Computer Science Honours
Final Paper
2015

Title: Adaptive Sensory Configurations in Robot Technologies [Co-Evolution]

Author: Ntokozo P. Zwane

Project Abbreviation: ASCiRT

Supervisor: Dr Geoff Nitschke

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	0
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	17
System Development and Implementation	0	15	10
Results, Findings and Conclusion	10	20	18
Aim Formulation and Background Work	10	15	15
Quality of Paper Writing and Presentation	10		10
Adherence to Project Proposal and Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	0
Total marks		**80**	

Adaptive Sensory Configuration in Robot Teams

[Morphology Evolution]

Ntokozo Zwane
18 University Avenue
University of Cape Town
Rondebosh
Cape Town, 7700
Ntokozo.Zwane@alumni.uct.ac.za

ABSTRACT

Extensive research has been done in the field of Evolutionary Robotics with the aim to design a robust controller, for a multi-agent based system, that exhibits behaviours that are crucial to complete a given task. In this series of papers, we discuss the potential advantage of achieving better agents when evolving both the controller and the agent body [1]. In order to test for significance of this approach, we compared it to one other approach where the morphology of the robot agents was evolved (with a static predefined controller). In this paper we explore the morphological evolution of robot agents.

We found that the morphology evolution approach did find some good solutions to the problem, however, these solutions were best suited for low complexity tasks only. The coevolution approach outperformed the morphology evolution approach in all tasks of different complexity levels.

CCS Concepts

•**Computing methodologies** → **Evolutionary robotics;** *Genetic algorithms;* Neural networks; Evolvable hardware;

Keywords

evolutionary computing, multi-agent systems

1. INTRODUCTION

There has been an increase in work aimed at using robot teams to complete a range of non-trivial collective behaviour tasks. The task we pay particular attention to is a search and retrieval (foraging) task. The task is constructed such that the team of robots have to make use of work cooperatively to efficiently complete the task. In the real world, such a task would involve a significant amount of noise and incomplete information which would require the team of robots to adapt to the environment through a suitable learning process. We would thus expect our team of robots to exhibit some form of Artificial Intelligence (AI) [29]. The concepts of Evolutionary Computation (EC) are used to achieve this goal, which builds the foundations of Evolutionary Robotics (ER).

EC covers a range of methods that create control systems that exhibit a desired behaviour. For the other approach - which involve an evolving controller - Neuro-Evolution (NE) is a suitable EC method. It has shown to have high performance in control tasks of this nature. Extending this further, a traditional Genetic Algorithm (GA) is coupled with the NE approach to incorporate a broader level of control over the design process - namely, the body design of the robots.

Historically, there has been a weighty focus on agent controller evolution, and little work done to explore the evolution of the sensor configuration (morphology) of the agents. We simulated the behaviour of a team of robots with adapting morphologies for each robot's static, pre-defined controller. To keep the simulated work closely related to a real world scenario, we desire a solution that requires the least number sensors while maximising the efficiency, thus minimising the cost and energy consumption of each robot.

2. BACKGROUND

2.1 Agent Design

2.1.1 Controller Design

A prominent problem in the field of robotics is that of *controller* design - the 'brain' controlling the robots. Controllers should possess the ability to efficiently complete the prescribed task. The other approach makes use of a flavour of EC that has shown promise, by modelling the controllers using similar constructs to biological central nervous systems, in attempt to inherit the adaptive features of these constructs. These computational modelas are called Artificial Neural Networks (ANNs) [22]. They are compromised typically of several interconnected process units (*neurons*) which have a given number of inputs and outputs.

In general, ANNs can be thought of as a directed graph, where the edges between neurons are weighted according to how much influence they have on the neurons connected to outgoing edges (see Figure 1).

The learning methods used for an ANN control model are based on a subset of the Darwinian evolution principles, called Evolutionary Algorithms (EAs) [1]. This subset that revolves around the evolution of ANNs is called Neuro-Evolution. Useful algorithms or algorithm parameters are found by iteratively selecting and refining a population of possible solutions.

2.1.2 Morphology Design

[1]Thought this text, an agent body represents its sensory configuration. The term agent *morphology* and agent body are used interchangeably

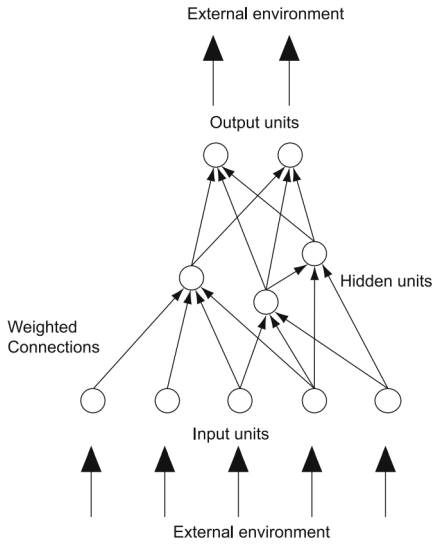

Figure 1: A generic neural network architecture. It consists of input and output nodes which are connected to the external environment, and hidden nodes which connect to other neurons but are not directly connected to the environment [7].

In this investigation, we pay particular consideration to the physical constructs of an agent. The design process aims to find a robust and cost efficient design for the robot morphologies. These morphologies are defined by an agent's sensory configuration - how many sensors, where they are placed, and in which direction they are facing etc. The Evolutionary Computing learning method used to construct a morphology best suited to the problem. This requires a good representation of the robot bodies such that a suitable EA can be used to find the parameters that achieve the task. A Genetic Algorithm has shown to be best suited in this type of task [13], since it provides best convergence for a multi-dimensional problem of this nature [20].
The foraging task that the team of robots completed required cooperation between the robots, since some of the resources to be collected required multiple robots to return it to the collection area. This type of cooperation between multiple robots is called a *multi-agent system* (MAS) [23].

