Evolving Morphological Robustness for Collective Robotics

Ruben Putter
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: PTTAND010@myuct.ac.za

Geoff Nitschke
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: gnitschke@cs.uct.ac.za

***Abstract*—This study evaluates a *Neuro-Evolution* (NE) method for controller evolution in simulated robot teams, where the goal is to evaluate the *morphological robustness* of evolved controllers. *Artificial Neural Network* (ANN) controllers are evolved for a specific sensory configuration (morphology) and then evaluated on a set of different morphologies. The morphological robustness of evolved controllers is evaluated according to team task performance given a collective construction task of increasing complexity. The overall objective was to ascertain an appropriate method for evolving ANN controllers that are readily transferable to robot teams with varied morphologies. Such controller transfer is necessary if task specifications change and different sensory configurations are required, or if robots are damaged and some sensors become disabled. In both cases it is ideal if teams continue to exhibit consistent behavior and a similar task performance. Results indicate that an indirect (developmental) encoding NE method consistently evolves controllers that fully function when transferred to teams with varied morphologies. That is, where comparable or higher task performances were yielded compared to controllers evolved specifically for the varied morphology.**

I. Introduction

Raw information that still needs to be edited

Although Reinforcement Learning (RL) techniques can theoretically solve a number of problems without examples of correct behaviour, in practice they scale poorly with problems that have large state spaces or non-Markov tasks, which are tasks where the state of the environment is only partially observable [1]. (This is from the honours literature review)

These definitions can maybe be placed in a separate section for the problem space specifically The collective construction task requires that a team of agents cooperate by coordinating their behaviours in order to assemble various structures within their environment [2]. [3].

The collective gathering task requires a team of agents to coordinate sub-tasks amongst themselves in order to produce a collective behaviour that maximises the resources gathered [2].

It has been shown that NE can successfully be applied to complex control tasks that require collective behaviour. These control tasks have no obvious mapping between the input from sensory configurations and motor outputs [2]

The high level of complexity inherent in real life tasks tend to result in deceptive fitness landscapes [4]. These are fitness landscapes of multimodal problems that have numerous local optima [5]. In multimodal problems, it is possible for the NE search to get stuck around a local optima and lead to the NE method prematurely converging to a suboptimal solution [4].

For this reason, various NE search functions (also referred to as fitness functions) are investigated. The first search function being the objective search which simply evaluates the suitability of a solution by using an objective fitness function [6]. Novelty search is an approach that provides candidate solutions with a reward based how different their behaviour is from other candidate solutions [4]. The third approach is a hybrid search which attempts to combine EAs with a local search [**?**].

The ability to automate construction tasks through evolving ANN controllers for multi-robot teams using EAs presents several considerable benefits. Automation such as this could be used to assist in projects such as producing low-cost housing and reduce high accident rates due to human error that accompany traditional construction methods [**?**]. It would be useful to send multi-robot construction teams to extraterrestrial planets in order to build habitable structures in anticipation of humans arriving. These robot teams would also be useful in underwater construction which is difficult and often extremely dangerous [7].

This is a nice segway and reference for moving on to the different examples of where this algorithm has been implemented before Another reason for using HyperNEAT is its success in evolving team (collective) behaviours for various multi-agent tasks including RoboCup soccer and pursuit evasion [8].

This is the INTRODUCTION from one of the previous papers that we did SSCI I think An open problem in *collective* [9] and *swarm robotics* [10] is ascertaining appropriate sensory-motor configurations (morphologies) for robots comprising teams that must work collectively given automatically generated controllers [11]. In *evolutionary robotics* [12], a popular method is to co-evolve robot behaviors and morphologies [13], [14], [15], [16]. Within the purview of such *body-brain* co-evolution, indirect encoding (developmental) methods have been effectively demonstrated for various single-robot tasks [17], [18], [19]. However, with few exceptions [20], [21], [22] there has been relatively little work that evaluates developmental methods to evolve behavior and morphology for collective robotics tasks, excluding related research in self-assembling and reconfigurable collective robotics [23], [24].

Specifically, this study focuses on developmental methods to evolve controllers that exhibit consistently robust behaviors such that robot teams effectively adapt to the loss of sensors or new sensory configurations without significant degradation of collective task performance, that is, *morphologically robust behavior*.

This study contributes to an open problem in evolutionary robotics, that is to evolve robot controllers that can continue to appropriately function given unforeseen morphological change to the robot such as sensor damage or malfunction [25], [26]. Such robust controller evolution would yield significant advantages in applications where robust autonomous behavior is continually required [27]. In this study, the specific focus is on the evolution of morphologically robust behavior for robotic teams that must accomplish collective behavior tasks.

Given the added computational complexity of co-evolving body-brain couplings for behaviorally and morphologically heterogenous robots, this study focuses on evolving collective behaviors for *homogenous* teams with a range of fixed morphologies. We apply the HyperNEAT [28] neuro-evolution method to evolve behavior for a range of morphologically homogenous teams that must solve various collective construction tasks. HyperNEAT was selected as it is a developmental encoding NE method with demonstrated benefits that include exploiting task geometry to evolve modular and regular ANN controllers with increased problem-solving capacity [29], [30].

1) Research Contribution: The research contribution is to demonstrate the effectiveness of developmental neuro-evolution (HyperNEAT) for addressing the evolution of morphologically robust controllers in the context of collective robotics. To address this objective, this study tested and evaluated five different robot team sensory configurations (morphologies) in company with three collective construction tasks of increasing complexity. The fittest *Artificial Neural Network* (ANN) controller evolved for a given morphology was then transferred and evaluated in each of the other team morphologies. Such transferred controllers were also evaluated in each of the three tasks. In all cases, evolved controller efficacy was evaluated in terms of collective construction task performance yielded by a robot team.

remember to have a section explaining why multi-robot teams instead of just one big robot and also find some use cases of these such as underwater construction, terraforming other planets etc.

End of Raw information

In this section you are going to try and mash together the contents from all of the different papers in order to hopefully get some coherent section.

information from Project Proposal

One of the central themes of research in mathematics and computer science is the design and development of auto- mated problem solvers [31].

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 the traditional controller design impractical [1], resulting in a growing demand for the automated design of these problem solvers [31]. (This section definitely needs to be reworded)

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 it own niche environment [31], which is why computer scientists have turned to natural processes for inspiration [1]. (You can also add something about how the constraints of the environment is what determines the fitness functions? Basically you can try and link EC with natural selection. Explain how we use ANN's to simulate an abstract representation of the brain and then we use simulations in order to reproduce the trial-and-error problem solving ability of evolution by natural selection)

Evolutionary Computing is a branch of computer science research inspired by the Darwinian process of evolution through natural selection [31].

Neuroevolution provides an effective approach to solving complex optimization and control tasks [32].

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 [33].

PROS OF PERFORMING THIS INVESTIGATION 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 [7].

Automated construction would also be useful in future endeavours such as construction in areas that are too dangerous or inaccessible to humans [7]. This includes examples such as sending a team of robots to extraterrestrial locations to build habitats in anticipation of human travellers and underwater construction [33].

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 [7]

end of Raw information

A. Problem Statement

This is explaining why we are investigating the morphological robustness of controllers, because there is an inherent danger in the environments meaning that it is likely that the agents would sustain damage while exploring the terrain. Because of this inherent danger, it is necessary that the individual robots are still able to complete their task after having sustained some level of minor damage. This co0uld also mean that the ANN controllers can be used for a variety of different robots without needing to be retrained with a new morphology. You can say that the controller has been evolved to solve the task independent on the input sensors that it gets provided with (independent on whether or not they are using the same sensors as they were evolved with) Real life implementation of these situations have an inherent danger in the environment. The whole point of using robots is because humans are unable to gain access, or it is too dangerous for a human to do it.

As such, the robot will have to be able to complete its task to some degree without requiring any human intervention or repairs, which may not even be possible in some situations.

There are also other ways of compensating the loss of a sensors, particularly because the robot team is homogenous. It could be possible that some of the other agents' are able to accommodate for an agent losing some of its input sensors and its abilities becoming restricted. Let's say that one of the agents loses it's construction sensor, it should still be able to assist other robots in carrying a resource around (it can follow another robot that has a construction sensor). Although this may require that the individual robots are able to communicate with each other.

Raw information that still needs to be edited

specifically in this case, their ability to deal with a change in their input sensor configuration. (Definitely need to reword this).

Since the controllers can't be retrained to use the new sensor morphology and the structure of the ANN can not be altered (set number of inputs an outputs) (when a sensor breaks, you can't redesign an ANN, train it and then still deploy it into the robot), 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 homogenous robots are still able to complete a task requiring collective behaviour [34], as well as an investigation into evolving specialised sensor resolutions in a multi-agent task [35]. Another similar project includes the investigation into evolving an optimal configuration of input sensors such that a homogenous team of robots can solve the collective construction task [21].

