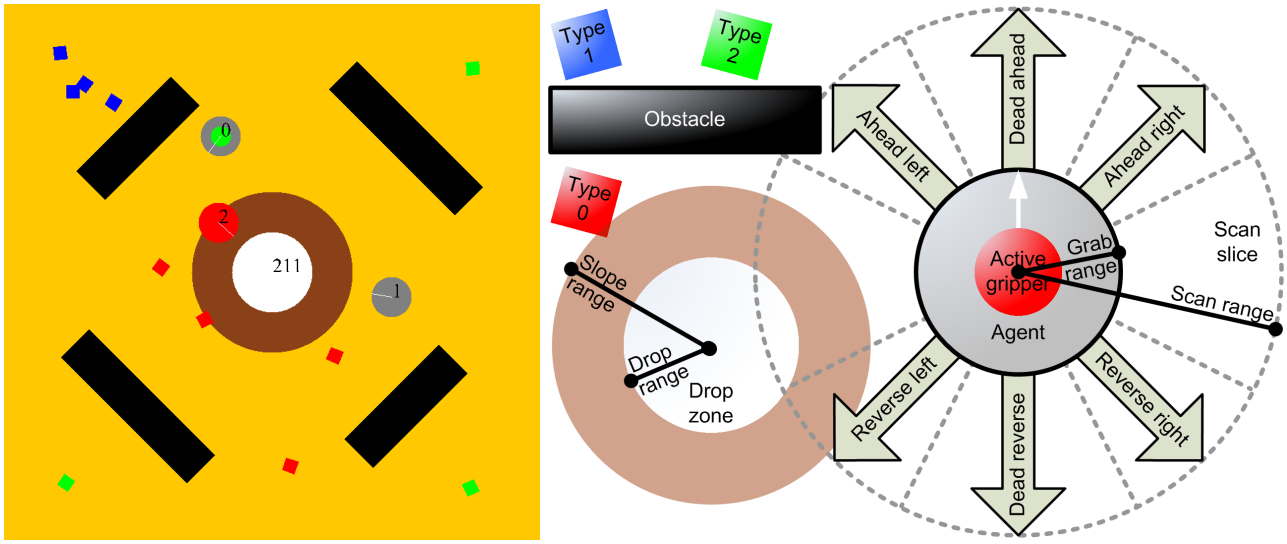

Figure 2: Example environment (left) and definitions (right). The white drop-zone on the brown hill shows the current sequence look-ahead, indicating that atomic object type 2 (green) is required first, then type 1 (blue) and again type 1. Currently robot 2 is carrying an object of type 0 (red) and awaiting assistance from robot 1 (grey) to push/pull it up the brown slope. Agent 0 is searching for objects for its type 2 gripper (green dot) while avoiding black obstacles.

1. If assist alert → move to assist location and drop/assist.
2. If empty, no alert, and selected object type is not in demand → select new object type.
3. If empty, no alert, and selected object type not in sight → explore.
4. If empty, no alert, and selected object type in sight → approach object.
5. If cargo, no alert, and not in drop-zone → move to drop-zone.
6. If cargo, no alert, in drop-zone, and cargo in demand → halt and drop cargo.
7. If cargo, no alert, in drop-zone, and cargo not in demand → leave drop-zone.

4. EXPERIMENTS

Robot controllers learn and evolve via one of three experimental design approaches: reactive, adaptive, or Lamarckian. *Reactive* means that the robots' ANN controller weights remain fixed during a trial. This produces reactive behavior so evolution has to do all the work of finding weights suitable for all situations. *Adaptive* means the robots update their ANN controller weights during a trial. With adaptive behaviors the evolutionary process is finding successful starting points for lifetime learning of the heuristics states (section 3). *Lamarckian* refers to the possibility of passing on lifetime experience gained through genes [22, 23]. This last approach means that adapted weights in a given controller are copied back to their genotypes at the end of a trial before crossover and mutation takes place. NE method comparisons included a *dummy* controller as a performance *benchmark*. That is, the dummy controller performed different actions with varying degrees of probability. That is, a dummy controller selects a random move/halt action with equal probability together with a random grab/drop action with $p = 0.7$ for the preferred object type's grab state and $p = 0.1$ for the other grab states.

