Investigating the robustness of multi-agent controllers evolved using HyperNEAT with objective, novelty and hybrid fitness functions to solve the collective construction problem

A. Ruben Putter
University of Cape Town
Rondebosch 7701
Cape Town, South Africa
Supervisor: Dr. Geoff Nitschke
PTTAND010@myuct.ac.za

ABSTRACT

This paper investigates the ability of objective, novelty and hybrid fitness functions to evolve multi-robot team controllers that are able to solve a collective construction task using the HyperNEAT evolutionary algorithm. These approaches are tested using tasks of increasing complexity and that require different levels of cooperation within a robot team. It was found that, although they produced relatively similar task performance results for the easier tasks, the controllers that were evolved with novelty fitness outperformed the other approaches in the most difficult task, further establishing its ability to deal with deceptive problem domains.

Keywords

HyperNEAT; Neuroevolution; Artificial Neural Networks; Evolutionary Algorithms; Evolutionary Robotics; MultiRobot Team; Collective Construction Task;

1. INTRODUCTION

1.1 Background

One of the central themes of research in mathematics and computer science is the design and development of automated problem solvers [7].

The current trend in increasing levels of computerisation coupled with the increasing non-linearity and complexity of tasks that need to be performed has made traditional controller design impractical [13], resulting in a growing demand for the automated design of these problem solvers [7].

The problem solving power of evolution is abundantly clear in nature with the diverse range of organisms that exist on Earth, each specially adapted to optimal survival within its own niche environment [7], which is why computer scientists have turned to natural processes for inspiration [13].

Evolutionary Computing (EC) is a branch of computer science research inspired by the Darwinian process of evolution through natural selection [7].

Neuroevolution (NE) provides an effective approach to solving complex optimization and control tasks [12].

One such complex task is that of collective construction, in which a team of agents must cooperate by coordinating their behaviour in order to assemble various structures within their environment [30].

The ability to automate the construction process provides several real-world advantages, such as the rapid production of low-cost housing and reducing the high accident rates caused by human error [31].

Automated construction would also be useful in future endeavours such as construction in areas that are too dangerous or inaccessible to humans [31]. This includes examples such as sending a team of robots to extraterrestrial locations to build habitats in anticipation of human travellers and underwater construction [30].

Multi-agent approaches to this task that use numerous simpler robots present several advantages over using a single more sophisticated robot, particularly with respect to parallelism, decentralization and robustness [31].

Some of these advantages include being able to use simple designed robots in the team instead of one very expensive robot as well as having increased fault tolerance since there is no longer just a single point of failure [1].

1.2 Problem Statement

Given that these multi-robot teams are intended to be used in treacherous and remote locations, the implemented controllers ideally need to be able to cope with certain unforeseen circumstances.

Specifically in this case, their ability to deal with a change in their input sensor configuration.

Since the controllers can't be trained from scratch to use the new sensor morphology and the structure of the ANN can not be altered, the existing controller must be robust in such a way as to be able to use its remaining sensors in order to complete the task at hand.

Similar research includes investigating the ability of NE methods to find a minimal sensory configuration such that a team of homogeneous robots are still able to complete a task requiring collective behaviour [28], as well as an investigation into evolving specialised sensor resolutions in a multi-agent task [22].

Another similar project includes the investigation into evolving an optimal configuration of input sensors such that a homogeneous team of robots can solve the collective construction task [29].

What makes this project different from existing research, is that it is specifically investigating the ability of Hyper-

NEAT evolved controllers to deal with the loss of input sensors and still be able to solve cooperative tasks of increasing levels of complexity.

The behavioural robustness of a controller will be tested by disabling a random set of input sensors for each homogeneous team and then comparing its task performance to that of a robot team with all the sensors in tact.

1.2.1 Research Question

Which implementation of the HyperNEAT algorithm is able to produce the most robust controllers in such a way that they are able to consistently solve a collective behaviour task of increasing complexity?

1.3 Research Aims

The aims of this research project can be summarised as follows:

- To investigate the behavioural robustness of multi-robot team controllers evolved using the HyperNEAT algorithm with respect to sensory configurations when implemented in solving increasingly complex tasks requiring collective behaviour.
- building on the first, the second aim is to investigate the three different variations of the HyperNEAT algorithm by implementing various fitness functions.

The complexity of the collective construction task is controlled by using a construction schema, which is implemented as a set of underlying connection rules.