What makes this project different from existing research, is that it is specifically investigating the ability of HyperNEAT evolved controllers to deal with the loss 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 homogenous team and then comparing its task performance to that of a robot team with all of its sensors in tact this outline above only refers to one of the experiment conditions now

End of Raw information

II. Background Research

Previous work in [7] are very similar to the investigation being conducted in this paper, except that they have a desired shape that the robots are meant to build

Raw information that still needs to be edited

End of Raw information

A. Artificial Neural Networks

Raw information that still needs to be edited

Artificial Neural Networks (ANN's) are a simplified model of how a biological brain functions[36]. It consists of numerous simple computational units, called neurons, that are interconnected and ordered into layers to create a neural net[37]. Each neuron has several inputs, some activation function and a single output. An ANN receives input from the environment at its input layer [1].

Each neuron in a non-input layer then calculates the weight-ed sum of its inputs, referred to as the activation value [38], [37], and evaluates it according to some activation function. If this result exceeds a predetermined threshold value, the neuron will "fire" an output signal, transmitting it as an input to the subsequent neuron across a weighted connection [38].

ANNs are universal function approximators. It has been shown that a network can approximate any continuous function to any desired accuracy [39]. This makes ANNs a favourable choice for agent controllers in accomplishing various tasks [38].

The topology of an ANN is determined by the number of neurons and the interconnections between them[40]. ANN's can be classified as feed-forward or recurrent depending on the connectivity between its neurons [41]. An ANN is classified as feed-forward if data moves through the network from inputs to outputs only [1]. That is to say, if there exists a method of numbering the respective layers of the network such that all the inter-neuron connections terminate at a neuron with a greater layer number than the neuron it started from. Conversely, an ANN is classified as recurrent if there are connections that terminate at nodes with smaller values than the source node such that it forms a feedback connection [37], [41] (from the honours Literature review)

In cases where training examples are available, the connection weights in ANNs are adapted using various supervised learning techniques, and in cases without such examples the weights are evolved using EAs [42], [41].

End of Raw information

B. Machine Learning

Raw information that still needs to be edited

End of Raw information

C. Biological Basis

In this section find some stuff regarding the natural models and things that this research is originally based on -¿ Darwinian process of evolution through survival of the fittest -¿ How we replicate these situations and manipulate variables to test the different outcome -¿ We implement survival of the fittest by implementing fitness functions that assign a value to an individual based on its performance metrics

Raw information that still needs to be edited

Evolutionary computing (EC) is a branch of computer science research inspired by the Darwinian process of evolution through natural selection[5]. Given the complexity and non-linearity of real world problems, traditional controller design is impractical since they rely on linear mathematical models [1], making it unsurprising that computer scientists have turned to natural processes for inspiration. The problem solving power of evolution is abundantly clear in nature with the diverse range

of organisms on Earth, each specially designed for optimal survival in its own niche environment [5].

End of Raw information

D. Neuroevolution

Raw information that still needs to be edited

Neuroevolution (NE) is a method of artificially evolving Artificial Neural Networks (ANNs) by implementing Evolutionary Algorithms (EAs) to perform tasks such as training the connection weights and designing the network topology[37], [41].

Neuroevolution (NE) provides a way of combining Evolutionary Algorithms (EAs) and Artificial Neural Networks (ANN's) (much in the same way that the brain was evolved by evolution, can maybe somehow use this to link up with that biological basis thing)

Learning in ANN's, also referred to as training, is accomplished by iteratively adjusting the connection weights between neurons until the desired output is achieved [42]. NE is able to train an ANN to accomplish a specific task by using EA's to perform this connection weight adaptation [43].

The main advantages to using NE to design these controllers are that it works in continuous and partially observable environments [44], the designer does not need to specify how the task should be solved [1], and it does not require some value function or a corpus of training examples [32].

By evolving a population of ANNs, NE searches through the space of possible controllers by evaluating each candidate solution using a fitness function and selecting the most successful individuals for reproduction, a process analogous to natural selection [1], [45].

The different NE methods can be roughly 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. In indirect (also called generative or developmental) encoding methods, each element in the genotype is implicitly described by a computable function allowing for a much more compact representation of the genotype. It also allows for genetic information to be reused [46], [**?**].

This section below is from the Literature review and still needs to be edited

Neuroevolution (NE) provides a way of combining EAs and ANNs [37]. It uses EAs to evolve the connection weights, topologies and activation functions of ANNs in order to learn a specific task or behaviour [47]. NE searches for an ANN with optimal performance based on a fitness function that determines an ANN's suitability to performing a task [37].

Using NE to evolve ANNs addresses several weaknesses that occur in reinforcement or supervised learning techniques. Reinforcement learning (RL) requires a value function which is costly to compute whereas NE removes the need of such a function by directly searching the policy space using a EA [1].

By searching through a population of solutions and evaluating several candidate solutions at a time, EAs are much less likely to get stuck in local minima than gradient-descent algorithms and therefore makes NE suitable for dealing with complex problems that generate numerous local optima [45], [41].

In a standard RL scenario an ANN interacts with its environment in discrete time steps [48]. NE can be used in any type of environment, whether it is continuous or partially observable, making it possible to find an optimal ANN using only information about how well the network is performing rather than what it is supposed to be doing [44].

Another challenge in using traditional learning methods or fixed-topology NE approaches, is that it requires a human to design the topology for the neural net. This presents a problem since these networks can get complex and would have to be altered according to the specific task being learned. This relation between topology and suitability is very unintuitive and difficult to get right without trial-and-error. By using a NE approach that evolves the topology and the weights of a neural network (TWEANNs), these networks are able to discover the most efficient topology [37].

Since the chromosomes in NE can be used to encode any aspect of ANNs and the choice of encoding affects the search space, deciding on an encoding method is fundamental aspect in the design of the system [37]. This literature review compares direct and indirect encoding, which will be discussed in depth at a later stage in the Encoding Schemes.

The below is from the paper Neuroevolution by Risto Miikulainen

Neuroevolution is a method for modifying neural network weights, topologies, or ensembles in order to learn a specific task. Evolutionary computation is used to search for network parameters that maximize a fitness function that measures performance in the task [define: fitness function].

It can also be combined with standard neural network learning to model biological adaptation. (this is like the learning algorithms per generation and then the survival of the fittest as determined by the fitness function) (the biological adaptation is referring to the process of evolution through natural selection over hundreds of generations.) maybe find an example of what this means? how does one combine NE with standard neural network learning algorithms? Neuroevolution can also be seen as a policy search method for reinforcement learning problems, where it is well suited to continuous domains and to domains where the state is only partially observable (look up what the policy has to do with reinforcement learning. seem to recall it was the steps to take for a solution or something like that. look in the reinforcement learning papers) (this is a good comparison or relation to link Neuroevolution and reinforcement learning, maybe include a whole section for reinforcement learning by itself as part of the literature review. this can help support a metaphor for natural processes, always good to have as mach background information as possible. just make sure to not go overboard and start including irrelevant stuff)

The main benefit of Neuroevolution compared to other Reinforcement Learning methods in such tasks is that it

allows representing continuous state and action spaces and disambiguating hidden states naturally. Network activations are continuous, and the network generalizes well between continuous values, largely avoiding the state explosion problem that plagues many reinforcement learning approaches. (look up what the state explosion problem is that plagues the RL approaches)

Recurrent networks can encode memories of past statess and actions, making it possible to learn in Partially Observable Markov Decision Process (POMDP) environments that are difficult for many RL approaches. (Look up what the POMDP environment is again) (Also, im not sure about that thing about encoding memories considering none of the networks even had a hidden layer. will need to check if any of the networks are actually recurrent)

Compared to other neural network learning methods, neuroevolution is highly general. As lon as the general performance of the networks can be evaluated over time, and the behaviour of the network can be modified through evolution, it can be applied to a wide range of network architectures, including those with non-differentiable activation functions and recurrent and higher-oorder connections. (from this paragraph above, look up some examples of non-differentiable functions)

While most neural learning algorithms focus on modifying the weights only, neuroevolution can be used to optimize other aspects of the networks as well, including the activation functions and network topologies. (This is probably not the best thiing to include considering you don't modify the activation functions or anything like that.)

End of Raw information

E. Evolutionary Computing

Raw information that still needs to be edited

End of Raw information

F. Evolutionary Algorithms

-¿ In this section use the information you got for your literature review with regards to various other algorithms that could have been used (NEAT, CE, etc)

G. HyperNEAT

-¿ How it came to be -¿ Why we chose it for these experiments

Raw information that still needs to be edited

HyperNEAT [28] is an extension of NEAT (*Neuro-Evolution of Augmented Topologies*) [49], where ANNs are indirectly encoded using a CPPN (*Compositional Pattern Producing Network*) [50] HyperNEAT was selected as it has a number of benefits demonstrated in previous work [30], [21]. This includes its capability to exploit geometric features such as symmetry, regularity and modularity in robot morphology and the task environment during controller evolution. in the geometric features include the relative positions of other robots, blocks, the direction robots and blocks are facing and the shape of the environment. Also, the structures to be built are modular (comprised of blocks) and often regular (the same sequence of blocks can be repeated).