4.1 Experimental Results and Analysis

The ESP, MESP and CONE methods were applied together with each of the three learning setups (*Reactive*, *Adaptive* and *Lamarckian*), plus the dummy setup, in order to evolve robot team controllers for accomplishing the GACC task. The parameter values (see Table 1) used were determined experimentally and found to be effective for the scenarios that we tested. Table 3 presents results yielded after 250 generations, averaged over 10 runs. Using time series data from all runs for each setup, we perform regression trend analysis and determine the Pearson product-moment correlation coefficients [14] to decide whether to accept or reject each of the hypotheses formulated in 1. We assume the correlation variables over all time points to be bivariate normal distributed as required [4].

4.1.1 Specialization and GACC Task Performance

Hypothesis 1 states that agent specialization improves collective task performance in the GACC task. This should be reflected by coefficients of correlation between collective task performance (*I. Best Fitness* or *II. Difficulty Level*) and agent specialization (*III. Action Entropy* or *IV. Degree of Specialization*). Correlations I/III and II/III are expected to be negative as performance should drop when action unpredictability (III) rises, whereas correlations I/IV and II/IV are expected to be positive as performance should increase when specialization (IV) also increases.

Table 2 summarizes the relevant performance/specialization correlation coefficients for the range $1 \leq G \leq 250$. Their significance depends on the number of data points, which is one for each generation in each runs, or $G \times N$.

Correlation pair	*I/III*	*II/III*	*I/IV*	*II/IV*
Expected	(−)	(−)	(+)	(+)
Reactive				
A. ESP	-0.10	-0.33^{**}	-0.23^{**}	-0.05
C. MESP	-0.73^{**}	-0.29^{**}	0.73^{**}	0.35^{**}
D. CONE	-0.47^{**}	-0.39^{**}	0.26^{**}	0.62^{**}
Adaptive				
A. ESP	-0.30^{**}	-0.36^{**}	-0.55^{**}	-0.52^{**}
C. MESP	-0.53^{**}	-0.67^{**}	0.46^{**}	0.43^{**}
D. CONE	-0.52^{**}	-0.59^{**}	0.46^{**}	0.49^{**}
Lamarckian				
A. ESP	0.44^{**}	0.29^{**}	-0.83^{**}	-0.73^{**}
C. MESP	0.28^{**}	0.57^{**}	-0.68^{**}	-0.88^{**}
D. CONE	0.37^{**}	0.74^{**}	-0.80^{**}	-0.85^{**}

$^{*}p < 0.05$ $^{**}p < 0.01$

Table 2: Performance/specialization correlation coefficients.

Method	*N*	*I. Best Fitness*	*II. Difficulty Level*	*III. Action Entropy*	*IV. Degree of Specialization*	*V. Neurons*	*VI. Burst Mutations*
Randomized							
NE Method NA	53	$3.4\% \pm 0.1\%$	1.0 ± 0.0	$0.0000G + 0.6783$	$-0.0000G + 0.0004$	$0.0268G + 9.6032$	$0.0664G - 1.0028$
Reactive							
A. ESP	11	$10.7\% \pm 3.5\%$	0.6 ± 0.2	$0.0012G + 0.1779$	$-0.0012G + 0.4131$	$0.0013G + 8.7279$	$0.0712G + 0.1905$
B. MESP	29	$7.5\% \pm 1.0\%$	0.1 ± 0.1	$0.0002G + 0.2713$	$0.0002G + 0.2017$	$-0.0112G + 9.2324$	$0.0850G - 0.3551$
C. CONE	27	$8.1\% \pm 0.7\%$	0.1 ± 0.1	$0.0003G + 0.2459$	$-0.0000G + 0.3390$	$-0.0048G + 5.0701$	$0.0877G + 0.0562$
Adaptive							
A. ESP	12	$14.9\% \pm 3.1\%$	1.0 ± 0.3	$0.0003G + 0.5038$	$-0.0001G + 0.1363$	$0.0008G + 9.3352$	$0.0679G + 0.2035$
B. MESP	14	$10.3\% \pm 2.0\%$	0.2 ± 0.1	$-0.0007G + 0.4569$	$0.0003G + 0.3174$	$0.0096G + 8.3262$	$0.0750G + 0.2297$
C. CONE	50	$13.0\% \pm 1.2\%$	0.6 ± 0.2	$0.0001G + 0.3183$	$-0.0003G + 0.4241$	$0.0066G + 5.5104$	$0.0775G - 0.0108$
Lamarckian							
A. ESP	12	$42.6\% \pm 11.0\%$	1.3 ± 0.3	$-0.0002G + 0.4565$	$-0.0001G + 0.1988$	$-0.0064G + 7.4195$	$0.0838G - 0.0820$
B. MESP	37	$31.3\% \pm 2.8\%$	2.4 ± 0.4	$0.0005G + 0.4688$	$-0.0005G + 0.1854$	$-0.0051G + 8.4171$	$0.0792G - 0.8925$
C. CONE	14	$33.3\% \pm 4.8\%$	3.5 ± 0.3	$0.0007G + 0.5396$	$-0.0001G + 0.0420$	$0.0137G + 5.4645$	$0.0634G - 1.3839$