2.2 Machine Learning

The field of AI tries to understand intelligent entities and build them based on the level of understanding reached. There are multiple definitions of what AI is, Russel and Norvig [29] have grouped some of these into four categories: systems that think like humans, systems that act like humans, systems that think rationally, systems that act rationally.
Machine Learning (ML) is a subfield of AI that is concerned with programs that learn from experience. ML is applied to the design of controllers where the task is too complex for human-designed procedures or a set of heuristics.

2.2.1 Algorithms for Agent Body-Brain Design

In order to develop some form of unguided behaviour, there has to be a process of learning to help develop such behaviour. There are three different categories to learning. These three categories are divided by the type of input and output given to the learning method, as well as the feedback given to the method.

ALGORITHM 1: The pseudo-code for a general Evolutionary Algorithm [5].

```
begin
    INITIALIZEpopulation with random candidate solutions;
    EVALUATEeach candidate;
    while TERMINATION CONDITIONnot is satisfied do
        SELECTparents;
        RECOMBINEpairs of parents;
        MUTATEthe resulting offspring;
        SELECTindividuals for the next generation;
    end
end
```

2.2.2 Supervised Learning

In supervised learning, the algorithms are given sets of inputs and the corresponding, correct output data. On each iteration, the task is to find a function that closely represents the relationship between the input and output values. In unsupervised learning, the task is to predict the future percepts using the previous ones, with no correct outputs given to the algorithm.
Considering problems revolving around designing controllers for a team of robots in an environment, using supervised learning is undesirable since the sets of input and output data are very large, thus, we turn towards another approach to learning: reward-based learning.

2.2.3 Reward-based Learning

Reward-based learning has two main sub-categories, reinforcement learning [27] and stochastic learning.
Algorithms trained using the reinforcement learning procedure approximate the output function values using previous estimates, and are given feedback in the form of a reward or penalty depending on how they perform in the environment [27]. Reinforcement learning relies on the principles of dynamic programming to estimate the value (*utility*) resulting in acting on a particular state.
Stochastic search methods do not estimate value functions, they instead learn behaviours directly. A strategy that has been found to be effective in generating control parameters and decision rules for robots which display collective behaviour is Evolutionary Computing (EC) [5].

2.2.4 Unsupervised Learning

Unsupervised approaches, particularly stochastic search methods, are more favourable since they provide a means to cover a large parameter search space without the shortfall of being as highly computationally expensive as reinforcement learning would be. This thus brings us into the realm of Evolutionary Algorithms [5].

2.2.5 Evolutionary Algorithms

Evolutionary Algorithms are a subfield of EC and are aimed at adapting solutions based on the concepts of Darwinian evolution.
The Darwinian process of selection and survival of the fittest is applied to a set of candidate solutions (*populations* or *individuals*) by refining them through processes of selection,

recombination and mutation at each iteration (*generation*) of the algorithm. These processes are known as genetic operators, the populations will be refined on each iteration until a termination condition is met, which is usually when either an individual that fits the criteria is found or a certain amount of time has elapsed (see Algorithm 1).

Evolutionary Algorithms are divided into several classes, which are categorized by the data structure used to represent the solutions. Genetic Algorithms (GAs) [9] - mentioned in Section 2.1.2, Evolutionary Strategies (ESs) [6], Genetic Programming (GP) [12] and Evolutionary Programming (EP) [8].
GAs and ESs are generally used for searching multidimensional space, GAs represent their individuals by using a bitstring, and ESs a real-valued vector. GPs evolve computer programs by refining the expression trees representing them, and EPs evolve the state tables for finit e state machines (FSMs) - machines with state tables that can be written as a truth table to represent the transitions between states.

2.3 Morphology Evolution

EAs have been successfully used in the construction of artificial neural networks for a variety of applications. One direct application of EAs is in the field of evolutionary robotics, where EAs are used to design controllers for the robots. These designs do work well in general, however, they have limited performance due to the number, placement, quality, efficiency and reliability of the sensors that robots are fitted with [2].

The most basic task that any multi-agent system needs to fulfil is some form of locomotion [19]. The paper by McGeer [19] demonstrates that the basic locomotion for a robot can be achieved by carefully designing the robot body. McGeer created a two-legged machine that is able to walk down a sloped terrain without any instructions from a controller. This paper demonstrates how crucial it is to select the correct body structure - as well as its parameters - in creating a machine that realises locomotion. The only driving forces for these machines is gravity, therefore emphasizing how the mechanics (body) of the machine play a weighty role in the overall effectiveness [19].

In the paper by Lungarella and Sporns [18], they investigate the effects of an organism's morphology on how the information it receives is processed and quantified. Three types of robotic platforms were used to test their hypothesis: a fixed humanoid robot, four mobile agents, and a simulated wheeled robot. In all platforms, which were given some form of vision and motor feedback, the robots induced a consistent structure of the information flow between the robot's sensor and motors.

