Controlling Crowd Simulations using Neuro-Evolution

S. Wang
sunrisewng@gmail.com

J. Gain
jgain@cs.uct.ac.za

G.S. Nitschke
gnitschke@cs.uct.ac.za

Department of Computer Science
University of Cape Town
Cape Town, South Africa

ABSTRACT

Crowd simulations have become increasingly popular in films over the last decade, appearing in large crowd shots of many big name block-buster films. An important requirement for crowd simulations in films is that they should be directable both at a high and low level. As agent-based techniques allow for low-level directability and more believable crowds, they are typically used in this field. However, due to the bottom-up nature of these techniques, to achieve high level directability, agent-level parameters must be adjusted until the desired crowd behavior emerges.

As manually adjusting parameters is a time consuming and tedious process, this paper investigates a method for automating this, using Neuro-Evolution. To this end, the Conventional Neuro-Evolution (CNE), Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES), Neuro-Evolution of Augmenting Topologies (NEAT), and Enforced Sub Populations (ESP) algorithms are compared across a variety of representative crowd simulation scenarios. Overall, it was found that CMA-ES generally performs the best across the selected simulations.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Multiagent systems

Keywords

Entertainment and media, Simulation optimization, Genetic algorithms, Multi-agent systems, Neural networks

1. INTRODUCTION

The short film *Stanley and Stella in: Breaking the Ice*, shown at the Electronic Theater at SIGGRAPH '87, showcased the revolutionary idea that complex group behavior can be achieved by providing simple local rules to the agents within the group, as opposed to using some set of enforced global rules. Accompanying this short film was a technical paper detailing the technique used to simulate the flocking behavior [21].

Much work has since been done on Crowd Simulations. These can be roughly divided into two groups. *Agent-based* [29, 19] techniques aim to simulate behavior at an agent level, with the aggregate crowd behavior being emergent. *Group-based* [20, 24] techniques aim to simulate behavior at a group level, with the individual agent behaviors being chosen so as to adhere to the rules of the group.

Agent-based techniques have the benefit that the agent behaviors are much more heterogeneous, leading to more believable scenes. Additionally, the behaviors of individual agents within the crowd can be directed. Group-based techniques, on the other hand, have the advantage of being less computationally costly to simulate, allowing for both larger crowds and interactive simulation speeds. The aggregate behavior of crowds can also be directed much more easily than with agent-based methods, which require the adjustment of agent-level parameters in order to obtain the desired emergent group behavior [25].

Crowd simulations have become increasingly popular in films in the last decade, appearing in a plethora of movies such as *The Lord of the Rings Trilogy*, *World War Z*, *Ratatouille*, and *Inception*. Directors need these crowd simulations to be both controllable and believable [25]. Group-based techniques are not a viable choice as they often have overly homogeneous behaviors, which leads to less believability, and lack low-level directability since the simulation occurs at a group level. Thus, agent-based simulations are used. However, this requires artists to perform the time consuming and tedious work of adjusting agent parameters until the desired crowd behavior emerges.

In order to alleviate this burden, it is possible to automate the adjustment of these parameters. This can be achieved with various machine learning algorithms. We investigate one such technique in this context, namely *Neuro-Evolution* (NE), which uses *Evolutionary Algorithms* (EAs) to train *Artificial Neural Networks* (ANNs). We apply NE in order to derive crowd behaviors that accomplish given tasks. That is, agent controllers are ANNs adapted with EAs such that crowd behavior conforming to user specifications is evolved. NE has the advantage that it does not require a specific rule-set for agent behaviors, allowing for an easier authoring process.

Due to the variety of NE methods, and to the dearth of work that applies NE to achieve user specified goals in the context of crowd simulations designed for film [27], it is unknown which algorithm yields the best results.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754715

The contribution of this paper is to present a preliminary comparative study for various NE methods applied to representative crowd simulation tasks. The objective is to ascertain if there is a single NE method that is appropriate for automating crowd behavior design. Given the diversity of tasks tested, NE method efficacy is measured as the best achieved fitness after a set number of fitness evaluations. The NE methods investigated are Conventional Neuro-Evolution (CNE) [28], Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [11], Enforced Sub-populations (ESP) [10], and Neuro-Evolution of Augmenting Topologies (NEAT) [22]. These methods were chosen in order to cover a range of different types of algorithms. Our results show CMA-ES to be the best performing algorithm across our simulation scenarios.