Table 3: Results averaged after $G = 250$ generations. NE Method NA: Neuro-Evolution Method Not Applicable

Values must be above (or below) certain thresholds that are based on this amount to imply a significant correlation between two variables [4]. The * and ** indicate whether the correlations are significant at the 0.05 or 0.01 level respectively. These values must be interpreted carefully, since the proportion of variance is explained by the value's square [14]. A correlation of $r = 0.20$ means just $r^2 = 0.04$ or 4% of the variance is explained, whereas a correlation of $r = 0.80$ explains about $r^2 = 0.64$ or 64% of the variance.

According to the values in table 2, *Reactive* and *Adaptive* agents perform as expected, although in the ESP cases we see opposite correlations in *Degree of Specialization* (I/IV and II/IV). These ESP runs showed increased performance when agents became all-rounders, switching more (decreasing the *Degree of Specialization*) between fewer actions (decreasing *Action Entropy*). *Lamarckian* agents show results opposite from *Reactive* and *Adaptive* agents. Given these results, hypothesis 1 is accepted for *Reactive* and *Adaptive* cases, but hypothesis 1 is rejected for *Lamarckian* cases where agents perform significantly better when their actions are less predictable and less focused.

4.1.2 Performance of GACC Evolved Teams

The second hypothesis states that CONE outperforms similar methods in the GACC task. This means that CONE's results after $G = 250$ generations for *Best Fitness* and especially *Difficulty Level* should be significantly higher than those yielded by ESP or MESP. Table 3 shows that for *Reactive* and *Adaptive* agents CONE is outperformed only by ESP and performs about the same as MESP in either performance measure (see also figure 3). With *Adaptive* agents, CONE's advantage over MESP becomes more pronounced, but ESP with its small search space still outperforms CONE (see also Figure 4). For *Lamarckian* agents, CONE reaches a significantly higher *Difficulty Level* while *Best Fitness* is not significantly higher (see also Figure 5).

However, *Best Fitness* simply measures how far objects were moved (which does not take much intelligence, as exemplified by the *Randomized* runs), whereas increasing the *Difficulty Level* requires more complex behaviors. Hypothesis 2 is thus accepted for the *Lamarckian* cases.

Additionally, the following observations can be made regarding results yielded from each of the experimental setups.

1. In both reactive and adaptive learning situations ESP's parallel evolution significantly outperforms co-evolution by reaching higher fitness and difficulty levels, whereas in Lamarckian setups this advantage is lost and MESP and CONE reach similar difficulty levels and somewhat better team fitness.

2. The more CONE is assisted by heuristics, the lower the GD metric has to become to enable merging of parent genotype portions that are similar enough for inter-population recombination.

3. In reactive setups, the weights of different ANN controllers are very dissimilar given that inter-population recombination occurs only in the later generations.

4. In adaptive setups, there is more similarity between the weights of different ANN controllers yet more inter-population recombination is performed in order to counter fitness stagnation.

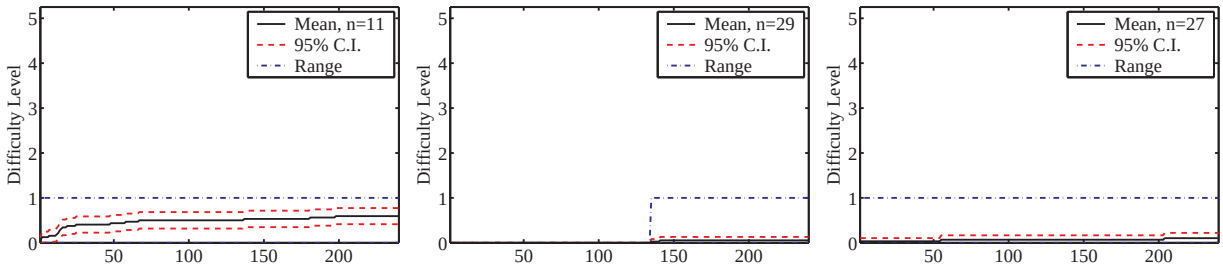

Figure 3: Results for* Reactive *agents, averages for first* $G = 250$ *generations.