These fundamental studies indicate the importance that lies in the selection of an agent's morphology. However, in most studies, robots have been carefully constructed with a fixed morphology. The paper by Bongard [3] shows that there is a drastic increase in performance when an evolutionary process is applied to morphology selection. Bongard's approach to test this theory was to evaluate the effectiveness of two differently evolved organisms. One of the organisms had the controller and morphology evolved with an initial,

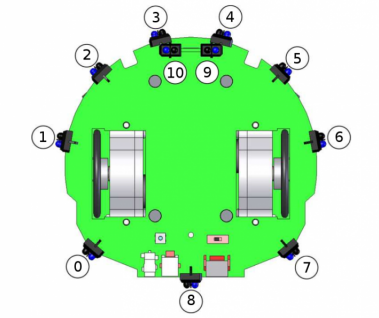

Figure 2: Physical and schematic view of the Khepera robot. The numbers 0 Through to 8 indicate the standard sensor positions, and $9 - 10$ indicate the positions of the bottom proximity sensor. The simulations were based on this robot and the sensors that are configurable for it [28].

legged, body. The other organism was evolved with initially an anguilliform - the form of an eel - body plan. The results showed that those organisms that started off with the anguiliform evolved to more robust robots, as opposed to the initially legged robots.
This suggests that morphological change is an important process in producing robust controllers [3]. It was also shown that the complexity of the bodies was influenced by the complexity of the environment the organisms were exposed to.

To truly test the effectiveness of evolving the agents' bodies, Pfeifer [25] analysed the evolution of agent bodies for 10 different evolved agent controllers. Pfeifer found that agents that had strong physical characteristics (eg. mass, number of legs etc.) of a certain type had large behavioural differences to agents with characteristics of a different type [25], which supports the idea of including morphological evolution in the agent training process.
Balakrishnan and Honavar [2] showed that far fewer sensors were used by the evolutionary system than the number made available - this behaviour was observed without any penalties given for use of an excessive number of sensors. Balakrishnan and Honavar [2] suggested that introducing sufficient noise to the system will decrease the chances of robots having a low team fitness, due to them getting stuck in repetitive or cyclic paths, and these designs can be successfully transferred on to real world robots.
The problem lies in attempting to redefine the sensor characteristics, as opposed to sensor positions and how many of each type is present. The sensor characteristics may be confined to the underlying architecture of the physical sensor, and its capabilities may be limited. We do, however still have full freedom to modify the positions, number and types of sensors on each iteration, the only constraints we thus took into considerations with that respect are: the size of the robot (how many sensors can fit on it, see Fig. 2), and a reasonable level of power consumption. Sensory system design allows us to determine efficient, often counter-intuitive, designs for the control mechanisms, and this EA-based design can easily be extended further to cope with different types of sensors. Balakrishnan and Honavar [2] suggests further that sensory system design is necessary in designing robots that can handle hazardous and unpredictable environments (e.g.

Environment dimensions		
Component	**Dimensions (cm^2)**	**Mass (kg)**
Environment	200 x 200	-
Target area	200 x 40	-
Small resource	40 x 40	1
Medium resource	60 x 60	2
Large resource	80 x 80	3
Robot body dimension		
Body diameter	30 cm	-
Mass	-	0.7 kg
Wheel diameter	6 cm	-
Maximum speed	0.5 m· s^{-1}	-

Table 1: The simulator component dimensions (top), and the robot body dimensions (bottom) as configured by Hewland and Nitschke [11]

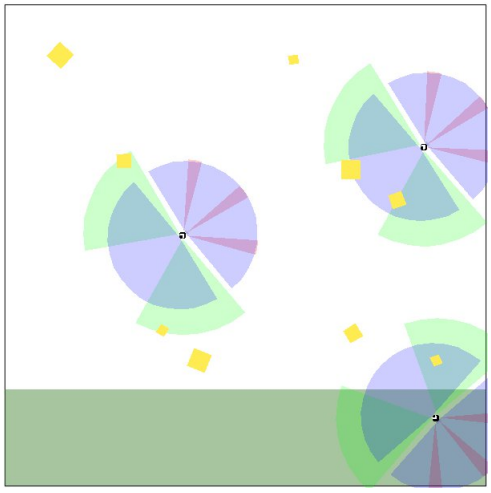

Figure 3: The simulation environment: the yellow blocks are the resource objects (to be collected by the agents) and the black circles are the agents, with the different sensors forming the conical shapes - indicating their range and field of view.

nuclear waste clean-up, toxic chemical handling processes, space exploration etc. [24]), where sensor failures might occur quite frequently.

The result of evolution of visual sensing morphologies and sensory-motor controller-networks for visually guided robots presents a number of networks that result from using incremental evolution with variable-length genotypes [4]. There are, however, two, of the many, networks - emerging from an evolution approach of this form - that display striking characteristics. Even though behavioural observations of these two networks are the same, the sensing morphologies and sensory-motor controllers are very different. Which means that this evolutionary process converges at a behavioural level, and is coupled with divergent evolution at the morphological level. The process of evolution used here holds many similarities to analysing biological nervous systems in the field of *neurothology*[4].

3. METHODS

3.1 Simulation Design

3.1.1 Simulation Environment

The simulation environment has been configured to accurately recreate the task as though it were executed in a real world setting. These types of virtual environment simulators are often called *agent-based simulation models* (ABMs). The simulator we used is an extension of the existing simulation environment written by Hewland and Nitschke [11]. The MASON [16, 15] ABM was chosen as the foundation of the simulator platform since it had underlying advantages (covered in detail by Hewland and Nitschke [11]) over other existing ABM platforms. The JBox2D [21] physics engine is used with MASON to provide the simulator with a variety of tools in the aim to make it more realistic with respect to the real environment.

The simulation environment was constructed to accommodate for the task that was used to test the agents, for both effective completion of the task, and the use of cooperation to completing the task efficiently. The dimensions of the environment have not been changed from those configured by Hewland and Nitschke [11] (see Table 1). The environment was a square field, surrounded by four walls and a rectangular target area (the drop-off point for the resources - see Figure 3).