The level of cooperation is controlled by using the different types of resource blocks in the simulation.

The various levels are outlined as follows:

- Level 1: no complexity and no cooperation
- Level 2: some complexity and no cooperation
- Level 3: no complexity and some cooperation
- Level 4: some complexity and some cooperation

These construction schemas and different block types are discussed in more detail later on in the paper.

2. LITERATURE REVIEW

2.1 Collective Construction Task

This project chose to investigate the collective construction task because it is sufficiently complex and requires cooperative behaviour. It is also easy to manipulate this complexity as well as the levels of cooperation required as outlined in the previous section.

The collective construction task requires that a team of agents cooperate by coordinating their behaviours in order to assemble various structures within their environment [23].

The general implementation of the collective construction task can be broken down into three separate sub-tasks:

First, the robot team needs to search as much of their environment as possible in order to locate the different types of resource blocks.

Second, once a robot has found a resource, it needs to be taken to the designated construction zone in order to connect it to the current structure. These first two steps can be seen as an implementation of the standard collective gathering task [23].

Finally, once the robot has brought the new resource to the construction zone, it needs to find an open side on the structure to which the new resource can be attached.

In order to perform the task successfully, the robots are required to explore their environment and search for the various randomly placed resources in the most efficient and effective way possible.

2.2 Neuroevolution

Neuroevolution (NE) provides a way of combining Evolutionary Algorithms (EAs) and Artifical Neural Networks (ANNs) [9].

Learning in ANNs, also referred to as training, is accomplished by iteratively adjusting the connection weights between neurons until the desired output is achieved [5].

NE is able to train an ANN to accomplish a specific task by using EAs to perform this connection weight adaptation [21].

The main advantages to using NE to design these controllers are that it works in continuous and partially observable environments [20], the designer does not need to specify how the task should be solved [13] and it does not require some value function or a corpus of training examples [12].

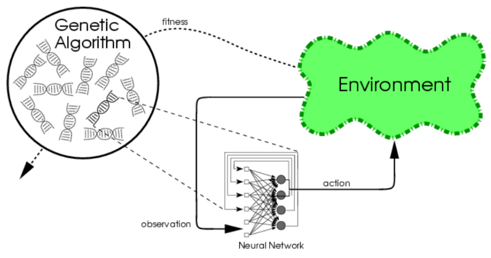

Figure 1: A basic overview of how neuroevolution works

NE searches for an ANN controller with optimal performance based on a fitness function of some kind that determines an ANN's suitability to performing a task [9].

The different NE methods can roughly be split into two categories according to their encoding method, either direct or indirect.

In direct encoding methods, each element in the genotype explicitly encodes an independent aspect of the phenotype such that there is a one-to-one mapping between the genotype and the phenotype [3, 25].

In indirect encoding (also called generative or developmental) encoding schemes, each element in the genotype is implicitly described by a computable function allowing for a much more compact representation of the genotype [25]. It also allows for genetic information to be reused [3].

This research project specifically investigates the HyperNEAT algorithm [24].

2.2.1 HyperNEAT

Hypercube-based Neuroevolution of Augnmenting Topologies (HyperNEAT) is an indirect encoding method that evolves both the connection weights and topology of ANNs [24]. It uses a generative encoding and results in modular, regular ANNs with greater learning capacities [26].

HyperNEAT is an extension of the NEAT evolutionary algorithm that works by evolving an indirect encoding of Compositional Pattern Producing Networks (CPPNs) instead of evolving the ANNs directly [24].

Each of these CPPNs is a computable function that produces an output based on an input and is used to define an ANN that is a candidate solution [2, 14].

The basic principle of CPPNs are found in nature, where it is possible to describe patterns as compositions of functions where each function represents a stage in development [4].

Research in generative and developmental encoding regularly shows that large structures can be compactly represented in DNA through the reuse of genetic material [24].

Indirect encodings such as the CPPNs allow for repeated structures in the genome to be represented by a single gene that gets reused in the process of mapping the genotype to the phenotype, making it a much more compact representation of a solution [24].

The HyperNEAT algorithm was chosen for this investigation due to its relative success in evolving successful multi-agent controllers for similar tasks such as RoboCup Soccer [27] and Pursuit Evasion [4].

It has been demonstrated that HyperNEAT is able to exploit regularity and modularity in multi-agent tasks in order to produce solutions that might otherwise not have been found [29].

It has also been shown to exploit intermediate problem regularity by producing regular ANNs which result in regular behaviour [3].

