The Impact of Sensory Distortion on the Evolution of Robots in Forage Tasks Requiring Cooperation

HSXSHU003

October 29, 2014

Contents

List of Figures

Chapter 1

Introduction

The work done in this project was one of three different experiments conducted under the same simulation platform. We used the field of Evolutionary Robotics (ER) to provide both a practical context for our research, as well as background theory on approaches to the simulation. This project studied the performance of robotic agent teams in a simulated cooperative foraging task, where the robotic agents did not communicate with each other using any explicit message-passing protocols. It aimed to investigate the impact of augmenting the strength of various environmental cues to the robotic agents on the evolutionary development of cooperative behaviour. Specifically, the research studied the effects of varying the linear transducer gain on typed sensors on the evolution of controllers that perform well on a cooperative foraging task. The motivation was that distorting the environmental cues to the robot, abstracted as complements of transducer gains on robotic agents, may aid in the evolution of cooperative behaviour.

The research presented in this paper aims to use simulated ER to answer three questions:

- Can artifically distorting the sensory input of agents increase performance on a foraging task that requires cooperation?
- Are the beneficial effects of 1 affected by the team composition and reward structure?
- Are the beneficial effects of 1 consistent over tasks requiring increasing levels of cooperation?

Experiments equipped simulated robotic agents with seven sensors arranged in fixed positions around the robot body. Three of the seven sensors were typed sensors that were able to receive signal from one type of object in the environment only: other robots, resource objects or the target area. Each typed sensor could take an additional linear parameter that would augment or diminish the signal processed by the sensor. The placement of these sensors (morphology) was fixed and different complements of linear parameters for the typed sensors were tested. The controller representation was chosen to be Artificial Neural

Networks (ANNs), with input nodes corresponding to environment sensors on the robots, and the output nodes providing actuator values for the left and right wheel drives on the robots. Neuro-Evolution of Augmenting Topologies, a popular Neuro-Evolution algorithm, was then used as a basis to evolve controllers for these morphologically homogeneous robots in a simulated collective foraging task.

The degree to which cooperative behaviour arises in teams composed of differing behaviours is explored by using both genetically (with regards to ANN encoding) homogeneous and genetically heterogeneous teams of agents. Global reward was used for the homogeneous teams and local reward was used for the heterogeneous teams. The robotic agents were tested on different variations of the task setup, each requiring differing degrees of cooperation to complete satisfactorily. Team task performance, along with the task setup variation, was used to measure the degree to which the robots had cooperated on the tasks.

Chapter 2

Background

2.1 Evolutionary Computation (EC)

EC is the discipline of solving computational problems with metaheuristics inspired by the biological processes of evolution (Eiben and Smith 2003). It is a sub-field of Machine Learning (ML) (Russell and Norvig 2003). In EC, algorithms are designed to iteratively design a model for a problem's solution based on input data and the desired outcome. Evolutionary Computation is most commonly applied when the global minimum or maximum of an optimisation problem's search space are unknown, making it impossible to use empirical optimisation methods such a gradient descent to reach a solution. Often the search space for such problems may be prohibitively large or present many local maxima (or minima), and would benefit from a directed, stochastic search rather than an exhaustive one. In EC, the pool of candidate solutions to an optimisation problem is compared to a population of organisms in nature. The encoded form of a candidate solution, upon which mutations, recombination and other evolutionary operations are performed, is referred to as the genotype of the individual. The input-output mapping of the individual within the test environment is referred to as the phenotype (Kicinger, Arciszewski, and Jong 2005). The refinement of these solutions is an iterative process involving repeated selection and perturbation of good solutions to obtain better solutions in successive generations (Darwin 1859; J. H. Holland 1975). A metaheuristic which describes the genotype representation, evolutionary operators and evolutionary parameters for these problem-solving approaches is referred to as an Evolutionary Algorithm (EA). The general form of an EA is described in Figure 1.

EAs may be applied to any problem that can be formulated as optimising on one or more functions (Sivanandam and Deepa 2007). Additionally, EAs may be combined with related disciplines such as Reinforcement Learning (Sutton and Barto 1998a) or supervised Machine Learning algorithms (Michie, Spiegelhalter, and Taylor 1994) to achieve desired results. The most well-established EAs are currently Genetic Algorithms (J. H. Holland 1975), Evolutionary Strategies (Rechenberg 1965), Evolutionary Programming (Fogel 1999) and Genetic Programming (Koza 1994).

Algorithm 1: The general form of the Evolutionary Algorithm (Kicinger, Arciszewski, and Jong 2005).

```
Initialize population of candidate solutions $P(0)$ ;
Evaluate all members of $P(0)$ ;
**for** $g := 1$ *to* $g_{max}$ *or Best Individual of* $P$ *is good enough* **do**
    Select parents from $P(g)$;
    Apply evolutionary operators (mutation and/or recombination) to parents ;
    Compose new population $P(g_{+1})$ ;
    Evaluate $P(g_{+1})$ ;
**end**
```

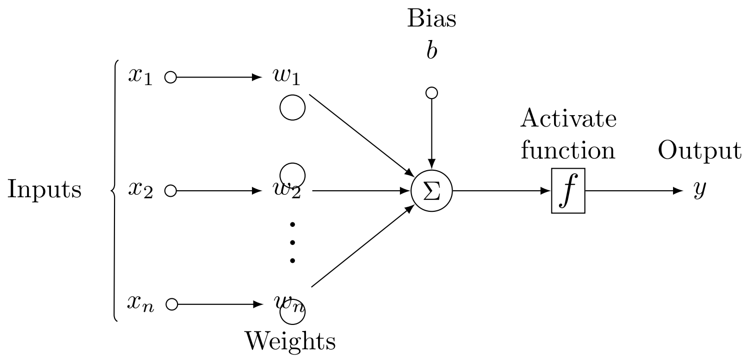


Figure 2.1: An Artificial Neuron.