End of Raw information

H. Fitness Functions

In this section, we take a closer look at the different types of fitness functions that were used as part of this investigation. These fitness functions were used to quantify the performance of an ANN agent at completing the Collective Construction Task. It provides us with a way in which to evaluate the controller's performance and be able to compare the performance of each controller in order to be able to determine the most successful controller.

These fitness functions were used during both the Evaluation and the Evolutionary phases. We use it to determine fitness for selection between generations of populations and we use it determine the performance of the controller in the evaluation phase in order to determine the results of the investigation.

A team's task performance was determined by the total number of blocks that were connected to the construction zone during a team's lifetime

An experiment is outlined as follows: For a single morphology: - a population of 150 agents is initialised - the evolutionary stage runs for 100 generations - between each generation, each agent in the population is selected and evaluated in order for Selection to be able to occur - each controller is therefore then deployed back into the simulated environment for another lifetime - at the end of this lifetime, the fitness of the controller is calculated and assigned. - Individuals are selected from the population in order to create the subsequent population and the procedure will be repeated until all of the generations are complete - A single experiment consists of a population of 100 individuals (?) that are evolved for 100 generations, in between each generation are a couple of runs in the simulator during which the fitness of the controller is calculated and selection is performed. This single experiment is for a single morphology configuration

The fitness function specifically refers to the way in which an agent's fitness is calculated between generations in order for selection to be able to occur. These functions evaluate different criteria respectively (objective novelty and fitness)

The task performance calculation was used only at the end in order to be able to compare the performance of the various teams that were selected as the best individual controllers that were found for that method.

Team task performance was calculated as the number of blocks connected in construction zones during a team's lifetime (equation 2), where average task performance was calculated as the highest task performance selected at the end of each run (100 generations) and averaged over 20 runs (table II). The fitness function to direct controller evolution was a weighted sum of the number of times *type A* blocks were pushed by *one robot* and connected (a in equation 2), the number of times *type B* blocks were pushed by *two robots* and connected (b in equation 2), and the number of times *type C* blocks were pushed by *three robots* and connected (c in equation 2).

A general fitness function can be illustrated as follows:

$$f = r_a a + r_b b + r_c c \quad (1)$$

Explain what a fitness function is Give an example Explain how it fits in with natural selection

1) Objective: Raw information that still needs to be edited

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 complex behaviour needs to be broken down into simpler quantifiable sub-tasks that either need to be maximised or minimised [6]. 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 by applying the chosen selection method (add some line on the selection method part of the Evolutionary algorithm).

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 [6]. As the complexity of the desired task increases, so does the probability that the search function will get stuck at a sub-optimal local optima [51]. This is because the objective function is only concerned with the end goal instead of rewarding intermediate states that could potentially result in a globally optimal controller [6].

End of Raw information

III. Applications of HyperNEAT

A. Evolving Quadruped Gait

"Furthermore, given how sensitive gait controllers are to slight changes in the configuration of a robot, a new gait must be created each time a robot is changed, which can lead to significant delays in the prototyping stage of robotic development [52]" (verbatim from [53])

"It is not surprising therefore, that people have tried to automate the process of gait creation. Evolutionary computation, frequently involving the evolution of neural network controllers, has been successfully used to this end [53]"

Additional references to take a look at for some more information/illustrations regarding Evolutionary Computing [52], [54], [55], [56],

"Evolved gaits are often better than those produced by human designers; one was even included on te commercial release of Sony's AIBO robotic dog [52]"

"However many researchers have found that they cannot hand the entire problem over to evolutionary algorithms, because of the large number of parameters that need to be simultaneously tuned to achieve success" [53]

The results of the following investigation [53], show that HyperNEAT (indirect encoding) greatly outperforms FT-NEAT (a direct encoding method) (can use this for another section on WHY HYPERNEAT)

B. Collective Construction Task

Raw information that still needs to be edited

The collective construction task requires that a team of agents cooperate by coordinating their behaviours in order to assemble various structures within their environment (MastersProjectProposal) [2].

The general implementation of the collective construction task can be broken down into 3 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 resoruces blocks that are scattered throughout their environment. Second, once a robot has found a resource, this resource needs to be taken to the designated construction zone in order to connect it to the current structure. These first 2 steps can be seen as an implementation of the standard collective gathering task. (This last sentence definitely needs to be reworded) [**?**]. 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 place resources in the most effective and efficient way possible.

The *Collective construction* task [57] was selected since it benefits from fully automated robot teams that must exhibit robust behavior to handle changing task constraints, potential robot damage and sensor noise. For example, collective construction of functional structures and habitats in remote or hazardous environments [58].

As in related work [57], the task was for robots to search the environment for *building-block* resources, then move them such that are connected to other blocks. The task is successfully completed if the team connects all blocks forming a structure during its *lifetime*, and this is equated with optimal task performance. However, the task is considered partially solved if only a subset of the blocks are connected by the team, though in such instances, team task performance is proportional to the number of blocks connected. In this study, task complexity was equated with the number of robots needed to collectively push blocks together (cooperation) to be connected as a structure and whether the blocks must be connected in a specific sequence, that is, according to a *construction schema*, table III. Task complexity was regulated via the environment containing three block types, where one, two and three robots were required to move each of the block types, respectively.

Hence, we report a preliminary investigation into developmental methods (HyperNEAT is tested in this case study) for evolving ANN controllers that are robust to morphological change in robotic teams or swarms that must operate in dynamic, noisy and hazardous environments.

End of Raw information

C. Research Aims

Raw information that still needs to be edited

The below is from the Masters Project Proposal

The aims of this investigation are as follows:

- To investigate the behavioural robustness of multi-robot team controllers that are evolved using the HyperNEAT algorithm with respect to sensory configurations when implemented in solving increasingly complex tasks requiring coordinated 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 (Objective, Novelty, Hybrid).

make sure that these are all of the AIMs that we are investigating

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 different types of resource blocks in the simulation.

The various difficulty levels that were implemented in this investigation are as follows:

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

IMPORTANT: Make sure that we actually did implement all of these levels because I think we actually left one out

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

End of Raw information

D. Research Question

Raw information that still needs to be edited

Literature Review research question: "Which implementation of the HyperNEAT algorithm is able to produce the most robust controller in such a way that they are able to consistently solve a collective behaviour task of increasing complexity?"

"which neuroevolution algorithm is able to produce the most robust ANN controller with regards to its ability to complete the collective construction task while varying the morphology (type of and configuration) of its input sensor?"

We evolve a controller with a fixed sensory configuration and use that to establish a base line performance. We then create several different sensory configurations that are a subset of the original sensor collections

There were different evaluations for the different experiments: -¿ In the SSCI experiments (I think?), we implemented a sort of filter that would randomly block input from some of the sensors that the robot is originally configured with. This was to try and simulate some of the robot's sensors becoming disabled while performing the task. In this case, we would change which sensors have been deactivated between each evaluation run so that the controller had to perform the task with a different subset of its sensors each time (essentially a different sensory input configuration with each evaluation run) -¿ For the other experiments, we chose a fixed set of morphologies. Each controller was evolved using a particular sensory configuration. We then established a baseline performance by performing a couple of evaluation runs in the simulator where the controller uses its original morphology. In order to determine the robustness of the controller, we would perform the evaluation stage for each controller where it would be implemented in the simulator with one of the other sensory configurations. This is repeated for each morphology per controller. We then calculate the average performance of the controller across all its evaluation runs with foreign morphologies and compare it to its baseline performance. If there is a difference in the values, we can conclude that the controller is or isn't robust.

We are evaluating the ability of a controller to be able to complete the desired task while changing the sensory input configurations.

This will also help determine whether or not an evolved ANN controller is transferable between robot bodies. In other words, once a controller has been trained to accomplish a certain task, will this controller be able to still complete its task after being transferred into another "body" Can you train controllers on a 'per task' basis only, and then just place it in any robot body. Or is an ANN controller's ability to accomplish a task dependent on the morphology that it was evolved with.

End of Raw information

IV. Experiments

Raw information that still needs to be edited

Experiments[1] tested 15 robots in a bounded two dimensional continuous environment (20 x 20 units) with randomly distributed type *A*, *B* and *C* blocks (table II). Robots and blocks were initialized with random orientations and positions throughout the environment. A construction schema (table III) dictated the sequence of block types that must be connected together in order that a specific structure be built [3]. Figure 2 presents an example of the team of 15 robots working to solve the collective construction task in the simulation environment containing a distribution of five of each block type (*A*, *B* and *C*), colored blue, green and red, respectively. Other colored blocks in the environment indicate those already connected in construction zones (three illustrated). The purple, blue and green semi-circles projecting from each robot represent the Field Of View (FOV) of active sensors, where the different colour11s correspond to different sensor types (table II).