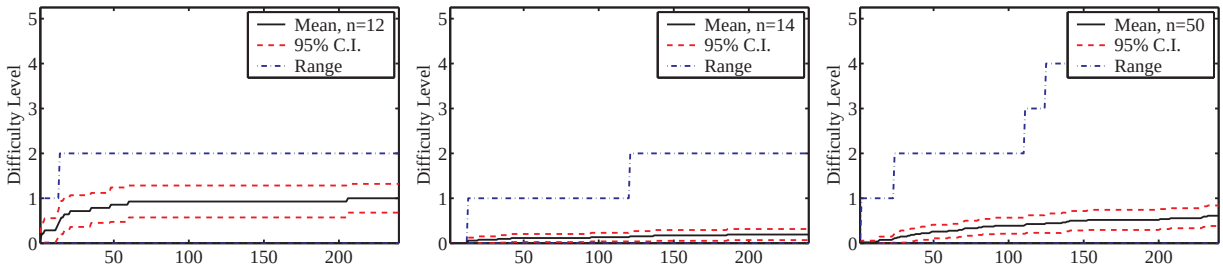

Figure 4: Results for* Adaptive *agents, averages for first* $G = 250$ *generations.

5. In the Lamarckian evolution setups, there is less fitness stagnation and inter-population recombination therefore occurs less frequently and at lower GD values as there is more similarity between the weights of different ANN controllers.

6. For task scenarios of a low difficulty level, such as the requirement that only one object be delivered in an environment with no obstacles, the ESP method is sufficient for controller derivation. In such simple task scenarios the co-operative co-evolutionary approaches of the CONE and MESP methods yield no advantage. However, for task scenarios of a higher difficulty level, such as the requirement that all objects be delivered cooperatively in a specific sequence, in an environment containing obstacles, the co-operative co-evolutionary approaches of MESP and CONE are appropriate.

5. CONCLUSIONS

Results indicate that the CONE method supported by heuristics derives a higher collective behavior task performance in the GACC task, comparative to that derived by the ESP and Multi-ESP methods. Furthermore, this task performance was able to effectively operate in tasks scenarios of a higher difficulty level. Also, results indicate that for simple task levels where no cooperative behavior is required, then the ESP method is sufficient, whilst for more difficult task scenarios that require coordinated (delivering objects in a specific order) and cooperative behavior (cooperative object transport), the cooperative co-evolutionary methods MESP and CONE are more appropriate.

6. REFERENCES

[1] M. Potter and K. De Jong Cooperative coevolution: Architecture for evolving coadapted subcomponents. *Evolutionary Computation*. 8(1): 1–29, 2000.

[2] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*. 27(1): 623–656, 1948.

[3] A. Eiben and J. Smith *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Germany. 2003.

[4] J. Bennett, W. Briggs, and M. Triola. *Statistical Reasoning for Everyday Life*. Addison-Wesley, New York, NY, USA, second edition, 3 Jul. 2002.

[5] F. Gomez. *Robust Non-linear Control through Neuroevolution*. PhD thesis, University of Texas, Austin, TX, USA, Aug. 2003.

[6] F. Gomez, D. Burger, and R. Miikkulainen. A neuroevolution method for dynamic resource allocation on a chip multiprocessor. In *IJCNN'01: Proceedings of the 2001 International Joint Conference on Neural Networks*, volume 4, pages 2355–2360. IEEE Computer Society Press, 2001.

[7] F. Gomez and R. Miikkulainen. Solving Non-Markovian Control Tasks with Neuroevolution. In *IJCAI'99: Proceedings of the 1999 International Joint Conference on Artificial Intelligence*, pages 1356–1361. Morgan Kaufmann, Stockholm, Sweden. 1999.