2.1.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational models inspired by neural networks found in nature (McCulloch and Pitts 1943). They consist of a network of artificial neurons connected to each other via weighted edges. A positive connection weight indicates an excitatory connection, and a negative connection weight indicates an inhibitory connection.

A single neuron in an ANN is a simple mathematical function that multiplies a vector of inputs $(x_1, x_2, \ldots, x_n)$ with a vector of weights $(w_1, w_2, \ldots, w_n)$, sums the results, and passes the result to an activation function ϕ to produce an output value. The output value of an artificial neuron, y, is represented by Equation 2.1:

$$[h]y = \phi(\sum_{j=0}^{n} w_j x_j) \tag{2.1}$$

(Yao 1999)

An ANN maps the input received from a layer of input nodes into a collection of values at the output nodes. ANNs may be classified according to their topological properties.

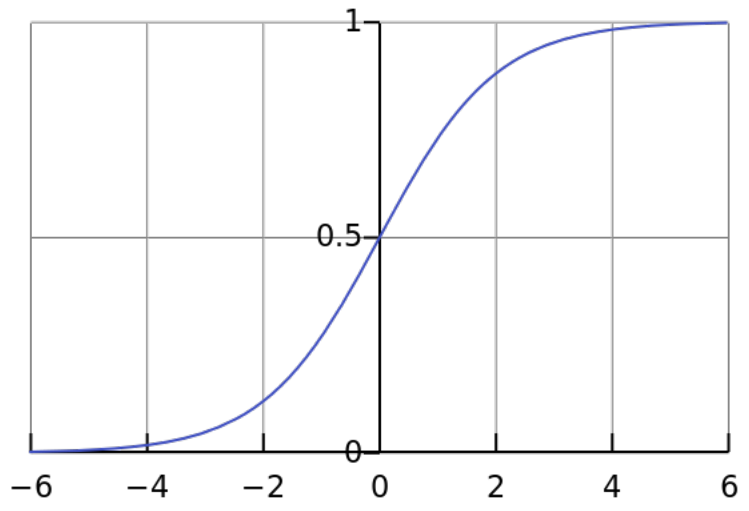


Figure 2.2: The Sigmoid activation function, $S(t) = \frac{1}{1+e^{-t}}$.(Wolfram—Alpha 2014)

Feed-forward networks are acyclic graphs of neurons directed from input to output, whereas recurrent networks allow cycles of connections. Each successive collection of nodes at the same depth in the input-output pipeline of the network makes up a layer in that network.

Additionally, ANNs may be classified according to the connection density of its nodes, where networks with all the nodes in each layer connected to all the nodes in its adjacent layer are "fully-connected" neural networks and those in which this is not the case are referred to as "partially connected".

The output of the kth neuron in an ANN may be described by Equation 2.2

$$y_k = \phi_k(\sum_{j=0}^{n} w_{kj}x_j - b_i) \tag{2.2}$$

(Yao 1999)

where the output y_k is calculated by subtracting the bias term b from the weighted sum of the inputs to the kth neuron, and passing this into the transfer function, ϕ_k. Some common transfer functions include sigmoid and step functions (**TODO**). The choice of activation function is specific to the nature of the problem domain (Duch and Jankowski 1999).

ANNs are used extensively in Machine Learning, and can be trained to model functions with techniques such as Backpropagation (Widrow and Lehr 1990), Reinforcement Learning (Sutton and Barto 1998b), Linear Regression (Specht 1991) and Neuro-Evolution, covered further in Section 2.1.2.

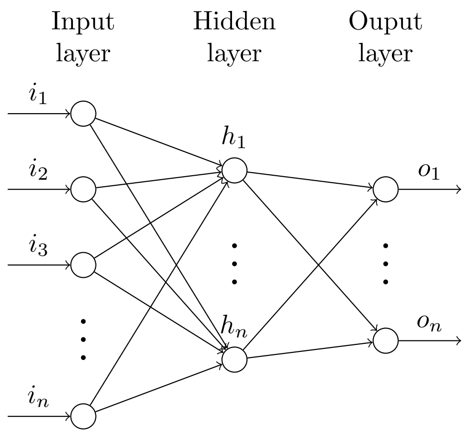


Figure 2.3: The Feed-Forward Network, a variety of ANN. The nodes in the network are connected in an acyclic fashion.

2.1.2 Neuro-Evolution

Neuro-Evolution is the process of applying an Evolutionary Algorithm to Artificial Neural Networks in order to obtain solutions to an optimisation problem (Miikkulainen 2010). Neuro-Evolution offers advantages over supervised learning approaches in certain domains, as it may offer satisfactory solutions without needing explicit input-output pairs. This is particularly relevant in situations in which no explicit input-output pairs can be procured, such as in unstructured tasks (ie. noisy task domains with incomplete sensory information) with autonomous agents.

The Genetic Algorithm (GA), as conceived by Holland, is an EA intended to develop complex adaptive systems (J. H. Holland 1975). Harvey, Husbands and Cliff argue for the appropriateness of GA for ER on the grounds that ER aims to find an agent behaviour for an unstructured environment rather than explicitly optimise a function (Harvey et al. 1997). A GA requires that the solution domain be represented as *genotypes*, a collection of symbols that may be manipulated. Each candidate solution, then, is an instance of such a genotype. Using an EA such as GA, to evolve Neural Networks is the basis of Neuro-Evolution. The genotype, a representation of the node and connection information, is mapped to a phenotype, the ANN representation of the genotype. The network itself is evaluated for its fitness while the evolutionary operators work with the genotype of the agent. The primary operators for generating new individuals within a GA are mutation and recombination.