2. BACKGROUND AND RELATED WORK

2.1 Controlling Crowd Simulations

Methods for controlling crowd simulations can be separated into two categories, high-level and low-level control [25]. Low-level control deals with directing individual agents within the scene. This is typically achieved by either modifying the variables associated with task or motor outputs [4], or by building up a state tree for an agent, and then traversing it to find the desired final state of the agent [9].

High-level control involves controlling the aggregate behavior of the crowd. Various group-based techniques have been successfully applied for such high-level control. For example, flow or potential fields can be used to steer agents towards desirable areas, whilst steering them away from undesirable areas [26, 6], whereas crowd configurations can be controlled through the traversal of motion graphs that depict group, instead of agent configurations [15]. Additionally, while not considered full crowd simulation systems, corridor-based path planning algorithms [13, 3] also represent a group-based approach to controlling group motion.

Agent-based techniques are also capable of providing high-level crowd control, typically by using an additional phase to learn the best parameters for achieving the desired behaviors. As an example, Anderson et al. [2] applied high-level control to bird-flock movement. This was achieved by performing a forward simulation of the flock, as well as a series of backward simulations that precisely satisfy the user provided constraints. The backward simulation most similar to the forward simulation was then chosen as the final output. In another method [12], the agent controllers are represented using fuzzy controllers, with the membership functions of these controllers being adjusted using Particle Swarm Optimization [14] in order to satisfy a fitness function representing the desired final crowd behavior.

Despite its success, the method of Anderson et al. [2] suffers from only being able to control the movement of crowds. Additionally, it only works with bird flocks as it uses a bird-flock specific *wander* element to measure the similarities between simulations. Jacka's method [12], whilst being much more general, is reliant on the artist constructing a well designed fuzzy controller structure in order for the training to be successful. NE is a promising alternative to these techniques as it is general, but also does not require the complex authoring process of fuzzy controllers, as users are merely required to specify the ANN structure, and the desired crowd behaviors.

2.2 Crowd Simulations in Neuro-Evolution

Crowd simulations fall into the large field of Multi-Agent Systems (MASs) [23], which have to take into account factors such as cooperation or competition between the agents, agent communication, and role specialization. These factors can be divided along two axes, namely *communicative versus non-communicative* and *homogeneous versus heterogeneous* [23].

There have been several studies of MASs in NE, such as Bryant and Miikkulainen's [5] work on role specialization of agents using homogeneous teams, which found that agents can learn to have adaptive role specialization dependent on their local context, and Yong and Miikkulainen's [30] work where ANN controllers learn to communicate via stigmergy, the act of implicitly communicating through perceived changes in the environment.

These works show that it is possible to use both homogeneous teams and stigmergy when dealing with communication and team composition in the design of our simulations. Both are desired as they allow for significant reductions in the problem dimensionality.

2.3 Neuro-Evolution Algorithms

NE methods can be divided up into three categories: single population fixed topology; cooperative co-evolution; and topology and weight evolving ANNs (TWEANNs) [7, 22, 18]. In order to test the viability of using NE to control crowd simulations, we select and compare a well-established algorithm from each category, in addition to *Conventional Neuro-Evolution* (CNE) [28], which is used as a benchmark to determine if the performance of more complex algorithms is significantly better.

CNE [28] is the simplest of the NE methods used, where a single population of fixed topology ANNs is evolved. These ANNs are represented as vectors of real numbers, with each real number corresponding to a weight within the ANN.

A different single population fixed topology approach is the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [11]. Much like CNE, CMA-ES also represents the ANNs as real valued vectors, however, it differs in the generation of offspring. Offspring are produced in CMA-ES through sampling a multi-variate distribution, with the covariance matrix, step size, and mean being iteratively updated so that the distribution covers high fitness areas of the search space.

The cooperative co-evolution algorithm selected is Enforced Sub-populations (ESP) [10]. In ESP, multiple sub-populations of real valued vectors representing neurons are evolved, as opposed to a single population of ANNs. In order to evaluate the fitness of these neurons, a neuron is selected from each sub-population to form an ANN, which can then get evaluated. The fitness obtained by this ANN is then shared across all participating neurons. Additionally, each neuron is evaluated multiple times in order to remove noise.