HyperNEAT is therefore potentially beneficial as structures that are to be built in collective construction tasks are modular (comprised of a set of blocks) and regular (the same sequence of blocks can be repeated) [15].

2.3 Fitness Functions

As part of the process of evolving an ANN controller, the performance of each candidate solution is evaluated and accordingly assigned a fitness value.

A good fitness function is meant to serve two different purposes, namely: defining the end goal of the algorithm and guiding the EA's search through the problem space, making it a challenge to design an effective one [6].

2.4 Objective Search

The first fitness function being examined is the objective function. This is perhaps the most straightforward approach to designing a fitness function.

In order to design an objective function, the desired/target complex task/behaviour needs to be broken down into simpler quantifiable sub-tasks that either need to be maximised or minimised [18]. These sub-tasks guide the EA towards the overall more complex behaviour.

The objective function can be seen as a weighted sum of these sub-tasks. The values for the sub-tasks are summed together in order to produce a final fitness value that is assigned to that controller.

For example, if the controller needs to learn a simple gathering task, the objective criteria can be *Distance Covered*, *Resources Found* and *Resources Collected*. These criteria can be easily observed and compared within the simulation. In order to perform the gathering task successfully, all these objectives need to be maximised meaning that the controllers with the highest total score are used for the next generation.

A general outline of an objective fitness function can be seen as:

$$f(x) = \sum_{i=1}^{n} w_i N_{xi} \tag{1}$$

where f is the total objective fitness for the x individual, i iterates over all the objectives in the sum, n refers to the total number of objectives, w_i refers to the weighting of a particular objective and N_{xi} refers to the score that individual x achieved for objective i.

A major downside to using an objective fitness function is that they tend to be easily fooled by deceptive problem spaces and often get stuck around local optima[18].

As the complexity of the desired task increases, so does the probability that the search function will be deceived by a local optima [8].

This is because the objective function is only concerned with the end goal instead of rewarding intermediate states that could potentially result in an optimal controller [18].

2.5 Novelty Search

The novelty fitness function provides a candidate solution with a reward based on how different its phenotypical behaviour is from the other individuals in its generation[11].

In order to calculate the difference in behaviour between individual candidate solutions, the experimenter must first define a similarity measure.

The behavioural characteristics of an individual need to be quantified and assigned a numerical value so as to be able to calculate its distance from its neighbours in the resultant behaviour space [11]. This distance is the overall fitness for the individual.

Not only has it been shown that novelty search can consistently find solutions faster than objective functions in deceptive search domains[17, 10] but it can also discover a set of solutions instead of converging to just a single area in the solution space [11]. By pursuing a dynamic objective, novelty search is able to maintain behavioural diversity within the population and is prevented from prematurely converging to a suboptimal solution [17].

The disadvantage of using novelty search is that it may result in a lot of time being wasted while searching through novel regions in the solution space that may not contribute towards an optimal solution [11].

2.6 Hybrid Search

The final fitness function that will be investigated is the hybrid search function, which is a weighted average between its objective and novelty fitness values [10].

In order to calculate the hybrid fitness of an individual, its objective and novelty fitness values are first calculated separately. These final values are then summed and an average is calculated.

3. PROPOSED SOLUTION

3.1 Method

The section below outlines the different steps in conducting the experiment as well as the details on implementing the various aspects of the investigation, such as the collective construction ask and the increasing levels of complexity.

3.1.1 ML Framework

In order to perform this investigation, the HyperNEAT algorithm will be used to evolve the ANN controllers.

The EA's will be implemented using the encog library [16]. This is a machine learning library that is implemented in Java and can be used to run several neuroevolution algorithms, most importantly NEAT and HyperNEAT. It should also be noted that the library allows for easy modification of the source code and manipulating the algorithms parameters.

3.1.2 Fitness Functions

This library allows for the experimenter to define their own fitness functions that are to be used. This means that the novelty, objective and hybrid functions can be specially designed to suit this instance of the collective construction task.

3.1.3 Simulated Environment

The MASON Multiagent Simulation Toolkit was used to implement the simulated environment [19]. This toolkit is implemented in Java and produces results that are consistent across various platforms [19] which is important since these experiments were programmed and run on computers using different operating systems. The toolkit also uses self-contained models that are independent of their visualisations. One of the main reasons for using the MASON library is that it can be run inside other Java frameworks [19], making it ideal to work with the Encog Machine Learning Framework.