Many GAs for NEs are designed to search for optimal networks that use fewer connections and neurons (Stanley and Miikkulainen 2002a). One way of optimising for smaller networks is to include a node-pruning operator in the EA (Sher 2011). Another way is to build up from a minimal topology, as used by algorithms such as NEAT (See Section 2.1.2).

Genotype representation in NE fall under two main approaches - direct and indirect (developmental) encoding.

Direct Encoding: We are chiefly interested in direct encoding methods for this research. Direct encoding refers to genotypes that directly represent the parameters of the neural network, such as node information and connection weights (Koehn 1994).

The GENITOR encoding strategy, proposed by Whitley, encodes each connection between neurons as fixed-length bit-strings (Whitley 1989). The index bit indicates whether or not the connection exists (is active) and the remaining bits encode the weight value of the connection. This is similar to the classical representation of GAs.

Montana Montana and Davis (1988) proposed a similar encoding to GENITOR, with the exception that the connection weights are encoded as real numbers instead of bit-strings .

A different strategy is Schiffman's node-based encoding. The genotype of Schiffman's method is a list of nodes and their connection information (Whitley 1989).

The encoding used in the GANNet algorithm is another node-based encoding, proposed by White White (1994). Similar to Schiffman, the parameter strings is a list of nodes and their connections. The chief differences between the two representations is that GANNet includes connection weight data, and enforces some restrictions on the types of networks represented. GANNet is used to represent feed-forward layered networks, with the number of inputs per neuron restricted to four.

Other direct encoding techniques include Layer-Based Encoding (Harp, Samad, and Guha 1989; Mandischer 1993) and Pathways-Based Encoding (Jacob and Rehder 1993).

Indirect Encoding: Genotypes using the indirect encoding scheme specify production rules by which network parameters may be generated (Koehn 1994). Cellular Encoding (CE) (Gruau et al. 1994) uses the cell-division metaphor to encode graph information (nodes and links) that undergo procedural divisions, resulting in a network of ANN nodes as terminals. CE has advantages in that it generates modular, hierarchichal sub-networks of neurons. This specification simplifies the task of overcoming the competing conventions problem described in Section 2.1.2 for the recombination of networks. Many indirect encoding schemes exploit the hierarchichal generation of structures in a similar manner. Kitano's grammar-encoding method procedurally rewrites symbols in a 2x2 matrix as more 2x2 matrices; each symbol represents a node and its connections (Kitano 1990). Another approach encodes a set of instructions into the genotype, which is used to grow and transform an initial network with one hidden node (F. Gruau 1992). It is also possible to define network production rules with a formal grammar, as in Lindenmayer-System encoding (Boers and Kuiper 1992; Lindenmayer 1968).

Topology and Weight-Evolving Artificial Neural Networks (TWEANN)

Neuro-Evolution methods may choose to optimise the connection weights of a fixed network topology (conventional NE), or evolve both connections weights and network topologies (TWEANN). The term Topology and Weight-Evolving Artificial Neural Networks was coined by Stanley et al. Stanley and Miikkulainen (2002b). Below we look at the three most influential approaches in TWEANN.

Neuro-Evolution for Augmenting Topologies (NEAT)

NEAT (Stanley and Miikkulainen 2002b) was designed to respond to three major challenges in Topology and Weight Evolving Artificial Neural Networks: matching genes during crossover, protecting emergent innovative genotypes from elimination, and preventing unnecessary bloat of evolved networks. NEAT has been demonstrated as leading to faster learning and more sophisticated solutions to optimisation problems than traditional NE. Tests showed that NEAT outperformed both CE and Enforced Subpopulations (ESP) (Gomez 2003) in the Double Pole-Balancing with Velocity optimisation task (Stanley and Miikkulainen 2002b).

NEAT consists of three major components:

1. An effective representation of ANNs for meaningful recombination
2. Preventing premature elimination of newly-formed networks
3. Minimising topologies to prevent bloat

1. An effective way of representing ANNs to allow for meaningful recombination

The competing conventions problem in Neuro-Evolution describes the difficulty of recombining parent networks that are structurally different but nevertheless represent the same input-output mapping. Crossing over two ANNs is possible out of the assumption that they consist of sub-networks, representing functional sub-units, that can be recombined. However, sub-networks at the same locus in two chromosomes may not represent the same contribution to the phenotype (Stanley and Miikkulainen 2002b). If two networks with functionally equivalent input-output mappings have differing sub-network structures, crossing over by swapping sub-networks may result in offspring that have multiple copies of a gene responsible for the same output and no copies of a gene for another. To solve this problem, the crossover operation must first determine which sub-networks correspond with each other in the parent genotypes.

In NEAT crossover, the offspring inherits the intersection of the two parents' genes as well the unmatched genes from the fitter parent. Instead of approaching the problem by using topological analysis of the networks, which is costly to implement on a large scale, Stanley et al. proposes using the historical origins of the mutations to see which genes match up. Tracking historical origin involves assigning each new mutation a globally-incremented 'innovation number', under the implication that ANNs which share nodes with the same innovation number necessarily descended from a common ancestral gene. The genes of two mating parents are matched as demonstrated in **??**, and sub-structures are recombined to produce the offspring.