Neuro-Evolution of Augmenting Topologies (NEAT) [22] is the selected TWEANN. In NEAT, both the structures and the weights of a single population of ANNs are evolved. The evolution of the ANN structures is achieved by mutation operators that add connections or nodes, and the evolution of the weights which is achieved by mutating connection weights, and by reproduction, where the weight of the better performing parent is chosen. NEAT also makes use of complexification, which initializes the ANNs to their simplest

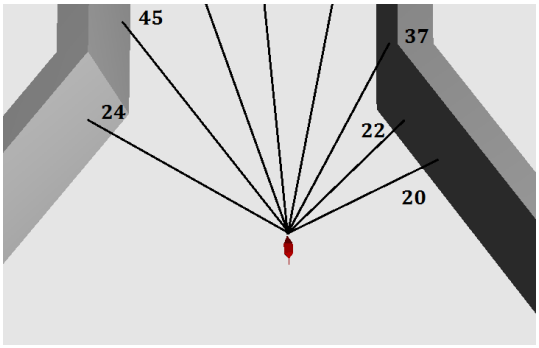

Figure 1: An agent's vision is simulated by casting rays, and returning the distance to the closest intersected object for each ray.

structure at the start of training, and speciation, which separates sufficiently different ANNs into different species, in order to protect structural innovations and minimize structure size.

All of the algorithms selected for comparisons are direct encoding methods [8], where the genotype has a direct mapping to the ANNs. A different approach would be to use generative encodings [8], where genotypes are interpreted using some set of rules in order to obtain the ANNs. Although comparing generative encodings lies outside the scope of this paper, it remains a promising direction for future investigation.

3. METHOD

Our system is divided up into three sub-systems: Neuro-Evolution, Simulation, and Rendering. The NE sub-system is responsible for loading the initial ANN structure and for running the NE methods, the simulation sub-system evaluates the fitnesses of candidate solutions sent from the NE sub-system within a given simulation scenario, and the rendering sub-system is responsible for visualizing a given simulation with a candidate solution and for loading 3D models.

Full results and source code used for this paper can be found in our online repository[1]. Additionally, videos of the various simulation scenarios can be found on our YouTube channel[2].

3.1 Controlling Agents

Each agent performs four steps every simulation cycle in order to update its behavior. It first *perceives* its local environment, using aspects such as vision (figure 1), agent velocity, and agent goals. The agent's ANN controller is then *evaluated* using its perception as input values. We use Fully connected Feed-Forward ANNs as our chosen ANN type as we found that they performed sufficiently, thus eliminating the need for using more complex ANN types. The output values of the ANN are then passed back to the agent, which then *interprets* them to determine what actions should be performed. Finally, the agent *acts* according to these instructions.

[1]https://bitbucket.org/igorawratu/ne-algorithm-comparison-paper

[2]https://www.youtube.com/channel/UCInMj1UK-1XXEo-RLidNrtw

3.2 Team composition

We use a homogeneous per species approach when assigning ANN controllers to agents, where all agents within a species use the same ANN controller. These species are used to represent different agent types, such as mice and robots, or different desired roles, such as defenders and attackers. The homogeneous teams approach is beneficial to crowd simulations as it allows for better search space scalability, as simulations can consist of tens to hundreds of thousands of agents. Species allow for simulations to have different agent types with differing inputs and outputs, and to have the same types of agents adopt different roles.

3.3 Fitness Evaluation

The fitness function of a given simulation is represented as a set of objectives, encoding the desired behaviors for the crowd. Examples of these objectives are reaching a physical goal location, or having a specific amount of agents alive at the end of a battle.

In order to evaluate a candidate solution, a simulation is run for a specified amount of time with the given solution, with the crowd's behavior being evaluated against the given objectives. After the simulation has completed, these objectives are then combined as a weighted sum in order to obtain the final fitness value. Fitness function weightings were obtained through exploratory experiments.

4. EXPERIMENT

Our crowd simulation testbed consists of eight different scenarios: Car Bridge Crossing; Car Race; Car Crash; War Robot Battle; Mouse Escape; Space-Ship Turnback; Space-Ship Obstacle; and Space-Ship Obstacle Field. Four different agent types were used in these scenarios, namely: Car, War-Robot, Mouse, and Space-Ship. This section elaborates on these scenarios and agents, as well as the parameters and experimental setup for our tests.

4.1 Agents

There are four different agent types used in our simulations. The **Car** agent is an agent that moves in a two-dimensional plane, with the ANN controller determining both its linear and angular acceleration. In order to simulate this agent's vision, eight rays are cast uniformly around the Car, thus simulating aspects such as rear and side-view mirrors.

Much like the Car agent, the **Mouse** agent's movement is also two-dimensional, with the ANN determining both its linear and angular acceleration. The Mouse agent also has the additional behavior that it can sense if there is an obstacle in front of it, and if so, it will come to a halt. The vision of this agent is simulated by casting eight rays in a cone in front of the Mouse.