As the purpose this study was to demonstrate the morphological robustness of HyperNEAT evolved controllers for a collective behavior task of increasing complexity, the first two versions of the collective construction task required no

[1]Source code for all experiments is online at: https://github.com/not-my-name/ExperimentsRerun

cooperation and some degree of cooperation, respectively, though any block could be connected to any other block. Where as, the most complex version of the task required cooperation and block types to be connected according to a construction schema (table III).

(the first level is to work as a sort of baseline difficulty, by using a construction schema that requires no cooperation and no complexity)

End of Raw information

A. Collective Construction Task

Raw information that still needs to be edited

This task required the robot team to search the environment for building-blocks and cooperatively push them together in order that they connected to form a structure, where connected blocks then formed a construction zone. Task complexity was equated with the degree of cooperation required to collectively push blocks and connect them together in the construction zone and whether or not a construction schema was required. In this construction task, there were three block types, *A*, *B* and *C* requiring one, two and three robots to push, respectively. Cooperation occurred when at least two robots simultaneously pushed a type *B* block, or at least three robots pushed a type *C* block.

Table III presents the three levels of task complexity for the collective construction task. Level 1 was the least complex as it did not require any cooperation, given that it in this case there were only type *A* blocks in the environment. Level 2 was of medium complexity as there are equal numbers of type *A*, *B*, and *C* blocks in the environment, where block types *B* and *C* required at least two and three robots to push, respectively. Level 3 was the most complex, as it required the same degree of cooperation as task level 2, though blocks had to be connected according to a construction schema. Figure 3 illustrates this construction schema, where the label on each of the four sides of each block type indicates what other block type can be connected to the given side. The *X* label indicates that no block can be connected to a given side.

The construction zone was formed via at least two blocks pushed together and was thus any structure being built in the environment. Once a construction zone was created, all blocks attached to it were fixed in position and could not be disconnected. The task mandated a maximum of three construction zones and unconnected blocks had to be pushed and connected to one of these construction zones. For task levels 1 and 2, any block could be connected to any other block, meaning that when two blocks were pushed together they automatically connected. For task level 3, blocks had to be pushed together such that they were connected on specific sides according to the construction schema (figure 3).

Team task performance was calculated as the number of blocks connected in construction zones during a team's lifetime (equation 2), where average task performance was calculated as the highest task performance selected at the end of each run (100 generations) and averaged over 20 runs (table II). The fitness function to direct controller evolution was a weighted sum of the number of times *type A* blocks were pushed by *one robot* and connected (a in equation 2), the number of times *type B* blocks were pushed by *two robots* and connected (b in equation 2), and the number of times *type C* blocks were pushed by *three robots* and connected (c in equation 2).

$$f = r_a a + r_b b + r_c c \tag{2}$$

Parameter tuning experiments found that setting the weights (reward values r_a, r_b and r_c) in equation 2 to 0.3, 0.6, and 1.0, respectively, resulted in functional controller evolution. Fitness was normalized to the range $[0.0, 1.0]$ using the maximum possible fitness yielded from all blocks being pushed and connected in construction zones. End of Raw information

1) Evolutionary Phase: This is the stage of the experiment during which the evolutionary algorithm is used to evolve the controllers using their fixed sensory morphologies. Each controller is assigned to a homogeneous robot team The team is "deployed" into the simulated environment 5 times and a score is calculated (average?) for that controller. This is repeated for each controller in the population Once each of the controllers has had a chance to attempt solving the task, Selection is performed and the next generation of controllers is selected and the evolutionary process starts all over again. This is repeated for 100 generations at which point the most successful controller is selected from each morphology experiment

A population of controller is evolved for each of the possible sensory configurations Each controller is evolved for 100 generations and the experiment is repeated 20 times for each controller.

2) Evaluation Phase: During this part of the experiments, each of the controllers was re-implemented in the simulated environment with their original evolved sensors in order to establish a baseline score for that network controller. The simulateion is run several times (5?) and then the fitness score is averaged over all of those runs

B. Broken Sensors

For this set of experiment

C. Set of Sensors

For this set of experiments, we first evolved the controllers with their own fixed sensor morphology. The morphology that produced the best score is then selected and used as a benchmark score. This same controller is then implemented in the same environment for a couple of iterations but then it uses

D. Experiment Design

Raw information that still needs to be edited

Experiments evaluated the *morphological robustness* of HyperNEAT evolved controllers for robot teams that must accomplish collective construction tasks of increasing complexity (section IV-A). We measured the average comparative task performance of controllers evolved for a given team morphology and task complexity where such controllers were then transferred to and re-evaluated in other team morphologies. Thus, teams that achieved an average task performance that was not

significantly lower across all *re-evaluated* morphologies were considered to be *morphologically robust*.

This study comprised five experiment sets, where the first four experiment sets evolved controllers given team morphologies $1-4$ (table I) for three levels of increasing task complexity (table III). The fifth experiment set investigated the *co-adaptation* of team morphology and behavior, where morphology 5 (table I) was used as the initial sensory configuration for all robots in the team. This fifth experiment set was included in order to gauge if co-adapting behavior and morphology yielded any benefits in this collective construction task as it did in related collective behavior tasks [22].

Each experiment set comprised a controller evolution stage and a re-evaluation stage (morphological robustness test). For controller evolution, each experiment applied HyperNEAT to evolve team behavior for 15 robots for 100 generations, where a generation comprised five team *lifetimes* (1000 simulation iterations). Each team lifetime tested different robot starting positions, orientations, and block locations in the simulation environment. The fittest controller evolved for each task level (yielding the highest absolute task performance) was then *re-evaluated* for morphological robustness in all other morphologies. For example, the fittest controller evolved for morphology 1 was re-evaluated in morphologies $2-4$ and the average task performance calculated across all re-evaluation runs.

Each re-evaluation run was *non-evolutionary*, where controllers were not further evolved, and each re-evaluation run was equivalent to one team lifetime. Re-evaluation runs were repeated 20 times for a given morphology, in order to account for random variations in robot and block starting positions and orientations. For each fittest controller, re-evaluated in a given morphology, an average task performance was calculated over these 20 runs, and then an overall average task performance was computed for all re-evaluated morphologies.

As per this study's objectives, these morphological re-evaluation runs tested how robust the fittest evolved controllers (for a given morphology) were to variations in that morphology. Thus, re-evaluating the fittest controllers on other morphologies emulated sensor loss due to damage or new robot morphologies introduced due to changing task constraints. End of Raw information

V. Methods

Raw information that still needs to be edited

This section is from the SSCI paper

In this study, HyperNEAT evolves the connection weights between each robot's sensory input layer, hidden layer and motor output layer, where each robot used the same controller, making teams homogenous. Controller evolution experiments were initialized with a given morphology (table I). However, in one experiment set, each robot's sensor configuration of team morphology could be *co-adapted* via HyperNEAT activating and deactivating sensory input node connections over the evolutionary process. For these experiments (section IV), *add connection* and *remove connection* mutation operators (table II) from previous work [22] were applied every generation to a sensory input node chosen with uniform random selection. The mutation operator applied depended on whether the chosen input node was connected or not. The construction zone sensor (table I) was permanently activated for all morphologies and could not be disconnected, as this enabled robots to detect *construction zones* (section IV-A).

Table I presents a list of *morphology identification* (ID) numbers and the number and type of sensors that correspond to each morphology. For example, morphology 2 has four proximity sensors, one ultrasonic sensor, one colour-ranged sensor, and one low-resolution camera. Note that all morphologies have a construction sensor as this is necessary to complete the collective construction task.

End of Raw information

A. Robots/Experiment Agents/Agents

In this section we group everything that has to do with how the actual robot/controller agents were implemented.

1) Team Controller: this is where you mention the ANN Homogeneous robot team controller where each single network is used to control a team of individual robots

Raw information that still needs to be edited This section is from the SSCI paper

Each robot in the team used an ANN controller with N sensory input nodes, determined by the given morphology being evaluated (table I). Each robot's controller mapped sensory inputs, via a fully connected hidden layer, to two motor outputs, the robot's left and right wheels (figure 1).

Figure 1 illustrates the sensory configuration for $N = 11$ (morphology 1), and the associated substrate and CPPN used by HyperNEAT. For each robot morphology (table I), the sensors corresponding to the input layer of the controller was a circle N nodes distributed about a robot's periphery, where the exact geometric configuration corresponded to the morphology being evaluated (figure 1 illustrates morphology 1)[2]. The intermediate ANN hidden layer reflects the configuration of the input layer, preserving the geometry of the sensory input layer, that is, the direction of each sensor's FOV (figure 1). The ANN was initialized with random weights normalized to the range [-1.0, 1.0], with full connectivity between adjacent layers, however, partial connectivity was evolvable via the CPPN generating a zero weight. Collectively all sensors approximated up to a 360 degree *Field of View* (FOV).