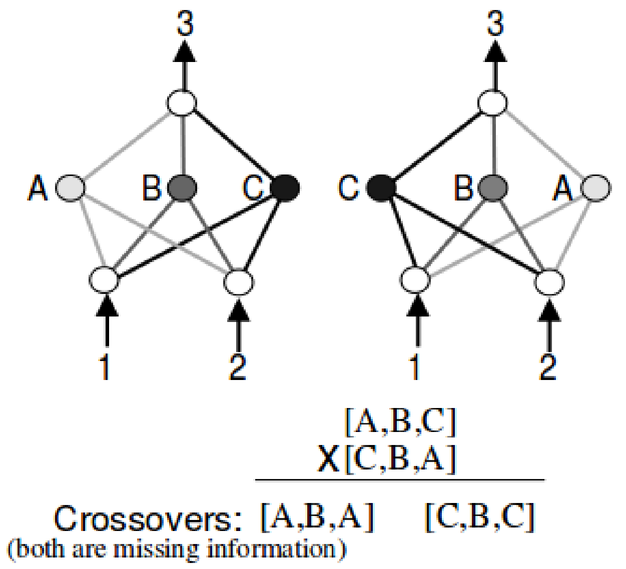


Figure 2.4: An example consequence of the competing conventions problem. Here, both parents share corresponding sub-networks (labeled A, B and C), but at different locuses within the network. Crossover produces children with duplicate and/or missing sub-network structures. (Stanley and Miikkulainen 2002b)

2. Allowing the optimisation of newly-created networks to prevent premature elimination

Another problem that arises in TWEANNs is that adding structures to a network can initially reduce fitness.(Stanley and Miikkulainen 2002a) This is a problem that affects NEAT, in particular, because of the way the weights of newly-added nodes are initialised. This poses a problem as networks may be culled by the selection process before they have a chance to optimise. NEAT protects these emergent structures by using a niching(Mahfoud 1995) method called *speciation*. Under this scheme, genetically similar individuals share fitness and compete with each other to reproduce before competing with the general population. Fitness sharing also ensures that the population does not converge prematurely to a local maxima on the fitness terrain by preserving diversity in the population (Mouret and Doncieux 2012).

3. Minimising topologies to prevent unnecessary bloat

The third problem encountered in TWEANN is that of extraneous genes, ie. networks may contain genetic components that are unnecessary. Traditional TWEANNs use random initialisation (Stanley and Miikkulainen 2002a) to introduce diversity into the initial population, but Stanley et al. argue that there is no way to know whether these randomly-generated topological structures are necessary. More nodes mean greater dimensionality of the ANN weight search-space. Furthermore, random initialisation introduces potentially detrimental structures that contribute to network bloat and complexifies the evolutionary task (Stanley and Miikkulainen 2002b). Thus, maintaining unnecessary connections may be needlessly costly. Instead of relying on a penalty in the fitness function that penalizes larger networks, NEAT minimises the need to maintain unnecessary topologies by starting with a uniform population of feed-forward networks with no hidden nodes and growing new connections only as necessary. This is effective as the speciation mechanism protects emergent innovation, so that new structures can be added and tested so that only fit additions to the topologies survive.

Using an ablation study, Stanley Stanley and Miikkulainen (2002b) demonstrated that all three components of NEAT were needed for the success of the algorithm in the double pole-balancing task with velocities. It compared the failure rates for versions of the algorithm with its components removed. As shown in Figure **??**, the failure rate for NEAT increased significantly for each component removed, demonstrating the importance of all three parts of the algorithm.

(Stanley and Miikkulainen 2002b)

Evolutionary Acquisition of Neural Topologies (EANT)

EANT is a hybrid learning technique to TWEANN - as well as searching for topologies, it also optimises existing topologies using Covariant Matrix Adaption Evolutionary Strategies(CMA-ES) (Hansen and Ostermeier 2001), a numerical optimisation method that does not require derivative information of the fitness landscape. Like NEAT, it starts with networks of minimal structure and builds them with mutation and crossover operators. EANT represent genotypes as a list of genes. Each gene can encode a neuron, a neural network input, or a connection between two nodes. The compact, linear genotype encoding method allows for fitness evaluation without needing to decode the genotype into its corresponding neural network.

EANT exploits existing topologies by keeping topologies fixed and optimising the weights of the networks with CMA until it detects convergence of fitness. Once convergence is reached, EANT initiates a structural mutation in order to explore the solution space. EANT is shown to outperform NEAT and several non-TWEANN algorithms in the Double Pole-Balancing Task (Kassahun et al. 2009).

Co-Variant Matrix Adapting Topology and Weight-Evolving Artificial Neural Networks (CMA-TWEANN)

CMA-TWEANN was devised as another hybrid learning alternative to some of the existing TWEANN methods. Moriguchi et al. **REF** criticises NEAT and EANT for the lapse in network fitness which occurs after mutation. Since each newly added node or connection is immediately initialised with non-zero weights, each new connection changes the input-output mapping of the network and each new node changes the behaviour of the entire network(Moriguchi and Honiden 2012). NEAT attempts to protect these vulnerable networks using speciation. CMA-TWEANN, on the other hand, zero-initialises new connections so that the input-output mappings and behaviour of the network remain unchanged after topological mutation. Like EANT, it also uses CMA-ES to evolve the neural network weights.

2.2 Evolutionary Robotics (ER)

Evolutionary Robotics is the discipline of automating the design of robots using Evolutionary Algorithms. Robotic design involves consideration for both the controller and the hardware (ie. morphology) of the robots. There are several approaches to evolving such robots. Evolutionary Algorithms may either be applied to the controller only, with fixed morphology, or the morphology may be co-evolved with the controllers.