There are three different type of resources - small, medium and large resources - to aid in testing for cooperation between the agents. The small resources can be collected by a single agent, medium resources by a minimum of two agents, and large resources by a minimum of three agents (i.e larger resources are heavier). The dimensions for the resources are included in Table 1.

The design of the robots is aimed at closely simulating the Khepera III robots (see Figure 2), with respect to their size, shape and weight (see Table 1). The force exerted on the wheels were also closely matched to those possessed by the Khepera III robot.

3.1.2 Agent Morphology

The agent morphology is how we configure the sensors on each agent. Any sensor on an agent is described by four properties (see Table 2), which were selected by the EA. The sensors are modelled as conical fields coming out from the agent's exterior, placed along the circumference.

The parameters in Table 2 define the morphology of an agent (see Figure 4). In Section 3, we will define how these parameters were encoded for the EA.

Three types of sensors were modelled for the purpose of the experiment, each with their own unique properties. Throughout the evolutionary process, the sensor parameters are confined to the bounds defined in Table 3,

The more powerful sensors are considered as more costly, and thus, should be used more sparingly by the robots. The three sensors increase in cost from the (1) Infrared Proximity (IR) Sensor being the least costly, followed by the (2) Ultrasonic Sensor (US) in cost, and followed lastly by the (3) Long Range IR Sensor (LIR) - being the most costly sensor.

Parameter	Valid range	Description
Bearing	$[-\pi, \pi]$ rad.	The position of the sensor on the robot's exterior (along the circumference) orientated with the robot's forward directional axis.
Orientation	$[-\frac{\pi}{2}, \frac{\pi}{2}]$ rad.	The offset of the sensor's forward direction, relative to the outward normal from the agent surface (at the sensor's position).
Range	$(0, 10^12)$ m	The maximum distance of the sensor's vision.
Field of view	$(0, \pi]$ rad.	The angle of the sensor's conical arch.

Table 2: The parameters that describe each sensor [11].

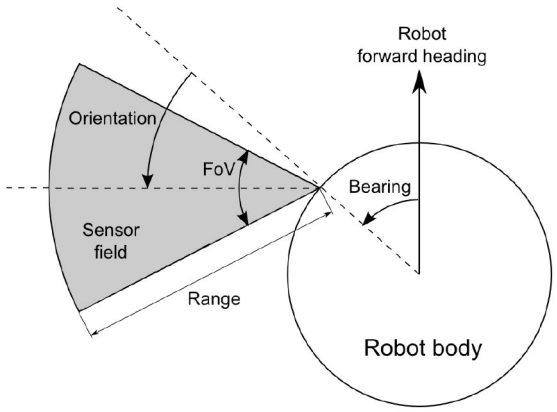

Figure 4: The parameters that define the morphology of the agents [11].

Parameter	Valid Range
Infrared Sensor Range	$(0, 4]$m
Ultrasonic Sensor Range	$[2, 3]$m
Long Range IR Sensor Range	$(0, 6]$m
Infrared Sensor Field of View	$(0, \pi]$ rad.
Ultrasonic Sensor Field of View	$(0, \frac{\pi}{2}]$ rad.
Long Range IR Sensor Field of View	$(0, \frac{\pi}{3}]$ rad.
Infrared Sensor Orientation	$[0, \pi]$ rad.
Ultrasonic Sensor Orientation	$[0, \pi]$ rad.
Long Range IR Sensor Orientation	$[0, \pi]$ rad.
Infrared Sensor Bearing	$(0, 2\pi]$ rad.
Ultrasonic Sensor Bearing	$(0, 2\pi]$ rad.
Long Range IR Sensor Bearing	$(0, 2\pi]$ rad.

Table 3: The valid parameter rangers for the different type of sensors simulated for the Kepera III robot.

Each agent can have a maximum of eight Infrared Sensors, four Long Range Infrared Sensors and two Ultrasonic Sensors, thus a total of fourteen sensors in total (which of course is the least desirable solution).

3.2 Experiment Design

3.2.1 Defining The Controller

The controller for the agents was kept fixed throughout the evolutionary process. Ideally, an ANN would be preferred to represent the controller, however, to decouple this paper's work with that done in the other approach (coevolution) a heuristic based controller was used.
The heuristics of the agent were defined such that maximum environment exploration was achieved using obstacle avoidance heuristics. A set of base heuristics was used in both approaches to govern how agents handle interactions with objects in the environment. These was used to give the agents obstacle avoidance, as well as the resource objects collection capabilities.

3.2.2 Evolving The Morphology

In this section, we will describe how the morphology of the robot will be represented to adopt the GA approach in evolving the morphology of the agents. We need to encode the *phenotype* or *phenome* (the physical characteristics of the agents) with a *genotype* or *genome* (the genetic material of the phenotype [5]).
In the paper by Lee *et al.*[13], they studied the effects of evolving both the controller and the morphology of their organisms. They, used a different structure, and thus a different evolutionary algorithm class, to represent and evolve their controller and morphology. Genetic Programming was used for the controller, and a Genetic Algorithm for their body. Lund [17] followed a very similar GA-GP approach to solve a similar problem, in which they evolved LEGO robot brain and bodies in order to achieve optimal performance. In our approach, we too use a GA for the evolution of the bodies (morphologies) of our agents.
The Encog Machine Learning Library [10] was used to construct the GA used for our experiments.