The nodes comprising each robot's ANN controller, connected by the CPPN, were placed in the substrate illustrated in figure 1. Each node in the substrate was placed at specific (x, y) locations in the two-dimensional geometric space of the substrate (x, y axes were in the range: [-1, 1]). Connection weights in the controller were evolved via querying the CPPN for the weight of any connection between two points (x_1, y_1) and (x_2, y_2) by inputting (x_1, y_1, x_2, y_2) into the CPPN, which subsequently output the associated weight. During HyperNEAT's evolutionary process, the CPPN was evolved via having nodes and connections added and removed, as well as connection weight values mutated [28].

[2]Illustrations of all robot morphologies tested can be found at: https://github.com/not-my-name/SSCI_Paper_Appendix

Thus, the CPPN evolved a connectivity pattern across the geometry of the ANN via querying all the potential connections for their weights. This connectivity pattern was effectively a function of the task and ANN geometry, which enabled HyperNEAT to exploit the structure (regularity, repetition and symmetry) of the task and robot morphology. For example, there was symmetry in the robot morphology in terms of the positioning of sensors about each robot's periphery (figure 1) and there was regularity and repetition in the collective construction task, in terms of repeating block types comprising modular and regular structures. In the collective construction task, *modularity* was defined as the composition of modular structures (buildings in construction zones) from a sequence of connected blocks and *regularity* was defined as the same sequence of blocks repeated in a building.

Previous work has demonstrated that the indirect encoding of an evolved CPPN facilitates the evolution of robot controllers with increased task performance enabled by a compact representation of task and robot geometry [59], [21]. Table II presents the HyperNEAT parameters used in this study, where *delta* was angle between the (x_1, y_1, x_2, y_2) positions of nodes in the substrate. These parameter values were determined experimentally. All other HyperNEAT parameters not listed in table II, were set as in previous work [59]. The ANN uses a three dimensional coordinate system for processing *x*, *y*, *z* positions in the CPPN in order to generate weight and bias values and connectivity.

Previous work also demonstrated that HyperNEAT exploits the sensory-motor node configuration in ANN controllers, Nodes for processing sensory inputs correspond to the direction each sensor faces.

End of Raw information

2) Detection Sensors: Raw information that still needs to be edited This section is from the SSCI paper

Each robot was equipped with various sensor types, where the exact sensor complement, including the relative position and direction on the robot depends upon the given experiment (section IV) and morphology being evaluated (table I). Each robot had *N* sensors corresponding to the *N* inputs comprising the robot's ANN sensory input layer (figure 1), each with a range of *r* (portion of the environment's length). A robot's sensory FOV was split into *N* sensor quadrants, where all sensors were constantly active for the duration of the robot's lifetime. The *nth* sensor returned a value in the normalized range [0.0, 1.0], in the corresponding *nth* sensor quadrant. A value of 0.0 indicated that no blocks were detected and a value of 1.0 indicated that an object was detected at the closest possible distance to the given sensor.

Table II presents the different sensor types used in this study, where the functional properties of each sensor (range and FOV) were abstractions of corresponding physical sensors typically used on the Khepera III robots [60]. In table II, range values are units defined in relation to the environment size (20 x 20) and FOV values are in radians. Each morphology also included a special construction zone detection sensor that activated with a value in the range [0.0, 1.0] whenever a robot came into contact with a block that must be connected with other already connected blocks.

The construction zone sensor calculated the squared Euclidean norm, bounded by a minimum observation distance, as an inversely proportional distance between *this* robot and the closest construction zone, where a value of 1.0 indicated the robot (pushing a block) was in contact with the construction zone and a value of 0.0 indicated that the robot (pushing a block) was the maximum possible distance from the closest construction zone. Robots were unable to detect each other, thus all cooperative interactions were *stigmergic* [61] where robots interacted via pushing blocks into the environment's construction zone. Furthermore, robots had no *a priori* knowledge of the construction schema, but rather must discover the construction schema rules by trial and error. Also, once at least two blocks had been pushed and connected together this formed a construction zone (section IV-A), that was then visible to each robot's construction zone sensor.

End of Raw information

3) Movement Actuators: Raw information that still needs to be edited

Two wheel motors control a robot's heading at constant speed. Movement is calculated in terms of real valued vectors (*dx* and *dy*). Wheel motors (*L* and *R* in figure 1) need to be explicitly activated. A robot's heading is determined by normalizing and scaling its motor output values by the maximum distance a robot can traverse in one iteration (table II). That is:

$$dx = d_{max}(o_1 - 0.5)$$

$$dy = d_{max}(o_2 - 0.5)$$

Where, o_1 and o_2 are the motor output values, corresponding to the left and right wheels, respectively, producing an output in the range: [-1.0, 1.0]. These output values indicate how fast each respective wheel must turn. Equal output equates to straight forward motion and unequal output results in the robot rotating about its own axis. The d_{max} value indicates the maximum distance a robot can move in one simulation iteration (normalized to 1.0, table II).

End of Raw information

B. Simulator

2-dimensional continuous environment

mention the discretised grid (unless you decide to take it out, in which case you will have to actually do it)

the redbridge simulator that was written in java

C. HyperNEAT

Should there maybe be another section for HyperNEAT under the Methods section? There is already one in the research Maybe just mention here how it was implemented (with Encog library) Also specify the experiment parameters for the algorithm

VI. Future Work

Maybe implement a more accurate abstraction of the sensors Implement other types of variations in the sensors (like partial degradation instead of on/off) More representations

of the collective construction task instead of just connecting blocks, maybe have more complex connection rules.

VII. Appendix

This is just temporary and this section is being used to store the figures and tables and what not Check here for different table formats

VIII. Results & Discussion

For each controller evolution experiment, the average maximum task performance of controllers evolved for a given morphology and level of task complexity, was recorded. Specifically, this task performance was calculated by running the absolute fittest controller evolved after 20 evolutionary runs (for a given a morphology and task level), in the same morphology over 20 non-evolutionary runs. This is presented in figures 4, 5 and 6 (left column), where controllers were evolved given morphologies 1-5 (table I). In figures 4, 5 and 6 (left column), these results are presented from left to right. For example, average task performance results for morphology 1 are plotted on the left-most side and average task performance results for morphology 5 are plotted on the right-most side.

For each re-evaluation experiment, the fittest controller evolved for a given morphology and task complexity level was re-evaluated in all other morphologies (for the same level of task complexity), and an average task performance computed over 20 runs. These morphological re-evaluation results are presented in figures 4, 5 and 6 (right column). Each of the five plots (from left to right) in each figure corresponds to the fittest controller evolved for each of the five morphologies and re-evaluated in all other morphologies. For example, the left-most plot presents the average task performance of the fittest controller evolved in morphology 1 and re-evaluated on morphologies $2-5$. Where as, the right-most plot presents the average task performance of the fittest controller evolved given morphology 5 (as the initial sensory configuration) and re-evaluated on morphologies $1-4$.

To gauge the impact of a given team morphology (table I) in company with a given level of task complexity (table III), the *t-test* [62] ($p < 0.05$), was applied in pair-wise comparisons between sets of controller evolution results[3] (figures 4, 5 and 6, left column). Within each given level of task complexity, no statistically significant difference was found between controllers evolved given morphologies $1-5$ (controller evolution experiments $1-5$). Controller evolution experiments $1-4$ were those implementing controller evolution in fixed sensory configurations (morphologies $1-4$). Where as, controller evolution experiment 5 used morphology 5 as the initial sensory configuration and subsequently co-adapted behavior (controller) and morphology (complement of sensors). The lack of statistical difference between controllers evolved given morphologies $1-4$ (table I) indicates that these sensory configurations were not sufficiently different so as to result in significantly different average maximum task performances. Also, the lack of any significant difference between the average maximum task performance of controllers evolved given morphologies $1-4$ and behavior-morphology co-adaptation (starting with morphology 5), supports previous results demonstrating that behavior-morphology co-adaptation yields at least comparable task performance benefits (compared to fixed morphology controller evolution) in collective behavior tasks [22].

However, to address this study's main objective it was necessary to ascertain the morphological robustness of the fittest controller evolved in each morphology when re-evaluated in all other morphologies. To gauge the morphological robustness of the fittest controllers evolved for a given morphology ($1-5$), and a given task complexity, we applied the t-test in pair-wise comparisons of two result data sets. First, the average maximum task performances yielded by controller evolution in morphologies $1-5$ and second, the average maximum task performances yielded from re-evaluating the fittest controller evolved for a given morphology in all other morphologies (section IV-D).

Statistical test results indicated no significant difference (with one exception) between average task performance results yielded by controller evolution and morphological re-evaluation experiments for all task complexity levels. Specifically, the average maximum task performance yielded by controllers evolved given morphologies 2-5 (figures 4, 5, 6, left column) was not significantly lower than the average task performance yielded by the fittest controllers (evolved in morphologies $1-5$), and then re-evaluated on other morphologies (figures 4, 5, 6, right column). The exception was morphology 1 in task complexity levels 2 and 3. In these tasks, the fittest controller evolved for morphology 1, yielded a significantly higher average task performance than that yielded when this fittest controller was re-evaluated in morphologies $2-5$.