2.2.1 Controller Design

One of the objectives of ER is building controllers for autonomous agents that are able to perform tasks in noisy, unstructured environments, with incomplete sensory information. The traditional methods of manually designing and evaluating robots and their controllers presents many difficulties for controller design. Mataric Matarić (1996) argues against the appropriateness of Reinforcement Learning to train robotic behaviour, on the grounds that the state space of a robot consists of continuous and discrete values from its sensor and effector values, which can be intractably large. Nolfi Nolfi, Floreano, et al. (1994) theorises that there are two main complications of manual controller design:

1. It is difficult to coordinate the interactions between the different modules of the robot's hardware, because the search space is a function of the number of possible interactions that arise between the modules (Cliff, Husbands, and Harvey 1993). Similarly, it is also difficult to coordinate the interactions between the different modules of the robot's controller architecture. To complicate matters, it is also diffcult to predict the interaction between the hardware and the controller.
2. The environment's feedback to the robot depends on the robot's preceding actions. Each action from the robot results in an evaluation of its effectiveness, as well as input data to determine the next action. Designing the robot to behave in a way such that the input stimuli it receives has enough spread and quality as training data is extremely challenging, not least because the state space of all sequences of actions is not only intractably large, but also difficult to predict.

Choosing a controller representation is related to the EA under which the controller will be evolved as well as the problem representation on which the EA will be applied. Current ER research employs techniques from many related disciplines in order to develop desired behaviour in robots (Sofge et al. 2007). A popular approach is to use EA for structural improvements on the behavioural representation of the individual, and Machine

Learning techniques such as Reinforcement Learning (Sutton and Barto 1998a) methods or backpropagation (Rumelhart et al. 1995) to fine-tune the behaviours during the generation (Miikkulainen et al. 2012; Parker and Bryant 2009).

The genotypic representation of the controller may either be a direct representation of the phenotypic construct used to map inputs to outputs, or it could be an indirect representation, a set of instructions specifying how to construct the phenotype (for example, Cellular Encoding (F. Gruau 1993)).

Rule-Based Controllers

The Rule-Based controller paradigm aims to develop a declarative mapping between the environmental stimulus and the resulting motor action of the robot (Sánchez-Parra and Vite-Hernández 2002). Fuzzy Logic can be employed by the mapping to process the inputs into a set of quantifying probabilistic variables (Carse, Fogarty, and Munro 1996). Given that the mappings provided by a rule-based system are discrete, there can be significant difficulty in using it in conjunction with analogue input and continuous action. Controllers that fall under this approach include Dorigo and Schnepf's Learning Classifer System Dorigo and Schnepf (1993), Finite-State Machine representations (Polikarpova, Tochilin, and Shalyto 2010; König, Mostaghim, and Schmeck 2009), and Grefenstette's SAMUEL system Grefenstette (1987).

Programmatic Controllers

Some authors have proposed evolving high-level programs as controllers, such as Brooks Brooks (1992). In contrast to rule-based controllers, programmatic controllers are imperative rather than declarative. This means that the controller consists of a set of instructions that modify the internal and external state of the robot instead of directly mapping inputs to desired behaviour.

Cliff criticises Koza's approach, that of evolving the high-level semantic primitives found in Behavioural Language, as biased towards certain paths of evolution. The choice of the semantic syntax fixes the paths that the eovlution can take. The granularity of such a set of semantic primitives favour coarser-grained fitness landscapes. This means that perturbations in a genotype consisting of a high-level syntax results in large dsiplacements of the individual on the fitness landscape. The designer's prejudices on the range of behaviours could also be encoded into their design of the behavioural language's semantics, which undermines the advantages of automated algorithm design (Cliff, Husbands, and Harvey 1993).

Neural Network Controllers

One of the most popular methods of controller representation is the Artificial Neural Network. Cliff, Harvey and Husbands advocated representing controllers as Artificial Neural Networks on the grounds that the primitives (ie. genotypic building blocks) that are manipulated in the evolutionary process should be as low-level as possible. This allows for a finer-grained search across the fitness landscape and supports the strengths of automated design, namely that of avoiding undesirable human bias.

2.2.2 Morphology Design

Evolving Hardware (EHW)

An emergent field in ER is that of applying Evolutionary Algorithms to the design of robot architecture instead of configuring them by hand, a discipline known as EHW. Different views of the scope of EHW alternately refer to “applications of evolutionary techniques to circuit synthesis” or, as Hautop et al. argued, a more effective holistic approach that includes applying EAs to the robot’s morphology, such as body plan and sensors (Lund, Hallam, and Lee 1997). An experiment by Bongard Bongard (2009) showed that increasing the number of evolved morphological parameters in a quadrupedal robot slowed evolution for simple tasks, but sped evolution as task complexity increased. Bongard concludes that EHW may show promise in evolving robots for more complex tasks.

Sensors

Sensors allow a robotic agent to receive feedback from its environment. The sensor maps the partial environmental state observable by the agents into a reading that can then be used as inputs to the controller. There exists a wide range of sensor types, such as tactile, SONAR (ulatrasound), range, camera and inertial sensors (Siciliano and Khatib 2007). However, for this research we are chiefly interested in ranged sensors such as Infrared Proximity sensors and Light Sensors.

Most modern sensors employ some kind of signal conditioning, in which non-linear signal responses are corrected for and amplification (gain) is performed. A signal conditioner may have in input terminal that receives the output signal of the sensor, applies the gain, and returns the amplified output signal. (DePauli 1989) We justify that the sensory distortion may be achieved in real-life using such signal processing on the sensor’s output signal.