3.2.2.1 The GA Genome.

The traditional GA approach is to represent the genome using a bit-string. The evolutionary operators are defined such that they mutate the bit-string genomes by flipping bits. Our problem is too complex to be encoded by a regular bit-string. Instead, a variation of the Genetic Algorithm will be used. One in which the genome is represented by a real-valued string with a finite language of real numbers. This is motivated by the paper by Lichtensteiger and Eggenberger[14], where a robot's morphology is evolved and the controller is a fixed, predefined neural network. Here they use Evolutionary Strategies to encode the genotype of the robot's morphology, which falls in the class of EAs that represent their genomes in real-valued vectors (see Sec-

tion 2.2.5). In Lichtensteiger and Eggenberger's [14] paper, their robot successfully adapted its sensor morphology (its compound eye) to achieve relatively good performance when tasked with estimating a critical lateral distance to an object.
In our approach, the hybrid GA-ES genotype was used to encode the structural parameters (see Section 3) that define an agent. These parameters are what define the structure of an agent's morphology. Therefore, evolving these parameters results in the evolution of robot morphologies. One more parameter that was considered is the sensor on/off parameter, since part of the evolutionary process is determining which sensors are or are not required for optimal and enhanced performance.

Theses parameters can be arranged as a linear string, which contains a real number at each position. The finite language of each parameter (*gene*) will be any x such that $x \in [-1:1]$ (this choice is motivated in Section 3.2.2.3). Thus, the resulting genome is a string of numbers in the range $[-1:1]$. The robot's morphology can then be represented as

$$G_0 G_1 G_2 \dots G_{n-1} \tag{1}$$

where

$$-1 \leq G_i \leq 1;\ 0 \leq i \leq n-1$$

Each parameter will be represented by a certain G_i in the string. The five parameters that the string will represent are the bearing, orientation, range and field of view (which will be abbreviated to BROF), and the sensor on/off parameter, which is appended to the abbreviation (SBORF). All the SBORF parameters for each sensor type will be concatenated together, to result in a genome of the form

$$\begin{aligned} &S_1^a \dots S_n^a B_1^a O_1^a R_1^a F_1^a \dots B_n^a O_n^a R_n^a F_n^a \dots \\ &\dots S_1^b \dots S_m^b B_1^b O_1^b R_1^b F_1^b \dots B_m^b O_m^b R_m^b F_m^b \end{aligned} \tag{2}$$

For n and m sensors of sensor types a and b respectfully and S_1^a, for example, is the gene G_0. This representation is for only two sensor types (type a and type b in this case), however, it is extended to accommodate for all three available sensor types.

3.2.2.2 The GA Phenome.

The genome is mapped to the phenotype using a one-to-one mapping function to ensure that each genome has a unique phenotype. A function that behaves this way is the hyperbolic tangent (tanh) function,

$$G_i := \tanh(x_i) \text{ where } x_i \in [-\pi : \pi] \tag{3}$$

To decode the genome, we simply apply the inverse hyperbolic tangent function to the gene and perturb the result into the range we need it to be,

$$P_i := \mathrm{Re}\left[\alpha \cdot \operatorname{atanh}(G_i) + \beta\right] \tag{4}$$

where α and β are our perturbation constants, such that,

$$\alpha \in \mathbb{R} \text{ and } \alpha \geq 1,\ \beta \in \mathbb{R}$$

	Co-Operation	Small	Medium	Large
1	Minimum	10	5	0
2	Medium	5	5	5
3	Maximum	0	5	10

Table 4: The resource configuration for each simulation run. The experiments are referred to as the small configuration medium configuration and large configuration respectively.

For values of $G_i = 1$ or -1, the inverse hyperbolic tangent function yields a complex number, thus only the real part of this value will be used for the decoding process.

3.2.2.3 Mutation Operators.

Two mutation operators were used, a two-point crossover and one-point mutation, to create the next the morphologies for the individuals in the next generation. Lee *et al.*[13] found that these two were sufficient to provide rapid convergence. The crossover operation used is the standard two-point crossover which involves two genomes, as parents, and two crossover points from those parents. This crossover has a slight variation. It performs two sub-operations of exchanging or averaging at random. The exchanging operation swaps the parents' genes G_i that fall between the crossover points, and the averaging operation replaces those genes with the average of the two corresponding G_i from each parent.
The parameters described by the genome (defined in Section 3.2.2.1) are highly dependent on the gene position. Consider the parameter string (2) where real valued ranges - corresponding to the ranges in Table 2 - are used in the string to represent each gene. If we apply the exchange two-point crossover operation to this genome, the resultant genome will no longer be a map to a valid phenome, since the parameters could possibly fall out of the bounds of the parameter range. To that end, it is thus crucial to define the genome with a language of numbers that is bound by the same limits (1).
The mutation operator replaces a randomly chosen G_i with a new randomly generated gene value. The probability of the mutation operators was inspired by the work done by Lund [17], where the probabilities were set to $p_c = 0.25$ for the crossover probability, and a mutation rate of $p_m = 0.04$, which they found through empirical tests.

4. RESULTS AND DISCUSSION

In order to acquire significant results, each GA run lasted for a duration of 250 generations. In each generation, statistical data was acquired from running 3 simulations (epochs) - where agents are placed into the environment for a fixed amount of time - for three sets of resource configurations (see Table 4). The small experiment configuration gives the task the lowest level of complexity, and increases in the medium configuration. The large configuration is the experiment with the largest level of complexity. The level of complexity is gauged by the level of complexity required to complete a task. Tasks that require more cooperation are expected to be more difficult for the robots to complete.