The simulated environment provides a 20x20 metres continuous two-dimensional area within which the entities can interact.

3.1.4 Construction Blocks

The resources scattered throughout the environment are implemented as squares and the different types of resources are denoted using different colours.

The resource blocks that are being used are also very basic. That is to say, they do not store any information and do not perform any sort of processing.

Each type of construction block also requires different numbers of robots to move them.

That is, the first type can be moved by a single robot, the second can only be moved by two robots and the third only by three.

3.1.5 Construction Schema

The construction schema is a set of underlying connection rules that specify which types of blocks can be connected on which sides of each other.

In other words, for every type of block, the construction schema indicates which other blocks can be connected to each of its respective sides.

This schema is used to control the level of complexity inherent in the collective construction task.

It is also used to control the level of cooperation required from the controllers in order to complete the task.

This schema not only specifies the connection rules, but also which types of blocks are to be used in the experiment.

The various levels outlined earlier on in the paper are implemented as follows:

- Level 1: in this configuration, only the first type of block is present in the experiment meaning that no cooperation is required from the agents. There are also no connection rules which means that every block can be connected to every other block.
- Level 2: all the blocks present can be moved by individual robots so no cooperation is require but there are some connection rules between the different blocks.
- Level 3: there are three different types of blocks which require the robots to cooperate but no connection rules.
- Level 4: the different types of blocks are present, requiring cooperation, as well as having complex connection rules.

3.1.6 Robustness

The robustness of a controller refers to its ability to still complete the task at hand after having been incapacitated in some way.

All of the robots start off with the same sensory configuration. Whenever the controller is being tested, a set of random sensors is deactivated. This means that the input from these sensors are blocked from reaching the input nodes in the ANN to simulate them having broken.

A different random set of sensors are disabled for each controller being tested.

3.1.7 Construction Zones

A construction zone refers to a structure that consists of two or more connected blocks.

It refers to the shape that is currently under construction and it is formed when the first two resources are successfully connected to each other.

Its position varies between simulations but is fixed at the location which the first two resources are connected.

Resources/blocks can only be connected to the outer edges of the construction zone.

Once a block is connected to a construction zone it can not be disconnected again.

3.2 Experiment Outline

A single experiment consists of evolving a population of ANN controllers for a set number of generations or until a controller is produced that has a fitness over a certain threshold. The independent variables of a single experiment run are the *Fitness Function* and the *Construction Schema*.

The experiments will be repeated twenty times for each case.

The best performing controllers out of these runs are selected for each experiment case and are then tested in the simulator against each other.

Each successful controller is then tested and evaluated in the simulator five times, a fitness value calculated each time, summed and averaged in order to get a final value for that controller.

3.3 Evaluation

3.3.1 Evaluation Runs

Since this project is investigating a stochastic algorithm, it is necessary to obtain an average fitness score in order to get an accurate measure of a controller's overall task performance.

In order to do this, the most successful controller from each approach is implemented in the simulator and it attempts to solve the task a number of times. Its task performance is then averaged over all the evaluation runs.

In this case, task performance is defined by the number of resources that were successfully connected during the simulation.

3.3.2 *Comparisons*

In order to determine which of the HyperNEAT approaches produce the most robust controllers, the most successful controller from each approach is once again implemented in the evaluation runs. In this case, each controller will have the same random sensors disabled for each evaluation run. There will be a different random sensor morphology for each evaluation run so as to be able to get an accurate representation of the controllers average task performance. Each controller will have the same random morphology for each evaluation run.

Each approach will also be run in the evaluation simulator while using a full sensory morphology and an average task performance is then calculated. Each approach can then be compared with a version of itself where no sensors are damaged and it will be possible to determine whether or not these controllers are robust by looking at the performance difference between full sensors and the damaged ones.

3.3.3 *Statistical Tests*

Various statistical tests will be performed on the results in order to compare the task performance for the respective controllers.

First, the K-S test will be applied in order to determine if the data is normally distributed.

In order to determine if there is a statistical difference between the data sets, the Mann-Whitney U Test is performed between each different approach, and then between each approach and its perfect version (full sensor configuration).

3.4 Expected Outcomes and Limitations

This investigation is expected to have the following outcomes:

- The HyperNEAT algorithm produces controllers that are able to consistently solve the collective construction problem.
- The novelty implementation of the HyperNEAT algorithm will produce controllers that outperform the other implementations.
- That the controllers produced are able to sustain significant damage to their input sensors while still being able to perform the task.