2.2.3 Emerging Disciplines

The co-evolution of controllers and morphologies is a concept arising from the observation that the physical structures and behaviour controllers of organisms in nature evolve in lockstep with each other (Mautner and Belew 1999). The effectiveness of this approach in nature is used to justify applying it to robotic design. This is a way of overcoming both of the problems of manual robotics outlined by Nolfi Nolfi, Floreano, et al. (1994) in Section 2.2.1: the inscrutability of interaction between the morphology and the controller, and the intra-layer coordination of the modules within the morphology and controller. In 2002, Bugaska and Schultz used GENESIS, an implementation of GENITOR, and SAMUEL (Grefenstette 1987) in tandem to evolve Micro Air Vehicles that could navigate an obstacle course to reach a target location. GENESIS was used to evolve the controller, while SAMUEL was used to evolve sensor characteristics such as rance, field of view and placement. The experiment resulted in only preliminary data on task performance, but was referred to by the authors as "promising" for the future of the discipline (Bugajska and Schultz 2002).

Another emerging concept is that of Embodied Evolution (EE). Embodied Evolution is the concept of having all the evolutionary processes of the population take place between real robots within the task arena in an autonomous, aynchronous manner much like the reproductive processes occurring in biological populations (Ficici, Watson, and Pollack 1999). This contrasts with the more common practice of *embodied trials*, where the embodied robots are used to test fitness only and the evolutionary processes take place offline in a synchronous, centralised manner. The Embodied Evolution paradigm was used by Watson et al. to evolve light-seeking behaviour in a small population of robots. The results showed that EE performed significantly better than hand-design to produce robots that successfully sought out a light target (Watson, Ficici, and Pollack 2002).

2.3 Multi-Agent Systems

Panait et al. defines an agent, in the field of Artifical Intelligence, as a computational unit with a high degree of autonomy, and whom receives stimulus from and interacts with the problem environment (Panait and Luke 2006). A Multi-Agent System (MAS) consists of many such agents in the same environment, able to interact with each other, and constrained so that no agent may have global knowledge of the environment at any given point in the simulation.

2.3.1 Communication

Panait Panait and Luke (2006) defined communication in MAS as:

> "[A]ltering the state of the environment such that other agents can perceive the modification and decode information from it."

This definition encompasses explicit and implicit message-passing, as well as the rudimentary signalling in the absense of explicit communicative protocols, as in stigmergic coordination (O. Holland and Melhuish 1999). Communication between agents is an important factor when jointly-solving tasks as agents can share information about the environment, share partial solutions, or organise more effectively with one another. It was argued that if the task can be solved without communication between agents, then the problem can be decomposed into a series of singular tasks solvable by single agents (Panait and Luke 2006).

2.3.2 Cooperative Multi-Agent Systems

Our research was chiefly concerned with Cooperative Multi-Agent Systems (CMAS). CMAS are a subset of Multi-Agent Systems that rely on multiple agents to solve a single problem. The precise conceptual definition that separates Cooperative Multi-Agent Systems from other forms of distributed problem solving is disputed in the existing body of literature on the subject (Doran et al. 1997).

Doran et al. Doran et al. (1997) compared three different approaches to define Cooperative Multi-Agent Systems. The first proposal was a hierarchical taxonomy of cooperation in the context of Multi-Agent Systems made by Franklin Franklin and Graesser (1997), shown in 2.3.2. According to the taxonomy, agents within a CMAS are classified as *independent* if the agents act according to their own agendas without considering those of the other agents. If independent agents each work on a modular sub-problem of the task without interaction from other agents, the system is called *discrete* and is not a CMAS. However, it is possible for cooperative interaction to arise amongst independent agents without explicit intention of cooperation encoded in the agent controllers, for instance, when the behaviour of agents pursuing their individual goals coalesce to fulfil the same broad task. Coordinated behaviour under these conditions is classified as *emergent cooperation.*

If agents are not independent in Franklin's taxonomy, then they must necessarily be *co-operative.* Cooperative systems involve explicit intention of cooperation. There are two possible ways to achieve cooperative intent; either *communicative* or *non-communicative.* Communicative cooperation is the coordination of agents by explicit message-passing; agents may either execute a pre-meditated set of actions together as in *deliberative* cooperation, or negotiate with each other in order to maximise group and individual gain in *negotiative* cooperation. Non-communicative cooperation, on the other hand, involves self-organising from the cues emitted from the behaviour of other agents in the same task; one of the canonical forms of non-communicative cooperation is stigmergic coordination.

Another possible approach outlined by Doran et al. is Doran's description of cooperation as solely a property of the actions and goals of the agents. According to Doran, intention is irrelevant. The agents must simply act to achieve a common goal, or to achieve their own goals as well as the goals of other agents in the task. Doran focuses not on the "why" but the "how" of cooperative behaviours, and classifies them by their underlying intra-agent processes:

- Reaction/reflex. Cooperation under this mechanism involves neither reflection on choices of action nor intention of cooperative action (Steels 1990). This corresponds to Franklin's classification of emergent cooperation.

- Deliberation/reflection. Agents that are able to choose different courses of action make joint decisions to execute a coordinated plan, or negotiate with each other until the goals of all agents involved in the negotiation converge (Durfee and Lesser 1991).
- Conceptualisation. Agents maintain a memetic, transmissible concept of cooperation and how it can be achieved amongst agents. The agent uses this in order to cooperate without the agent needing to consider possible sets of actions (Doran et al. 1997).

Doran also describes the types of cooperative behaviour in terms of the level of altruism involved in the agents' decision-making processes, and classifies them under two main types:

- Selfish cooperation, in which agents cooperate only as a means to a selfish end.
- Partly or wholly altruistic cooperation, in which agents select cooperative courses of actions that will further the interests of a group, regardless of its own goals.