Hence, these results indicate that controllers evolved by HyperNEAT for a given morphology (table I), overall have the capacity to continue to effectively operate when transferred to other morphologies. This result was found to hold for all four of the five morphologies that controllers were evolved for, and for all levels of task complexity tested (table III).

The efficacy of HyperNEAT for evolving morphologically robust controllers is further supported by the controller evolution experiments that used morphology 5 (section IV). In this case, the number of sensors was adapted meaning that team behavior and morphology were co-adapted. Specifically, these controller evolution experiments began with the sensory configuration of morphology 5 (table I) and enabled and disabled sensor connections to better couple morphology with the evolved controller. Hence, the fittest controller evolved in this case often corresponded to a sensory configuration dissimilar to morphology 5 (the initial sensory configuration). Results indicated that the fittest controller evolved for morphology 5, when re-evaluated in other morphologies, yielded an average task performance that was statistically comparable to the average task performance yielded by controller evolution given morphology 5. This result was observed for all three task complexity levels (figures 4, 5, and 6, right column).

Thus, controllers evolved for fixed morphologies ($1-4$, table I), were found to be *morphologically robust*, as there was no significant difference in average maximum task performance when the fittest controller (evolved for morphology $1-4$), was re-evaluated in other morphologies. Furthermore,

[3]Statistical test results for pair-wise comparisons for the fittest evolved controllers (for a given morphology) tested in each other morphology (individually) is online at: https://github.com/not-my-name/SSCI_Paper_Appendix

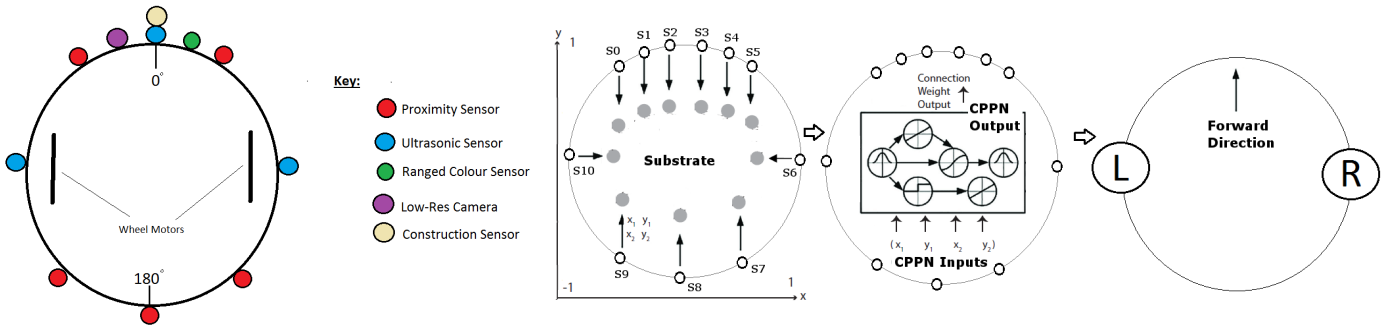

Fig. 1. *Left (color):* Robot morphology 1, with relative positions of various sensors on the robot. *Right (gray-scale):* ANN Topology as it relates to robot morphology 1: 11 Sensory inputs [S0, S10]. Sensory inputs connect to a hidden layer (left). Connection weight values between two nodes (x_1, y_1, x_2, y_2) are evolved by querying the CPPN (center) with x, y values in the range [-1, 1] (axis shown). The hidden layer is fully connected to all inputs and outputs (connectivity not depicted). Motor outputs (right) *L* and *R* determine the speed of the left and right wheels, respectively, and thus a robot's speed and direction.

TABLE I. SENSORY CONFIGURATION (NUMBER OF SENSORS) FOR EACH MORPHOLOGY.

Morphology ID	Proximity Sensors	Ultrasonic Sensors	Color Ranged Sensors	Low-Resolution Camera	Construction Zone Sensors
1	5	3	1	1	1
2	4	1	1	1	1
3	0	0	1	1	1
4	2	0	1	1	1
5	2	2	1	1	1

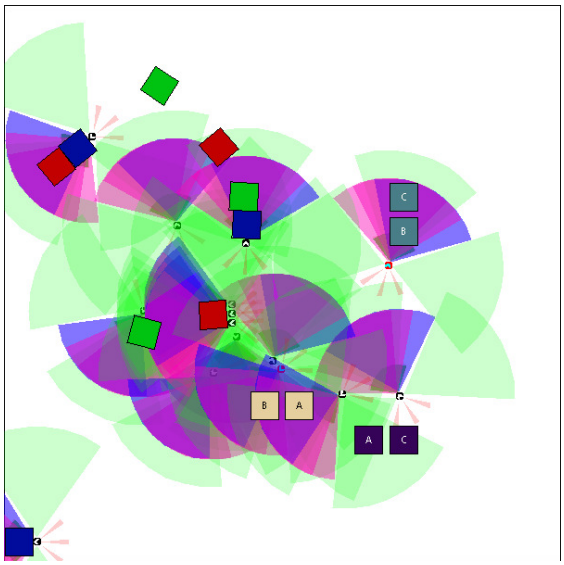

Fig. 2. Example of the simulation environment. Robots search for randomly distributed type A, B, and C blocks (blue, green and red, respectively). Other colored and labeled blocks indicate those already connected in construction zones. Different coloured semi-circles emanating from each robot represent the field of view of currently active different sensor types (table II).

Fig. 3. Task level 3 construction schema: *A*, *B*, and *C* are the block types. The label on each side of each block type indicates what block type can be connected to this side. An *X* label indicates that no block can be connected.

the fittest controller evolved for an adaptive morphology (5, table I), was similarly found to be morphologically robust, given that the average maximum task performance yielded when this fittest controller was re-evaluated in other morphologies ($1-4$), was comparable to the average maximum task performance yielded from morphology 5 controller evolution experiment. This is theorized to be a result of the complexity of co-adapting effective controller-morphology couplings [63] within limited periods of artificial evolution (100 generations in these experiments, table II), offset by the transference of evolved *connectivity patterns* [64] as functional controllers across varying robot morphologies [65]. Such connectivity patterns encode behaviors that do not rely upon specific sensory-motor mappings in controllers and thus do not necessitate specific task environment configurations, such as specific numbers of agents or objects. This in turn facilitates the transfer of controllers across varying team morphologies [66], [67], [68].

These results are corroborated by related work [65], [21], and contribute further empirical evidence that HyperNEAT yields significant benefits in evolving robot controllers that effectively operate in other morphologies. That is, this study further demonstrated HyperNEAT's capability to exploit geometric properties such as regularity, repetition and symmetry in robot morphology and environment [28], where such modularity and geometric properties are encoded in evolved connectivity patterns. This is prevalent in this study, as the configuration of sensors on each robot's periphery was symmetrical for all morphologies tested (section **??**). Also, the collective construction task required that blocks be connected together in a repeated manner in a symmetrical bounded simulation environment (20x20 units, table II). The capability of HyperNEAT evolved controllers to operate in different morphologies is further supported by other research [66] demonstrating that evolved indirect sensory-motor mappings

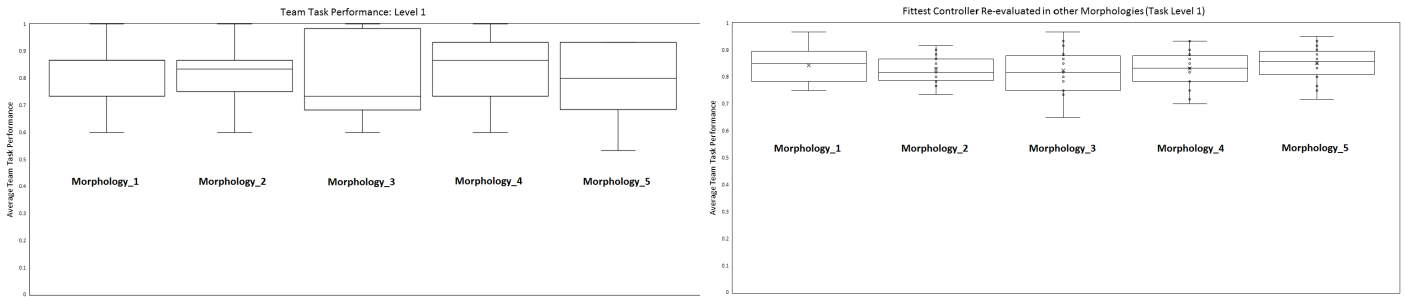

Fig. 4. *Left column:* Average team task performance for controller evolution (*task level 1*) given morphologies $1-5$ (depicted from left to right). *Right column:* Average task performance given the fittest controller evolved for each successive morphology ($1-5$, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology 1, re-evaluated in morphologies $2-5$. Right-most plot is the average task performance of fittest controller evolved for morphology 5, re-evaluated in morphologies $1-4$.