Since most of the code has already been implemented and the experiment setups have already been tested on the designated computers, a lot of the initial risks have been eliminated.

The main risk that this investigation faces is the time limit. These experiments are to be run using the machines in the senior computer laboratory which is shared between the senior computer science students and also used for various computer science-based events such as hackathons and exam venues.

The best way to work around this is to run the experiments on weekends and holidays so that as many of the machines can be used as possible. This means that all the experiments need to be ready and all the code needs to properly implemented before the winter holiday starts, which will be the time when the most computers are available for the longest period of time. In order to accomplish this, all the code needs to be properly implemented and the experiments correctly setup by the time the holiday starts so that none of that time is wasted on developing.

4. REFERENCES

[1] L. Chaimowicz, T. Sugar, V. Kumar, and M. F. M. Campos. An architecture for tightly coupled multi-robot cooperation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2992–2997. IEEE, 2001.

[2] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2764–2771. IEEE, 2009.

[3] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *Evolutionary Computation, IEEE Transactions on*, 15(3):346–367, 2011.

[4] D. B. D'Ambrosio and K. O. Stanley. Generative encoding for multiagent learning. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 819–826. ACM, ACM, 2008.

[5] J. E. Dayhoff and J. M. DeLeo. Artificial neural networks. *Cancer*, 91(S8):1615–1635, 2001.

[6] S. Doncieux and J.-B. Mouret. Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, 7(2):71–93, 2014.

[7] A. E. Eiben and J. E. Smith. Introduction to evolutionary computing. 53, 2003.

[8] S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the sixth international conference on Artificial life*, pages 238–247, 1998.

[9] J. Ghosh, B. J. Kuipers, and R. J. Mooney. Efficient evolution of neural networks through complexification. 2004.

[10] J. Gomes, P. Mariano, and A. L. Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 943–950. ACM, 2015.

[11] J. Gomes, P. Urbano, and A. L. Christensen. Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, 7(2-3):115–144, 2013.

[12] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In *European Conference on Machine Learning*, pages 654–662. Springer, 2006.

[13] F. J. Gomez and R. Miikkulainen. Robust non-linear control through neuroevolution. 2003.

[14] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone. Hyperneat-ggp: A hyperneat-based atari general game player. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 217–224, New York, NY, USA, 2012. ACM.

[15] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone. Hyperneat-ggp: A hyperneat-based atari general game player. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 217–224. ACM, 2012.

[16] J. Heaton. Encog: Library of interchangeable machine learning models for java. *Journal of Machine Learning Research*, 16:1243–1247, 2015.

[17] J. Lehman and K. O. Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *ALIFE*, pages 329–336, 2008.

[18] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[19] S. Luke, G. Balan, K. Sullivan, and L. Panait. Mason library homepage. http://cs.gmu.edu/ eclab/projects/mason/.

[20] R. Miikkulainen. Evolving Neural Networks. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, pages 2441–2460, New York, NY, USA, 2010. ACM.

[21] R. Miikkulainen. Neuroevoluion. In *Encyclopedia of Machine Learning*. Springer, New York, NY, USA, 2010.

[22] G. S. Nitschke, M. C. Schut, and A. Eiben. Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task. *Evolutionary Intelligence*, 3(1):13–29, 2010.

[23] G. S. Nitschke, M. C. Schut, and A. E. Eiben. Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm and Evolutionary Computation*, 2:25–38, 2012.

[24] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[25] K. O. Stanley and R. Miikkulainen. *A taxonomy for artificial embryogeny.*, volume 9. 2003.

[26] P. Tonelli and J.-B. Mouret. On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PloS one*, 8(11):e79138, 2013.

[27] P. Verbancsics and K. O. Stanley. Evolving Static Representations for Task Transfer, 2010.

[28] J. Watson and G. Nitschke. Deriving minimal sensory configurations for evolved cooperative robot teams. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 3065–3071. IEEE, 2015.

[29] J. Watson and G. Nitschke. Evolving Robust Robot Team Morphologies for Collective Construction. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 1039–1046. IEEE, 2015.

[30] J. Wawerla, G. S. Sukhatme, and M. J. Mataric. Collective construction with multiple robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2696–2701. IEEE, 2002.

[31] J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal. Distributed construction by mobile robots with enhanced building blocks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2787–2794. IEEE, 2006.