The last definition, offered by Norman, is also the most restrictive definition. Norman argues that for agents to be called cooperative, they must act with a common purpose and for common benefit. The purpose of an agent may be encoded in the set of behaviours the agent is able to take, or in the task definition in which the agent is a participant. Such requirements are necessary, but does not necessarily mean that cooperative behaviour will arise from systems fulfilling such requirements. The agents must also intend to act together in order to achieve the shared purpose. Norman argues that the requirement of explicit intention to cooperate in the agents implies the need for some kind of internal state (Castelfranchi 2000). This definition precludes modes of joint behaviour mentioned by Franklin and Doran (*emergent cooperation*) from being called cooperation.

From the above survey, it is clear that there is disagreement within the MAS community regarding philosophical underpinnings of cooperative behaviour. However, there are common concepts that are shared amongst the approaches discussed above, such as coordination in order to accomplish a shared goal. Also noteworthy is that Franklin and Doran both describe processes in which cooperation is negotiated to maximise both group and individual gain. This seems to be one way in which the concept of competition can be reconciled with individually-acting agents who share a common goal. Furthermore, Doran's panel provides a description of the type of behaviour that may arise when agents inadvertently succeed at a task that requires multiple agents to accomplish, even when from the agent's viewpoint it is simply fulfilling its own goals. The discussion coins it *emergent cooperation*, and it is of particular interest in the context of this research as the agents in this foraging task possessed neither state nor communicative protocols (See Section 3).

Multi-Agent Learning and Team Composition

Panait et al. Panait and Luke (2006) define Multi-Agent Learning as

> ...[T]he application of machine learning to problems involving multiple agents

The learning approach of the system defines the level at which the machine learning process is applied. In this context, the concept of a learner is defined as a unit tasked with discovering the solution model for the problem. Panait believed there to be two major categories of multi-agent learning approaches: team learning and concurrent learning.

In team learning, one learner aims to find a set of behaviours for all the agents in a team. Team learning implies mandatory cooperation, as the agents are constituents for achieving the goal of the learner. It is not possible for the agents in a single learner to compete within itself. One of the potential advantages of this method is that it removes the need to consider the game-theoretic aspects governing agent behaviour on the team. However, complex behaviour can still arise from the interactions between agents in what Panait dubs *emergent complexity*.

There are three main approaches to team composition in team learning: homogeneous, heterogeneous and hybrid learning.

Homogeneous team learning attempts to find a single behaviour, adopted by all agents, that will increase the fitness of the team. One of the potential advantages of this method is simplifying the task of credit assignment. Since the algorithm evaluates performance on the team level rather than the agent level, there is less of a need to develop credit-assignment solutions for each individual agent. Another advantage is a smaller search space compared to heterogeneous teams, as it lacks the additional dimensionality of deciding which agents in combination form a good team. Genetically homogeneous agents can still behave differently within a task if the behavioural path taken by the agent is dependent on its initial conditions, or it includes sub-behaviours described in terms of the agent's relationships with other agents.

In heterogeneous team learning, each agent learns its own separate behaviour that will increase the fitness of the team. Heterogeneous team learning composes teams with genetically distinct individuals. Under CMAS setups that amount to a multi-objective optimisation task, homogeneous learning may result in convergence to a Pareto Optimal (**Deb:1999:MGA:1326811.1326812**) solution adopted by all agents. In such setups, heterogeneous teams would be more appropriate, as the ability to speciate means that each evolutionary individual can potentially maximise on one objective of the task.

In 2009 Waibel Waibel, Keller, and Floreano (2009) et al. investigated the effect of genetic team composition and level of selection on performance of co-operative behaviour in a multi-robot foraging task. The controllers used were ANNs and no direct communication between robots was implemented. The study did not observe emergent behaviour; rather it measured performance under tasks that required different levels of co-operation. Waibel et al. identified three different tiers of cooperation:

- Individual Foraging: Resources to be transported could be moved by one agent alone.
- Cooperative Foraging: Resources required cooperation to be moved and fitness was divided evenly amongst cooperating agents.
- Altruistic Cooperative Foraging: Included both small and large resources; robots co-

operated to push a large resource at the expense of pushing a small one.

The study used several learners for each forage task, each composed of either behaviourally homogeneous or heterogeneous teams. The results showed that for both task 2 and 3, homogeneous teams evolved with individual-level selection and team-level selection performed significantly better than heterogeneous teams of either selection method. Among others, Waibel provides the possible explanation that heterogeneous teams had to additionally optimise the co-ordination of agents with different behaviours for cooperative foraging (Waibel, Keller, and Floreano 2009).

Hybrid team learning divides the set of agents into individual squads, with homogenous team learning taking place within each squad, and each squad being an agent of a heterogeneous team learning approach for the whole population.

Concurrent learning, another popular approach, contrasts with team learning in that it involves each learner in the system learning its own set of behaviours that will improve a part of the team. A single learner may consist of single or multiple agents focusing on the same area of optimisation. In this case, evaluating the learner relies on a way to quantify the learner's contribution to the system.

Credit Assignment

When multiple learners are involved in solving parts of a task, as in concurrent learning, the proportion of fitness received by each learner must be decided. There are a variety of different approaches which have been studied.

The most simple credit assignment method is global credit assignment, in which overall fitness is divided equally amongst individuals regardless of individual performance. This, however, presents problems when we are trying to select for desirable behaviour on the individual level. For some tasks, we may wish to reward or punish agents based on their behaviour in the team. For other tasks, global reward is not always possible to compute efficiently. The alternative then, is to assign credit on a local level, where each agent is rewarded separately according to its actions. This, too is problematic when the task requires cooperation to be performed adequately, as an individual reward may not give the agent enough rational incentive to cooperate. There is strong evidence to suggest that the best credit assignment scheme differs according to the nature of the task.