The **War Robot** agent is another agent whose movement is on a two-dimensional plane, with the ANN controlling the linear and angular accelerations of the agent. The war-robot is also capable of sensing if there is an enemy in front of it, and if there is, it will fire and destroy it. The vision of this agent is similar to the Car agent, where rays are cast uniformly around the agent. However, it differs in that in addition to distance, the type of obstacle is also returned, with possible types being friendly, neutral, and enemy.

The **Space Ship** agent moves in three-dimensional space, with the ANN controller determining its x and y angular ac-

celeration (allowing it to rotate around both x and y axis), and its linear acceleration. Although it is typical for these types of agents to also rotate around the z axis, we eliminated this as we found that the agents evolved a visually unpleasant behavior, where they achieved the desired objectives very inefficiently by spiraling towards the goals. The vision of this agent is simulated by casting rays in a forward facing 5x5 grid.

In addition to vision, the agents are also provided with a variety of other inputs. However, as these inputs are simulation specific, they will be discussed in section 4.2. Additionally, all ANN controllers have six hidden nodes, which was found to work consistently well (we experimented with ANNs with 0 to 10 nodes). However, no hidden nodes were used for the initial ANN structure when using NEAT, to allow complexification.

4.2 Simulations

There are a total of eight different simulations used in our tests. While these simulations are not representative of all possible crowd simulation use cases in films, they do provide a reasonably diverse set with which to evaluate NE.

A set of common objectives were used across all of our simulations:

- D - The sum of the distances of all the agents to their final goal at the end of a simulation;
- A - Accumulated angular velocities of all agents throughout the simulation, used to eliminate oscillating agent movement;
- C - Total number of collisions occurring throughout the simulation;
- P - Difference between desired and actual population size of a crowd.

In addition to these, there are a couple of specific objectives, which are further explained within their respective simulation sections.

We also define a fitness threshold for each simulation. This threshold represents what a candidate solution's fitness should achieve in order to display acceptable behavior in the given simulation. These thresholds were obtained by viewing the simulations at various fitnesses, and then selecting a value close to an acceptable simulation's achieved fitness.

In the **Car Bridge Crossing (CBC)** simulation, ten Car agents starting in a wide area are tasked to cross a narrow bridge area within 10 seconds whilst avoiding collisions with both the environment and other agents. The Car agents, in addition to vision, also provide the ANN with their current position, the finish line that they must cross, current linear velocity, and current angular velocity. The fitness of a genotype is calculated using $f = 2D + 2C + A$. The acceptable fitness threshold of this simulation is 100.

In the **Car Race (CR)** simulation, ten Car agents are tasked to complete a race course within 13 seconds while avoiding collisions, where the agent at the back must comeback and win the race. The Car agents are provided with additional inputs in the form of position, the points defining the finishing line, velocity, the desired winner's position, and a flag specifying whether or not they are the desired winner. The fitness for this simulation is calculated using $f = C + D + 20W$ where W is the distance of the underdog to the actual winner of the race at the time the first Car crosses the finish line. The fitness threshold defined for this simulation is 250.

In the **Car Crash (CC)** simulation, two groups of 10 Cars each are initialized opposite to each other in a narrow corridor, with the aim being that they need to move to the opposite end of the corridor within 10 seconds, while avoiding collisions between both the environment and other Cars. Additional inputs for an agent include its position, the finish line for its respective group, and its velocity. The fitness is calculated with $f = 5D + C$. The fitness threshold for this simulation is 250.

In the **War Robot Battle (WRB)** scenario, two groups of 40 War Robot agents are tasked to battle each other for 10 seconds. One group is initialized behind various buildings, whereas the other is initialized in an open field. The aim of this simulation is to have between nine and 11 agents left on each side at the end of the simulation. The additional inputs for the War Robot agents are position and velocity. The fitness for this simulation is calculated using $f = C + 20P$. The fitness threshold defined for this simulation is 150.

In the **Mouse Escape (ME)** simulation, thirty-five Mouse agents are tasked to escape from five robot agents, with the aim of there being between 11 and 13 Mouse agents escaping by the end of a 15 second simulation, with the rest of the mice having been killed by the robots. The Mouse agents are initialized on one side of the environment, behind various buildings which act as cover, and have to cross a finish line situated at the opposite end of the environment in order to count as having escaped. The Mouse agents have additional inputs of position, linear velocity, and the finishing line. The War Robot agents have additional inputs of position and velocity. The fitness of the simulation is calculated by using $f = 4D + 25P + C$. Fitnesses under 100 are regarded as acceptable for this simulation.