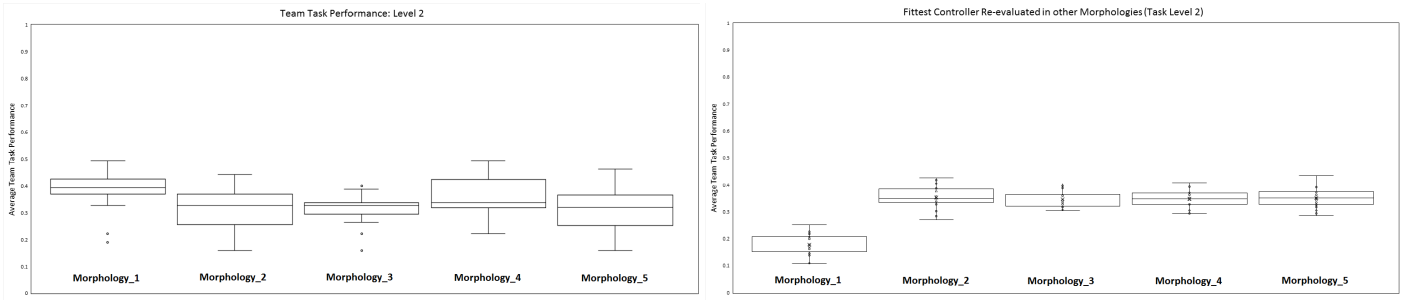

Fig. 5. *Left column:* Average team task performance for controller evolution (*task level 1*) given morphologies $1-5$ (depicted from left to right). *Right column:* Average task performance given the fittest controller evolved for each successive morphology ($1-5$, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology 1, re-evaluated in morphologies $2-5$. Right-most plot is the average task performance of fittest controller evolved for morphology 5, re-evaluated in morphologies $1-4$.

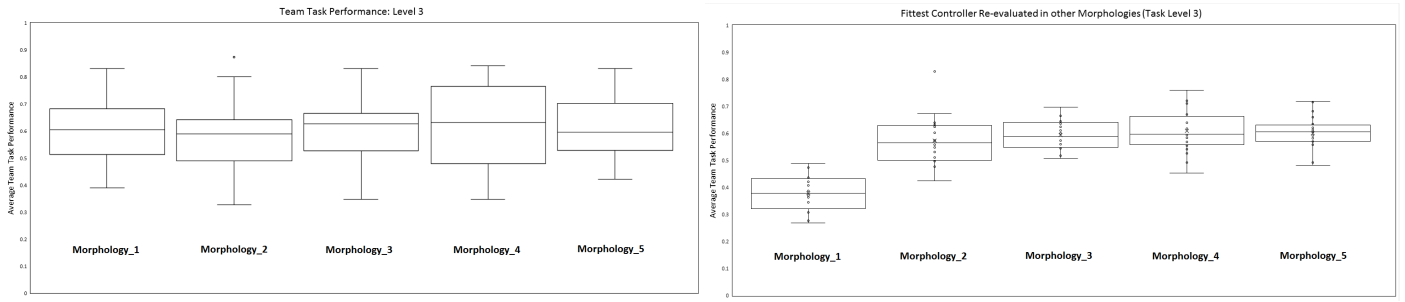

Fig. 6. *Left column:* Average team task performance for controller evolution (*task level 1*) given morphologies $1-5$ (depicted from left to right). *Right column:* Average task performance given the fittest controller evolved for each successive morphology ($1-5$, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology 1, re-evaluated in morphologies $2-5$. Right-most plot is the average task performance of fittest controller evolved for morphology 5, re-evaluated in morphologies $1-4$.

can encapsulate effective behaviors with relatively few task environment and robot geometric relationships, such as desired positions and angles between robots and different object types.

The efficacy of HyperNEAT for evolving morphologically robust controllers for collective behavior tasks of varying complexity is also supported by related research in *multi-agent policy transfer* [66], [67], [68]. Policy transfer methods facilitate the transfer of behaviors across tasks of increasing complexity or between dissimilar tasks. Such studies have demonstrated that HyperNEAT is an effective method for evolving behaviors in one collective behavior task and then transferring the evolved behavior to a related but more complex task (for example, where robots have more complex sensory-motor configurations to process increased task complexity) with relatively little loss in average team task performance.

However, we hypothesize that the morphological robustness of HyperNEAT evolved controllers demonstrated across all morphologies tested (table I) and all levels of task complexity (table III) was facilitated by the use of morphologically and behaviorally homogenous teams. Specifically, one controller was evolved for all robots in a team and all robots used the same sensory configuration, meaning all robots had the same *collective behavior geometry* [69]. This in turn simplified the transfer of evolved controllers across varying morphologies with no significant degradation in average task performance.

Hence, overall, this study's results demonstrate that HyperNEAT is an appropriate method for evolving morphologically robust controllers. That is, controllers that are fully functional in a range of team morphologies. In order to ascertain how well HyperNEAT evolved controllers generalize, ongoing research is evaluating the evolution of morphologically robust behav-

TABLE II. EXPERIMENT, NEURO-EVOLUTION AND SENSOR PARAMETERS

Generations	100	
Sensors per robot	11, 8, 4, 6, random	
Evaluations per genotype	5	
Experiment runs	20	
Environment length, width	20	
Max Distance (Robot movement per iteration)	1.0	
Team size	15	
Team Lifetime (Simulation iterations)	1000	
Lifetimes per generation	5	
Type A blocks (1 robot to push)	15	
Type B blocks (2 robots to push)	15	
Type C blocks (3 robots to push)	15	
Mutation rate	Add neuron	0.25
	Add connection	0.008
	Remove connection	0.002
	Weight	0.1
Population size	150	
Survival rate	0.3	
Crossover proportion	0.5	
Elitism proportion	0.1	
CPPN topology	Feed-forward	
CPPN inputs	Position, delta, angle	
Sensor	Range	FOV
Proximity Sensor	1.0	0.2
Ultrasonic Sensor	4.0	1.2
Ranged Colour Sensor	3.0	1.5
Low-Res Camera	3.0	1.5
Colour Proximity Sensor	3.0	3.0

TABLE III. TASK COMPLEXITY. NOTE: TASK LEVEL 3 INCLUDES A CONSTRUCTION SCHEMA (FIGURE 3).

Construction Task Complexity	Level 1	Level 2	Level 3
Type A blocks (1 robot to push)	15	5	5
Type B blocks (2 robots to push)	0	5	5
Type C blocks (3 robots to push)	0	5	5
Construction schema	No	No	Yes

iors in behaviorally and morphological heterogenous teams for complex collective behavior tasks that are irregular and without repetition or symmetry. Furthermore, current research is comparing HyperNEAT to related evolutionary approaches that have demonstrated controllers able to accomplish multiple disparate tasks in dynamic environments [70], as well as direct encoding neuro-evolution methods such as NEAT [49].

IX. CONCLUSIONS

This research presented a study on the efficacy of HyperNEAT for evolving *morphologically robust* behaviors for homogenous robot teams that must solve a collective behavior task of increasing complexity. That is, the average maximum task performance of behaviors evolved for a given team morphology (robot sensory configuration) that was then transferred to a different team morphology. Controllers that did not yield degraded task performance when transferred to another morphology were considered to be morphologically robust. The objective was to test and evaluate methods that generate morphological robust behaviors, where varying morphologies emulated sensor damage or intentional changes to the sensory systems of robotic teams.

Results indicated that HyperNEAT was appropriate for generating morphologically robust controllers for a collective construction task of increasing complexity. This task required robots to cooperatively push blocks such that they connected together to form structures. Task complexity was regulated by the number of robots required to push blocks and a construction schema mandating that specific block types be connected in specific ways. These results support the notion that developmental neuro-evolution methods, such as HyperNEAT, are appropriate for controller evolution in robotics applications where robot teams must adapt during their lifetime to damage or otherwise must dynamically adapt their sensory configuration to solve new unforseen tasks.

REFERENCES

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

[2] 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*, vol. 2, pp. 25–38, 2012.

[3] G. Nitschke, M. Schut, and A. Eiben, "Evolving behavioral specialization in robot teams to solve a collective construction task," *Swarm and Evolutionary Computation*, vol. 2, no. 1, pp. 25–38, 2012.

[4] J. Gomes, P. Urbano, and A. L. Christensen, "Evolution of swarm robotics systems with novelty search," *Swarm Intelligence*, vol. 7, no. 2-3, pp. 115–144, 2013.

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

[6] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[7] 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*. IEEE, 2006, pp. 2787–2794.

[8] 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*. ACM, 2012, pp. 217–224.