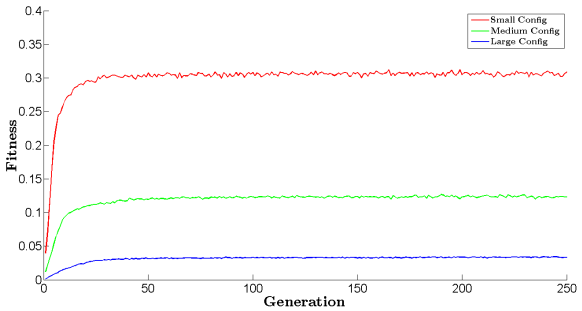

Figure 5: The average of the best team fitness scores for each generation of the evolutionary process for each simulation configuration. The fitness has been normalized such that 1.0 is the maximum possible task performance.

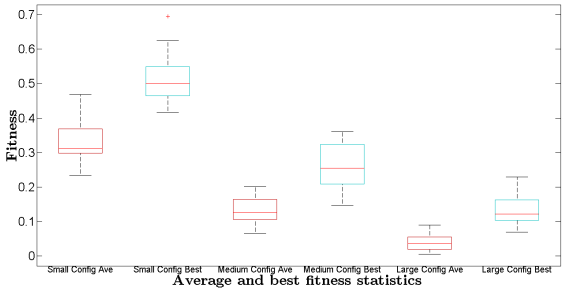

Figure 6: Box plots for average (red) and best (blue) team fitness for the small, medium and large configuration experiments (from right to left) respectively. The fitness has been normalized such that 1.0 is the maximum possible task performance.

4.1 Evolved Robot Morphologies

The results showed that agents that have only their morphology evolved are best suited for tasks that require no co-operation to accomplish the task. A statistical analysis of the data sets supports this result. In order to conduct a two-tailed t-test [26], a ks-test[26] was done to confirm that the data sets are log-norally distributed. The data sets for the small and medium configurations are log-normally distributed, with p-values $p = 0.97$ and $p = 0.73$ respectfully. The dataset for the large configuration is not log-normally distributed ($p = 0.13$), however it has normally distributed ($p = 0.67$). We will make the assumption that a much larger dataset for the large configuration experiments, it would be log-normally distributed.
There are statistically significant differences between the small vs. medium, small vs. large, and medium vs. large configurations (with $p < 0.0001$ for all three tests). This thus supports the result that the small configuration truly does outperform the other two experimental configurations for the morphological evolution approach.

4.1.1 Team Fitness

In Figure 5 is the average of the best fitnesses for each generation, of each experiment configuration. The graph in Figure 5 follows the typical behaviour of a generic evolutionary algorithm (*anytime behaviour* [2]). The fitness of the algorithm should ideally tend toward an optimal (or even suboptimal) solution. For each set of experiments, the solution conformed to the expected anytime behaviour, implying that the best solution for each initialisation was found within the 250 generations.
Figure 5 shows that fitness scores for the experiments with the small configuration are significantly higher than that of the medium and large configuration experiments. It is clear that the experiments that required cooperative behaviour between the agents resulted in poor team fitness scores. The level of cooperation required is inversely proportional to team fitness scores achieved.

[2]This effect is called anytime behaviour because evolutionary algorithms may be stopped at 'any time' during the evolutionary process and have some solution - regardless of how optimal the solution is [5].

The box plot in Figure 6 shows the average of the best, and average of the mean team fitnesses for the experiments described in Table 4. Each experiment configuration was run twenty times. Here we see again that the fitness decreases as the task complexity increases.

4.1.2 Agent Morphologies

In the experiment design, no penalty was employed for an agent using as many sensors as it could. The cost of 'expensive' sensors was also not factored into the evolutionary algorithm. Figure 8 is a box plot showing the average number of each type of sensor on the agents. There were three different types of sensors (see Table 2), (1) the Infrared Sensor, (2) the Ultrasonic Sensor, and the (3) Long Range Infrared Sensor, and there was a maximum of eight, four and two of each type of sensor (respectively) that could be simultaneously attached to an agent (i.e fourteen sensors in total).
In this graph, we see that the EA did not converge towards morphologies that consisted of the maximum number of sensors allowed. Instead, individuals in the GA population that had solutions with a high number of sensors did not survive for long periods in the evolutionary process. This can be seen on Figure 8, where the black lines show the mean selected number of sensors, which is consistently close to half the total number of sensors throughout all three experiments (small, medium and large configurations). The source of this result is likely to be the level of noise that the environment sends to the agents.
An agent with a morphology that is composed of a high number of sensors is likely to perform poorly . This is due to the reason that agents with plenty sensors would become subject to a lot of noise from the environment, thus making it harder for the agent to determine what its next movements in the environment should be. This result is consistent with the findings of similar work by other researchers [2]. A number of researchers strongly believe that a noisy system is crucial to give rise to robots that can be transferred to a real environment [4].

On average, the EA converged to a solution in early generations. This can was seen on Figure 5. In Figure **??** shows the average convergence generation for each exper-

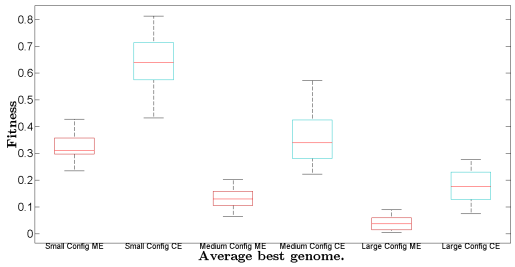

Figure 7: Box plots showing average of the team fitnesses of the best genomes of each experiment run. The averages for the experiments run with the morphology evolution (ME) implementation (red) are compared to those of the experiments run with the coevolution (CE) implementation (blue). The fitness has been normalized such that 1.0 is the maximum possible task performance.

iment [3]. The box plots are close to symmetric, and the median is just below halfway along the y-axis, which implies that a large majority of the generations converged close to halfway (at ∼150 generations) through the full duration of the evolutionary process. This relatively early convergence is called *premature convergence*. The EA converging prematurely implies that the solutions found are subomptimal. This is caused by a number of factors: two of which are the genetic operators not being able to produce superior individuals, or the genetic operators were not exploratory enough to search the entire search space.