In the **Space Ship Turn back (SSTB)** simulation, 40 Space Ship agents are tasked to move towards a goal sphere, which is initialized behind their starting position, within 10 seconds. In order to enforce cohesion amongst the crowd, an extra goal sphere with a significantly larger radius is used. Although it is also possible to instead use the sum of the distances to the crowd centroid as an objective, we found using the extra goal spheres to be a much simpler method. The Space Ship agents are provided the additional inputs in the form of current agent position, position of the goal spheres, current linear velocity, and the magnitude of their angular velocity. The fitness for this simulation is calculated using $f = 10C + 2G_1 + 2G_2 + 0.2A$ where G_1 is the goal sphere and G_2 is the extra goal sphere used to enforce cohesion. The fitness threshold for this simulation is 750.

The **Space Ship Obstacle (SSO)** simulation is identical to the Space Ship turn back simulation, with the exception of the goal spheres and the environment. The goal spheres are placed in front of the agent, with a large obstacle in between the goal point and the agent starting positions. Additionally, the fitness threshold of this simulation is higher than that of the turnback simulation due to the presence of the extra obstacle, leading it to be defined at 1000.

The **Space Ship Obstacle Field (SSOF)** simulation is identical to the Space Ship obstacle simulation, with the exception of there being a field of small obstacles instead

Table 1: Mean fitness after 20,000 fitness evaluations achieved by the NE methods across the various simulations. Bolded fitness values achieved better values than the fitness threshold. Acceptable thresholds for each simulation can be found in section 4.2.

Simulation	NEAT	CMA-ES	CNE	ESP
CBC	**7.87**	**60.14**	**3.35**	**43.14**
CC	946.78	389.03	752.48	935.16
CR	401.85	**147.59**	**171.34**	265.68
ME	**17.82**	**5.64**	**1.70**	**7.75**
WRB	**72.09**	**74.39**	**90.2**	**108.72**
SSTB	**428.63**	**47.8**	**203.21**	**101.93**
SSO	1262.45	**773.99**	1273.87	2190.49
SSOF	**891.21**	**675.13**	**847.42**	1031.02

Table 2: Mann-Whitney U test p-values of the various algorithms when compared to the algorithm that achieved the lowest mean after 20,000 fitness evaluations in each simulation scenario. N/A is used to signify the algorithm with the lowest mean.

Simulation	NEAT	CMA-ES	CNE	ESP
CBC	9.5e-4	0.21	N-A	2.2e-3
CC	1.1e-10	N-A	3.8e-9	1.6e-11
CR	1.1e-10	N-A	0.47	6.6e-5
ME	2.4e-7	3.1e-6	N-A	4e-5
WRB	N-A	0.39	1.6e-4	7.7e-10
SSTB	6e-11	N-A	1.8e-8	2.5e-8
SSO	3.7e-10	N-A	6.2e-5	5.4e-10
SSOF	4.5e-8	N-A	2.8e-4	4.1e-7

of one large one. The fitness threshold of this simulation is defined at 1000.

4.3 Parameters and Operators

The operators used for our experiments are: Linear Rank-Based Selection for CNE, NEAT, and ESP; Blend crossover for CNE and ESP with $\alpha = 0.5$ and a crossover probability of 0.8; and Gaussian weight mutation for CNE, NEAT, and ESP, with a mutation probability of 0.2, and a mutation probability of 0.02 for CNE and ESP, and 0.8 for NEAT. Additionally, the add node and add connection mutation probabilities for NEAT were 0.03 and 0.05 respectively, and the compatibility threshold was set at 0.5. We used populations of 100 for CNE and NEAT, and sub-populations of 50 with each neuron being evaluated 5 times for ESP, with 5% elitism for CNE, ESP, and NEAT. We used an initial step size of 0.2 for CMA-ES. We run burst mutation on an ESP sub-population after it has stagnated, which we detect by seeing if it has not improved after 20 generations. These operators were determined through preliminary experiments.

4.4 Experimental Setup

Each of the four NE methods were run for 20,000 fitness evaluations 30 times each on every simulation, with the best fitness obtained by each algorithm being sampled every 10 fitness evaluations. We then plot the averages of the fitness values obtained, using the one-tailed Mann Whitney U test [16] to test if the distributions are significantly different using a confidence interval of 95% and a significance threshold of p=0.05.