[9] R. Kube and H. Zhang, "Collective robotics: from social insects to robots," *Adaptive Behaviour*, vol. 2, no. 2, pp. 189–218, 1994.

[10] G. Beni, "From swarm intelligence to swarm robotics," in *Proceedings of the First International Workshop on Swarm Robotics*. Santa Monica, USA: Springer, 2004, pp. 1–9.

[11] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[12] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. Eiben, "Evolutionary robotics: what, why, and where to," *Frontiers in Robotics and AI — Evolutionary Robotics*, vol. 2(4), pp. 1–18, 2015.

[13] H. Lipson and J. Pollack, "Automatic design and manufacture of robotic life forms," *Nature*, vol. 406, no. 1, pp. 974–978, 2000.

[14] H. Lund, "Co-evolving control and morphology with lego robots," in *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, H. Fumio and R. Pfeifer, Eds. Berlin, Germany: Springer, 2003, pp. 59–79.

[15] G. Buason, N. Bergfeldt, and T. Ziemke, "Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments," *Genetic Programming and Evolvable Machines*, vol. 6(1), pp. 25–51, 2005.

[16] J. Auerbach and J. Bongard, "Environmental influence on the evolution of morphological complexity in machines," *PLoS Computational Biology*, vol. 10(1), p. e1003399. doi:10.1371/journal.pcbi.1003399, 2014.

[17] C. Mautner and R. Belew, "Evolving robot morphology and control," *Artificial Life and Robotics*, vol. 4(3), pp. 130–136, 2000.

[18] G. Hornby and J. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artificial Life*, vol. 8(3), pp. 1–10, 2002.

[19] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Amsterdam, Netherlands: ACM Press, 2013, pp. 167–174.

[20] Y. Asai and T. Arita, "Coevolution of morphology and behavior of robots in a multiagent environment," in *Proceedings of the SICE 30th Intelligent System Symposium*. Tokyo, Japan: The Society of Instrument and Control Engineers, 2003, pp. 61–66.

[21] J. Watson and G. Nitschke, "Evolving robust robot team morphologies for collective construction," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. Cape Town, South Africa: IEEE, 2015, pp. 1039–1046.

[22] J. Hewland and G. Nitschke, "Evolving robust robot team morphologies for collective construction," in *The Benefits of Adaptive Behavior and Morphology for Cooperation in Robot Teams*. Cape Town, South Africa: IEEE, 2015, pp. 1047–1054.

[23] R. O'Grady, A. Christensen, and M. Dorigo, "Swarmorph: Morphogenesis with self-assembling robots," in *Morphogenetic Engineering, Understanding Complex Systems*, R. Doursat, Ed. Berlin, Germany: Springer-Verlag, 2012, pp. 27–60.

[24] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345(6198), pp. 795–799, 2014.

[25] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314(5802), pp. 1118–1121, 2006.

[26] A. Cully, J. Clune, D. Tarapore, and J. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521(1), pp. 503–507, 2015.

[27] R. Brooks and A. Flynn, "Fast, cheap and out of control: A robot invasion of the solar system." *Journal of the British Interplanetary Society*, vol. 1, no. 1, p. 478485, 1989.

[28] K. Stanley, D'Ambrosio, and J. Gauci, "Hypercube-based indirect encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 1, pp. 185–212, 2009.

[29] P. Verbancsics and K. Stanley, "Constraining connectivity to encourage modularity in hyperneat," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2011, pp. 1483–1490.

[30] D. D'Ambrosio and K. Stanley, "Scalable multiagent learning through indirect encoding of policy geometry," *Evolutionary Intelligence Journal*, vol. 6, no. 1, pp. 1–26, 2013.

[31] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Berlin, Germany: Springer, 2003.

[32] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Efficient non-linear control through neuroevolution," in *Machine Learning: ECML 2006*. Berlin, Germany: Springer, 2006, pp. 654–662.

[33] J. Wawerla, G. Sukhatme, and M. Mataric, "Collective construction with multiple robots," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2002, pp. 2696–2701.

[34] J. Watson and G. Nitschke, "Deriving minimal sensory configurations for evolved cooperative robot teams," in *Proceedings of the IEEE Congress on Evolutionary Computation*. Sendai, Japan: IEEE, 2015, pp. 3065–3071.

[35] G. S. Nitschke, M. C. Schut, and A. E. Eiben, "Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task," *Evolutionary Intelligence*, vol. 3, no. 1, pp. 13–29, 2010.

[36] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

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

[38] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[39] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[40] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[41] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[42] J. E. Dayhoff and J. M. DeLeo, "Artificial neural networks," *Cancer*, vol. 91, no. S8, pp. 1615–1635, 2001. [Online]. Available: http://dx.doi.org/10.1002/1097-0142(20010415)91:8+{\textless}1615::AID-CNCR1175{\textgreater}3.0.CO;2-L

[43] R. Miikkulainen, "Neuroevolution," in *Encyclopedia of Machine Learning*, C. Sammut and G. Webb, Eds. New York, USA: Springer, 2010, pp. 716–720.

[44] ——, "Evolving Neural Networks," in *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 2441–2460. [Online]. Available: http://doi.acm.org/10.1145/1830761.1830902

[45] F. J. Gomez, D. Burger, and R. Miikkulainen, "A neuro-evolution method for dynamic resource allocation on a chip multiprocessor," in *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, vol. 4. IEEE, 2001, pp. 2355–2360.

[46] 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*, vol. 15, no. 3, pp. 346–367, 2011.

[47] F. J. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," in *IJCAI*, vol. 99, 1999, pp. 1356–1361.

[48] C. Igel, "Neuroevolution for reinforcement learning using evolution strategies," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4. IEEE, 2003, pp. 2588–2595.

[49] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation.*, vol. 10, no. 2, pp. 99–127, 2002.

[50] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines: Special Issue on Developmental Systems*, vol. 8, no. 2, pp. 131–162, 2007.

[51] S. Ficici and J. Pollack, "Challenges in coevolutionary learning: Armsrace dynamics, openendedness, and mediocre stable states," in *Proceedings of the Sixth International Conference on Artificial Life*. Cambridge, USA: MIT Press, 1998, pp. 238–247.

[52] G. S. Hornby, S. Takamura, T. Yamamoto, and M. Fujita, "Autonomous evolution of dynamic gaits with two quadruped robots," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 402–410, 2005.

[53] 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*. IEEE, 2009, pp. 2764–2771.

[54] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE transactions on Robotics and Automation*, vol. 19, no. 4, pp. 703–719, 2003.

[55] F. Gruau, "Automatic definition of modular neural networks," *Adaptive behavior*, vol. 3, no. 2, pp. 151–183, 1994.

[56] K. Sims, "Evolving 3d morphology and behavior by competition," *Artificial life*, vol. 1, no. 4, pp. 353–372, 1994.

[57] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343(6172), pp. 754–758, 2014.

[58] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *The International Journal of Robotics Research*, vol. 27(3-4), pp. 463–479, 2008.

[59] D. D'Ambrosio and K. Stanley, "Generative encoding for multiagent learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Atlanta, USA: ACM Press, 2008, pp. 819–826.

[60] F. Lambercy and J. Tharin, *Khepera III User Manual: Version 3.5*. Lausanne, Switzerland: K-Team Corporation, 2013.

[61] R. Beckers, O. Holland, and J. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," in *Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems*. Cambridge, USA: MIT Press, 1994, pp. 181–189.

[62] B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes*. Cambridge, UK: Cambridge University Press, 1986.

[63] R. Pfeifer and J. Bongard, *How the body shapes the way we think*. Cambridge, USA: MIT Press, 2006.

[64] J. Gauci and K. Stanley, "Autonomous evolution of topographic regularities in artificial neural networks," *Neural Computation journal*, vol. 22, no. 7, pp. 1860–1898, 2010.

[65] S. Risi and K. Stanley, "Confronting the challenge of learning a flexible neural controller for a diversity of morphologies," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Amsterdam, The Netherlands: ACM, 2013, pp. 255–261.

[66] P. Verbancsics and K. Stanley, "Evolving static representations for task transfer," *Journal of Machine Learning Research*, vol. 11(1), pp. 1737–1769, 2010.

[67] S. Didi and G. Nitschke, "Hybridizing novelty search for transfer learning," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. Athens, Greece: IEEE Press, 2016, pp. 10–18.

[68] ——, "Multi-agent behavior-based policy transfer," in *Proceedings of the European Conference on the Applications of Evolutionary Computation*. Porto, Portugal: Springer, 2016, pp. 181–197.

[69] D. D'Ambrosio, J. Lehman, S. Risi, and K. Stanley, "Evolving policy geometry for scalable multi-agent learning," in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*. Richland, USA: ACM Press, 2010, pp. 731–738.

[70] E. Izquierdo-Torres and T. Buhrmann, "Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: Walking and chemotaxis," in *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Artificial Life IX)*. Winchester, UK: MIT Press, 2008, pp. 257–264.