4.2 Approach Comparison

In this section, we will analyse and compare the results of the two approaches for robot design, namely co-evolution and morphology evolution. A Neuro-Evolution method called NEAT-M [11] was used to evolve the agent body and controller. This method is an extension of NEAT, which has been adapted to include morphology evolving operators. The experiments run with the NEAT-M implementation were configured the same as those for morphology evolution GA implementation. With the exception of the number of experiments executed (fifteen runs for each experiment configuration). Thus only fifteen of the experiments from the this paper's approach will be used in this comparison.
The datasets for the experiments executed in the both implementations are normally distributed (see ks-test results in Table 5), and are statistically different (t-test yields p-value $p << 0.0001$). This statistical analysis supports the result we observed, in which the coevolution approach greatly outperforms the morphology evolution approach.

The box plot it in Figure 7 shows the differences in the average best genome for the two different approaches. On average, the fitnesses of the best genomes from the coevolution approach are better than those of the morphology evolution approach. For the small and medium experiment configurations, the worst team fitness of the coevolution is approximately the same as the best from the morphology evolution fitness.
For all three experiment configurations, coevolution surpassed morphology evolution. This means that regardless of the complexity of the task, coevolution produces more robust robot agents. The relatively lower fitness of the morphology evolution approach can be directly linked to the fact that its heuristic based controller is unable to make the agent make more reasonable choices.
The ANN present in the coevolution approach helps the controllers adopt a certain behaviours which promote cooperation between the agents. This is a property that the morphology evolution agents do not possess. This is the drawback that the static controller has, in that it cannot adapt to new and complex environments. An agent's movements is controlled only by what it sees, and no lasting behaviours arise in their movements. An evolving robot controller allows the agents to adopt behaviours that can not arise from a static one.
We can see that the fitnesses of the morphology approach had converged towards a particular maxima. This is evident in the fact that the data presented in the box plots (in Figure 7) are not sparsely distributed. This same observation can be applied to the coevolution datasets, which are distributed over a larger range. This implies that the coevolution experiments had not yet reached a plateau of convergence. This is a result of having run the algorithm for too few generations before terminating it.

5. CONCLUSIONS

In this paper, we simulated a robot environment in which agents had to complete a search and retrieval task. The agents were given a limited lifetime to retrieve resource objects (from an unseen environment) and return them to a known home area. The environment had resource objects that differed in sizes (weight), where larger boxes required a larger number of agents pushing it simultaneously to be able to move it to the home area. This promoted the requirement the team of agents possess some form of cooperative behaviour.
The agents had heuristic controllers, and a the morphology was evolved in order to find an set of sensory parameters that will significantly improve the agents' efficiency in resource collection. We compared this approach to finding an optimal agent morphology to how it performed in different experiment configurations, which required different levels of cooperation from the agents. A particularly interesting result is that, in general, the surviving genomes were not those that had the maximum number of sensors attached activated. This was observed despite having no penalty being included in the fitness function. This indicates that there is a natural preference for solutions that require only the required number of sensors to complete the task. This emphasizes the significance in evolving the morphologies of the team of robots.
Results from the experiments showed that this approach to agent design was best suited for the tasks that required the least amount of cooperation to complete the task. We found that the the average of the team fitnesses of the best genomes of each experiment run was inversely proportional to the level of complexity of the experiment. A t-test analysis confirmed that the morphology evolution approach preformed better in the small configuration (least complex), since there was a statistical difference between the results when tested

[3]See Figures (1), (2) and (3) on http://people.cs.uct.ac.za/~zwnnto004/ntokozo/appendix.pdf for the resulting morphologies of the best genomes from the small, medium and large configurations respectively

	Morphology Evolution				Co-Evolution		
	Small	Medium	Large		Small	Medium	Large
Minimum	28.130841	7.829699	0.736246		51.983333	26.67789	11.466667
Maximum	51.292835	24.153686	10.771028		97.518467	68.691151	33.308333
Standard Deviation	6.31070214	4.64953286	3.07774354		11.8338345	11.79189	6.65126441
Normality Test (p-value)	0.57	0.9	0.68		0.94	0.89	0.92
Log-normalit y test (p-value)	0.82	0.71	0.49		0.76	0.97	0.98

Table 5: The ks-test was conducted to check for normality in the distributions of the fitness statistics for the two approaches.