5. RESULTS AND DISCUSSION

As shown in tables 1 and 2, CMA-ES was found to either perform the best, or perform comparably to the best algorithm. The one exception is the Mouse Escape simulation. However, this simulation proved very easy for all algorithms to learn, with the differences in mean fitness being negligible. One problem that we found with CMA-ES is that its computational cost far exceeds the other three algorithms on problems with higher dimensions, with thethe mean time taken to execute CMA-ES being between 2-4 times that of the other three[3] (table 3) on some of the simulations. This increased computational cost is mainly due to the Eigen-decomposition step of the algorithm, which is

[3]Processor: i5-3570 @ 3.4GHz

Table 3: Mean execution times of the NE algorithms across the various simulation scenarios when evolved for 20,000 fitness evaluations.

Simulation	NEAT	CMA-ES	CNE	ESP
CBC	280.1s	305.3s	231.3s	241.4s
CC	514.1s	602.5s	538.9s	531.1s
CR	385.8s	1402.1s	372.2s	356.4s
ME	1356.6s	2590.8s	955.6s	961.4s
WRB	2671.1s	5671.2s	2612.1s	2659.7s
SSTB	1086.4s	2201.4s	1013.4s	1017.7s
SSO	1062.5s	2185.8s	1021.3s	1000.1s
SSOF	1261.9s	2383.5s	1256.2s	1170.1s

$O(n^3)$. In order to verify this we used CMA-ES in a simulation where 4 agents were tasked to complete a race track within a given time limit, with the team setups being heterogeneous, 1 ANN for every 2 agents, and homogeneous. The execution times found for these team compositions were 3689.02s, 1398.85s, and 283.86s respectively.

CNE performed above expectations and achieved the best mean fitness for two out of eight of the simulations. It was also one the second best performing algorithm for four out of eight of the simulations. This high performance was mainly unexpected due to its simplicity, and to it achieving poorer results than the other algorithms in previous works [22, 10]. In addition to the algorithm's high task performance, it is also the least computationally intensive, leading to it being an attractive option for evolving ANN controllers to control crowd behaviors. One issue with CNE is that its task performance deteriorated on higher dimensional problems. This may lead to it performing poorly in more heterogeneous team setups, however, further investigation is required.

NEAT's task performance was average compared to the other algorithms, often being either the second or third best performer. One unique benefit to NEAT, however, is that it is not necessary to predetermine an effective ANN structure, as both structure and weights are evolved.

ESP was shown to be the worst performing algorithm in our simulations. On the other hand, as shown in figure 2, ESP is often still improving at the end of our preset 20,000 fitness evaluation limit. Thus it may be possible that it will perform significantly better given longer evolution run times. This is most likely due to our use of sub-population sizes of 50 with five fitness evaluations per neuron, causing ESP to run for less than half the number of generations of