[8] K. Stanley and B. Bryant and R. Miikkulainen. Real-time Neuro-evolution in the NERO Video Game. *IEEE Transactions Evolutionary Computation*, 9(6):653–668, 2005.

[9] J. Gautrais and G. Theraulaz and 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.

[10] L. Li and A. Martinoli and A. Yaser. Learning and Measuring Specialization in Collaborative Swarm Systems. *Adaptive Behavior*, 12(3):199–212, 2004.

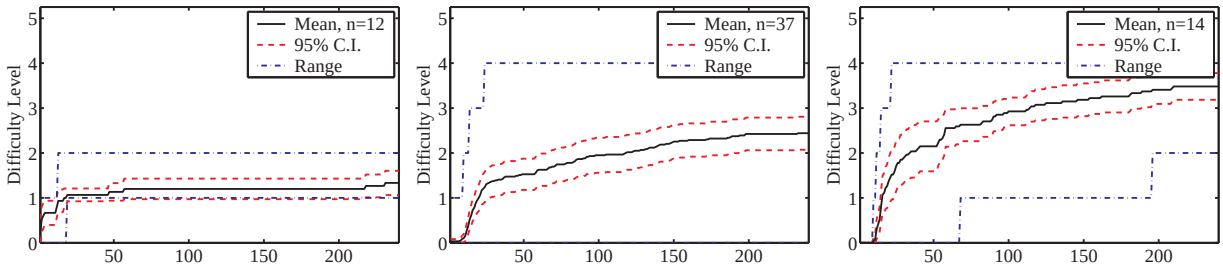

Figure 5: Results for* Lamarckian *agents, averages for first* $G = 250$ *generations.

[11] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behaviour*, 5(3-4):317–342, 1997.

[12] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. MASON: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[13] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Harrison, J. Bassett, R. Hubley, and A. Chircop. ECJ 16: A java-based evolutionary computation research system, 2008. `http://cs.gmu.edu/~eclab/projects/ecj/`

[14] R. McCall. *Fundamental Statistics for Behavioral Sciences*. Wadsworth Publishing, eighth edition, 2000.

[15] D. Moriarty and R. Miikkulainen. Evolutionary networks for value ordering in constraint satisfaction problems. Technical Report AI94-218, University of Texas, Austin, TX, USA, 1994.

[16] G. Nitschke, M. Schut, and A. Eiben. Collective specialization in multi-rover systems. *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, pages 342–342, 2007. ACM Press, London, United Kingdom.

[17] D. Rumelhart, G. Hinton, and R. Williams. *Learning internal representations by error propagation*, volume 1 of *Parallel Distributed Processing*. MIT Press, Cambridge, USA, 1986.

[18] A. Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. Methuen and Co., Ltd. 1904, London, United Kingdom, fifth edition, 1776.

[19] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, Cambridge, MA, USA, 1974.

[20] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Publishers Inc., San Francisco, 2nd edition, 2005. `http://www.cs.waikato.ac.nz/ml/weka/`.

[21] C. Yong and R. Miikkulainen. Cooperative coevolution of multi-agent systems. Technical Report AI01-287, University of Texas, Austin, TX, USA, 2001.

[22] B. Bryant and R. Miikkulainen. Acquiring Visibly Intelligent Behavior with Example-Guided Neuroevolution. *AAAI-07: Proceedings of the 2007 National Conference on Artificial Intelligence*, pages 801–808, 2007. AAAI Press, Menlo Park, USA.

[23] R. Miikkulainen. Evolving neural networks. *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, pages 3415–3434, 2007. ACM Press, London, United Kingdom.

[24] G. Nitschke and M. Schut Designing Multi-Rover Emergent Specialization. In *GECCO'08: Proceedings of the 2008 Genetic and Evolutionary Computation Conference*. 2008, ACM Press, Atlanta, USA.

[25] M. Potter and K. De Jong Evolving Neural Networks with Collaborative Species. In *Proceedings of the Summer Computer Simulation Conference*. pages 340–345, 2008, The Society of Computer Simulation.

[26] M. Potter and L. Meeden and A. Schultz Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists. In *IJCAI'01: Proceedings of the 2001 International Joint Conference on Artificial Intelligence*. pages 1337–1343, 2001, AAAI Press, USA.

[27] F. Gomez and R. Miikkulainen Active Guidance for a Finless Rocket Using Neuro-evolution. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*. pages 2084–2095, 2003, ACM Press, Chicago, USA.