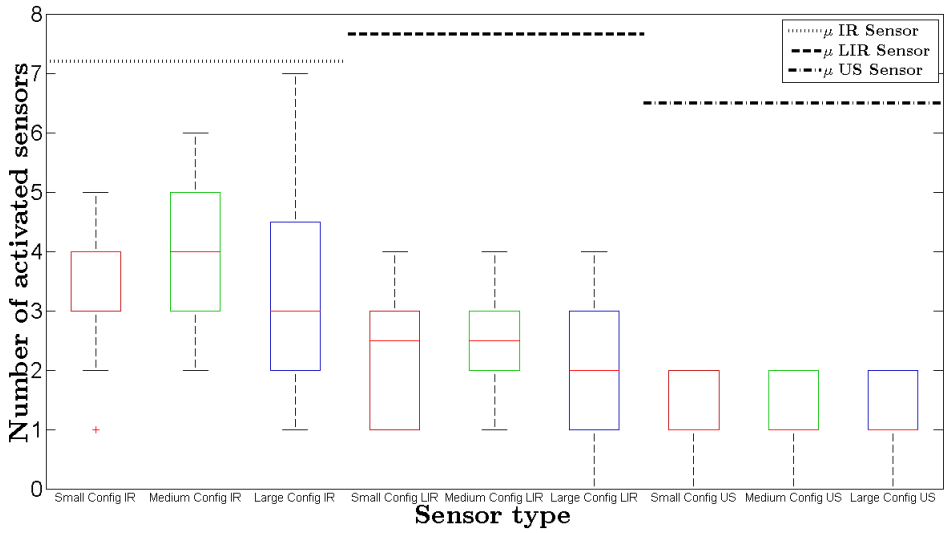

Figure 8: Box plots showing the average selection of each type of sensor in each experiment: the small configuration (red), medium configuration (green) and large configuration (blue). The dotted, dashed and dash-dot black lines show the mean (μ) total number of sensors of the IR, LIR and US sensors respectively for each experiment run (the curve of the average for a given sensor type spans (horizontally) the width of the box plots for that sensor type. The number of sensors has been normalised

against the medium and large configurations.
This approach was compared to a different robot design, in which both the controller and morphology of the agents was evolved. The results showed that the coevolution approach out preformed the morphology evolution approach on all task complexity levels.
The morphology evolution experiments converged prematurely. This is an unexpected result can only be determined after a number of successive algorithm runs. A future improvement could be made to increase the algorithm's exploration of the solution space, such as increasing the population size, or tuning the mutation probabilities and selection pressures.
On the opposite end of premature convergence, the coevolution approach did not settle into a convergence point, thus running these experiments for more iterations would likely lead to the EA finding more robust solutions.

In conclusion, we confirm our hypothesis, that cooevolution of both agent controllers and morphologies simultaneously provides more robust agent designs. This result is supported by the t-test, which told us that the dataset from the morphology evolution approach is significantly different from that of the coevolution approach.

6. ACKNOWLEDGEMENTS

We thank our supervisor Dr. Geoff Nitschke. To the current team ASCiRT: J. Jang and N. Ganey. To the previous team ASCiRT of 2014 for providing the foundations for our research. To the 2015 Honours course convenor A/Prof. Michelle Kuttel.

7. REFERENCES

[1] T. Back. *Evolutionary algorithms in theory and practice*. Oxford Univ. Press, 1996.

[2] K. Balakrishnan and V. Honavar. On sensor evolution in robotics. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 455–460, Cambridge, MA, USA, 1996. MIT Press.

[3] J. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239, 2011.

[4] D. T. Cli, P. Husbands, and I. Harvey. Analysis of evolved sensory-motor controllers. In *CSRP-264, COGS, University of Sussex. Also to appear in Proceedings of Second European Conference on Artificial Life (ECAL93), Brussels, May*, pages 24–26, 1993.

[5] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.

[6] M. Eigen. *Ingo Rechenberg Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. mit einem Nachwort von Manfred Eigen, Friedrich Frommann Verlag, Struttgart-Bad Cannstatt, 1973.

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

[8] D. B. Fogel, L. J. Fogel, and V. Porto. Evolving neural networks. *Biological cybernetics*, 63(6):487–493, 1990.

[9] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[10] J. Heaton. Encog machine learning framework, 2014. URL http://heatonresearch.com/encog.

[11] J. Hewland and G. Nitschke. Adaptive robot team behavior and morphology for collective gathering. In *Computational Intelligence, 2015.,Proceedings of the IEEE Symposium Series on,*. IEEE Press, 2015.

[12] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[13] W.-P. Lee, J. Hallam, and H. H. Lund. A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 384–389. IEEE, 1996.

[14] L. Lichtensteiger and P. Eggenberger. **Evolving the morphology of a compound eye on a robot**. In *Proceedings of the Third European Workshop on Advanced Mobile Robots*, pages 127–134. IEEE Press, 1999.

[15] S. Luke. Multiagent simulation and the mason library. *George Mason University,*, 2011.

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

[17] H. H. Lund. Co-evolving control and morphology with lego robots. In *Morpho-functional Machines: The New Species*, pages 59–79. Springer, 2003.

[18] M. Lungarella and O. Sporns. Mapping information flow in sensorimotor networks. *PLoS computational biology*, 2(10):e144, 2006.

[19] T. McGeer. Passive dynamic walking. *the international journal of robotics research*, 9(2):62–82, 1990.

[20] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[21] D. Murphy. Jbox2d: A java physics engine. *accessed August*, 2, 2014.

[22] S. Nolfi and D. Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[23] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3), Nov. 2005.

[24] G. B. Parker. Co-evolving model parameters for anytime learning in evolutionary robotics. *Robotics and Autonomous Systems*, 33(1):13–30, 2000.

[25] J. Pfeifer. A method for isolating morphological effects on evolved behaviour. In *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, volume 7, page 305. MIT Press, 2002.

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, et al. *Numerical Recipes*. Cambridge UP Cambridge etc, 1986.

[27] A. G. Richard S Sutton and Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[28] C. Robotics. Como robotics - khepera iii, 2010. URL http://ai.vub.ac.be/robotics/wiki.

[29] Russell and Norvig. A modern approach. *Intelligence, Artificial*, 2003.