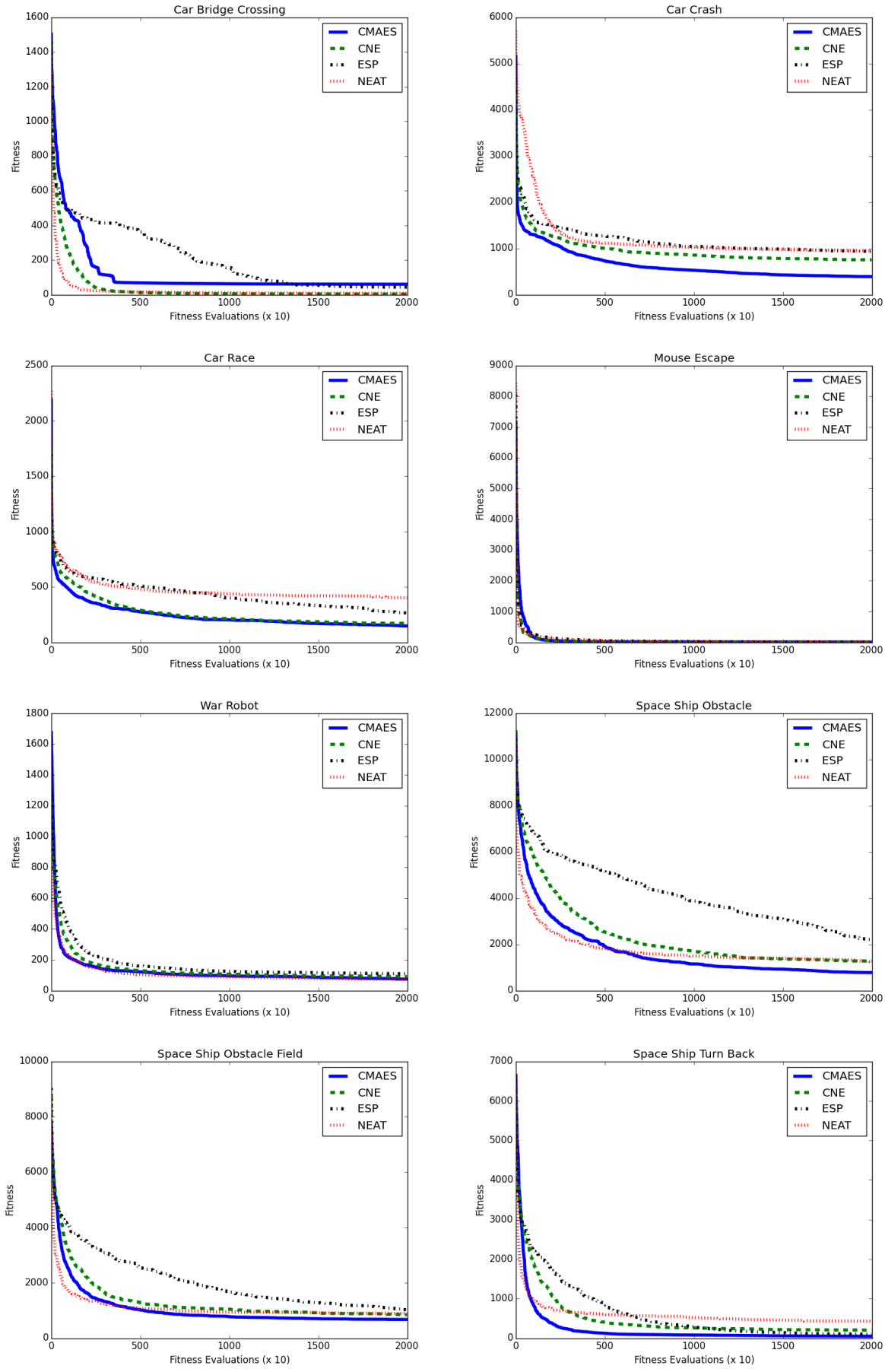

Figure 2: Mean fitness values of the NE methods across the various simulation scenarios using a population of 30 samples throughout 20,000 fitness evaluations for each simulation

NEAT and CNE, and less than 10 times the amount of generations of CMA-ES for the same number of fitness evaluations. Additionally, ESP's performance increased relative to the other algorithms on higher dimensional problems, such as the space-ship turn back scenario, possibly due to its cooperative nature. This increased performance makes ESP a worthwhile avenue of investigation when researching more heterogeneous controller setups.

Of the eight test simulations, we found that the Car Bridge Crossing, Mouse Escape, Space-Ship Turn Back, and War Robot Battle scenarios to be the easiest to evolve effective controllers for, with all the algorithms ultimately achieving acceptable final crowd behaviors. Scenarios of moderate difficulty were the Space-Ship Obstacle, Space-Ship Obstacle Field, and the Car Race simulation as only 1-3 algorithms (depending on the actual simulation) were able to achieve the target fitness thresholds for those scenarios.

The one simulation that all algorithms struggled with is the Car Crash scenario, with none of the algorithms able to achieve a mean fitness below the threshold. The difficulty of learning this simulation is due to the close proximity of the agents, as well as the extremely narrow corridor, resulting in the agents having to evolve near perfect movement in order to avoid collisions. One way to deal with this is to have an ANN controller for each group as opposed to a single ANN controller for both groups, which would allow each group of agents to adopt different avoidance strategies.

In addition to dealing with the Car Crash scenario, investigating more heterogeneous controller setups is also desirable for generating more diverse behaviors amongst agents, as we found that in certain simulations with sparse environments, such as the War Robot Battle, agents tend to adopt very homogeneous behaviors. This is undesirable for films as it leads to the crowds being less believable. This issue can be attributed to many of the ANN inputs being the same, resulting in identical outputs for multiple agents. We hypothesize that using more heterogeneous controller setups may result in more heterogeneous behaviors, leading to more believable scenes. Additionally, this would also allow for the use of pareto-based multi-objective optimization to evolve differing but optimal agents [1]. This is achieved by selecting sufficiently different candidate solutions on the pareto-front as agent controllers for the simulation.

A different approach to achieve more heterogeneous behavior is to instead employ a hierarchical model [20, 17], where an agent's high level goals and cognition are separated from their low level motor functionalities. The ANN would then control the cognitive-level functions of the agents, as opposed to the direct motor behaviors as in our current simulations. This approach may allow for more complex agent behaviors, whilst keeping the problem dimensionality low. However, the additional behavioral layers could result in the crowds being more difficult to control.

6. CONCLUSIONS AND FUTURE WORK

This paper evaluated the task performance of Conventional Neuro-Evolution (CNE), Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES), Neuro-Evolution of Augmenting Topologies (NEAT), and Enforced Sub-populations (ESP) in the domain of controlling emergent crowd behaviors for film across eight different simulation scenarios. Overall, we found that CMA-ES achieves the best task performance. However, it comes at the cost of computational complexity, as it performs significantly slower than the other three algorithms on higher-dimensional scenarios. For this reason, it may be desirable to instead use CNE, which combines good task performance and cheap computation cost. Nonetheless, we found that CNE's mean fitness is poorer in our higher dimensional scenarios relative to lower-dimensional ones. Thus, further investigation is required in order to determine how well it performs on higher dimensionality problems.

Future directions of research include investigating more heterogeneous controller setups, exploring various generative encoding NE methods, performing user evaluations on the believability of the generated crowds, using pareto-based multi-objective optimization in order to generate more heterogeneous behaviors, and investigating the use of NE with more hierarchical agent behavior models.

7. ACKNOWLEDGMENTS

This research was funded by the National Research Foundation, the Centre for High Performance Computing and Triggerfish Animation Studios under THRIP Grant TP2011071600004.

8. REFERENCES

[1] A. Agapitos, J. Togelius, S. M. Lucas, J. Schmidhuber, and A. Konstantinidis. Generating diverse opponents with multiobjective evolution. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games, 2008. CIG'08.*, pages 135–142. IEEE, 2008.

[2] M. Anderson, E. McDaniel, and S. Chenney. Constrained animation of flocks. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 286–297. Eurographics Association, 2003.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, pages 104–113. IEEE, 2002.

[4] B. M. Blumberg and T. A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM, 1995.

[5] B. D. Bryant and R. Miikkulainen. Neuroevolution for adaptive teams. In *Proceedings of the 2003 congress on evolutionary computation*, volume 3, pages 2194–2201, 2003.

[6] S. Chenney. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 233–242. Eurographics Association, 2004.

[7] D. Dasgupta and D. R. McGregor. Designing application-specific neural networks using the structured genetic algorithm. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92.*, pages 87–96. IEEE, 1992.

[8] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[9] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 29–38. ACM Press/Addison-Wesley Publishing Co., 1999.

[10] F. J. Gomez and R. Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, volume 99, pages 1356–1361, 1999.

[11] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer, 2004.

[12] D. Jacka. *High-Level Control of Agent-based Crowds by means of General Constraints*. PhD thesis, University of Cape Town, 2009.

[13] A. Kamphuis and M. H. Overmars. Finding paths for coherent groups using clearance. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28. Eurographics Association, 2004.

[14] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[15] Y.-C. Lai, S. Chenney, and S. Fan. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 281–290. ACM, 2005.

[16] H. B. Mann, D. R. Whitney, et al. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60, 1947.

[17] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. Eurographics Association, 2007.

[18] J. C. F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence*, 8(1):73–84, 1998.

[19] F. Qiu and X. Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190–205, 2010.

[20] S. Raupp Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164, 2001.

[21] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 25–34, New York, NY, USA, 1987. ACM.

[22] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[23] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[24] D. Thalmann, H. Grillon, J. Maim, and B. Yersin. Challenges in crowd simulation. In *2009 International Conference On Cyberworlds*, number EPFL-CONF-159013, pages 1–12, 2009.

[25] D. Thalmann, C. Hery, S. Lippman, H. Ono, S. Regelous, and D. Sutton. Crowd and group animation. In *ACM SIGGRAPH 2004 course notes*, page 34. ACM, 2004.

[26] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168. ACM, 2006.

[27] S. Wang, J. Gain, and G. Nitschke. Comparing crossover operators in neuro-evolution with crowd simulations. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 2298–2305. IEEE, 2014.

[28] A. P. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks.*, volume 2, pages 667–673. IEEE, 1991.

[29] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin. Composite agents. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 39–47. Eurographics Association, 2008.

[30] C. H. Yong and R. Miikkulainen. Coevolution of role-based cooperation in multiagent systems. *IEEE Transactions on Autonomous Mental Development*, 1(3):170–186, 2009.