Balch (Balch 1997) compared the learning rate and task performance of global versus local reward for a robotic soccer team. The team was trained using Q-learning (Watkins 1989), a RL algorithm. When the team scored a goal, the global reward scheme would reward one point to all team members regardless of who scored the goal. The local reward scheme rewarded one point to the robot closest to the ball from the scoring team, and subtracted one point from the robot closest to the ball from the non-scoring team. The study found that local reward led to faster convergence of agent behaviour, but not necessarily to good team performance. One possible explanation for this is that soccer requires specialised roles, not all of which would be rewarded under the reward scheme. Local rewards tend to optimise for greedy behaviour, so the team ends up performing poorly.

Another study by Balch also used Q-learning to train teams of agents, but in a multi-agent foraging task (Balch 1999). In the study, Balch set out to investigate the relationship between reward structure and the behavioural diversity of the convergent teams. One was a global reward, in which all agents were awarded for a retrieved object; the next was a local reward, in which the agent who retrieved the object was rewarded. The final one was a heuristic *shaped local reward*, in which an agent was rewarded locally for accomplishing portions of the task. The task was set up such that the forage object could be manoeuvred by a single robot. Balch found that local and shaped local reward resulted in better task performance and faster convergence than global reward, in contradiction to the findings for robot soccer. Furthermore, with reference to Section 2.3.2, the fittest teams converged to a high degree of behavioural homogeniety, which corresponds orthogonally to Waibel's (Waibel, Keller, and Floreano 2009) results of better team performance with homogeneous agents.

Rajagopalan Rajagopalan et al. (2011) investigated the impact of credit assignment on the performance in a predator-prey task. A team of hyenas were evolved with the Multi-Component Enforced Sub-Populations (Rawal et al. 2010) method to catch zebras and gazelles in a simulated hunting task. Zebras were faster and required the hyenas to cooperate and surround them for a kill, but yielded a higher reward. Gazelles were slower and could be caught by a single hyena but yielded a lower reward. The results of their experiments for the different reward setups and levels of communication are shown in Figure 2.3.2. Their findings showed that cooperation (and, in turn, higher overall fitness) took longer to emerge when local reward was used than when reward was shared. Additionally, their findings suggested that predators would choose whether or not to hunt alone or cooperatively based on whichever method results for a higher net return for the individual. This suggests that reward structure can be used to bias evolution towards cooperative behaviour, at least according to the proposals of Franklin and Doran in the previous section.

2.3.3 Foraging Task

There are a great number of problem domains within the field of Cooperative Multi-Agent Systems such as air-traffic control (Steeb et al. 1988), meeting scheduling (Crawford and Veloso 2005), and game-theoretic social dilemmas (Knack 1998). We focus on collective foraging as the task for our investigation. Foraging is one of the classical problem tasks for cooperative robotics (Ostergaard, Sukhatme, and Matari 2001). Foraging tasks may be implemented under different metaphors such as box-pushing, as well as variations on the task such as collective construction (Nitschke 2011). However, Mataric and Goldberg (Goldberg and Mataric 2001) formally define the task as a two-step repetitive process in which:

1. $N(N >= 1)$ robots search designated regions of space for certain objects
2. Once found, these objects are brought to a goal region using some form of navigation.

The complexity of the foraging task depends on the parameters of the task environment and composition of the agent population. Different forms of the foraging task may adjust the number of agents involved in the task, the number of source and target destinations for the objects, the space constraints of the task environment, the number of object types to be collected, the placement of objects in the area, genetic homogeneity or heterogeneity of the agents and communication style.

2.4 Conclusion

The study of the relationship between the robot's controller, its actions and the perception of its environment is a key area in optimising the performance of robots. One of the areas of interest proposed by Nolfi and Floreano Nolfi and Floreano (2002) is that of exploiting agent-environment interaction in order to improve the learning process for the robot. This is based on the observation mentioned in Section that the environmental feedback received by an agent at timestep N is dependent on its actions at timestep $N - 1$. An agent evolved with well-coordinated sensori-motor processes can perform actions that help it receive good-quality feedback from the environment in order to adapt its behaviour. Coordination between the controller and the actions can also help the agent compensate for limitations in its morphology that may prevent it from receiving useful training data. For the purposes of this research, we argued that manually altering sensory patterns received by the agent can remove some of the reliance on exploiting agent-environment interactions in order to receive useful input from the environment.

The emergence of the type of cooperative behaviour in MAS is said to be dependent on a variety of factors such as task setup, communication method and credit assignment. Bjørkøy argues that competitive behaviour in artificial systems is explicitly programmed into evolutionary algorithms through selection mechanism design and credit assignment (Bjørkøy 2011). In support of this, Waibel's study of team composition and level of selection in a foraging task requiring cooperation showed that genetically homogeneous teams evolved with either individual or team selection performed significantly better than heterogeneous teams of either selection strategy on tasks requiring cooperation (Waibel, Keller, and Floreano 2009).

The selection mechanism has parallels with the reward structure, as reward forms the basis for evaluating and selecting individuals for breeding. Balch's study of reward structure in multi-agent foraging shows that local reward for concurrent learners results in better team performance than global reward (Balch 1999). However, the agents in Balch's experiments were trained using RL instead of an EA. Furthermore, the forage objects in the task did not require multiple agents to carry, and so lacked a cooperative element.

Waibel's study of team foraging showed that there is a signficiant fitness dropoff for heterogeneous teams compared to homogeneous teams as the foraging task required more and more cooperation in order to accomplish (Waibel, Keller, and Floreano 2009). This is relevant to this research, as it is important to discover whether or not the hypothesised effects of altering sensor processing is effective across foraging tasks of all levels of cooperation.

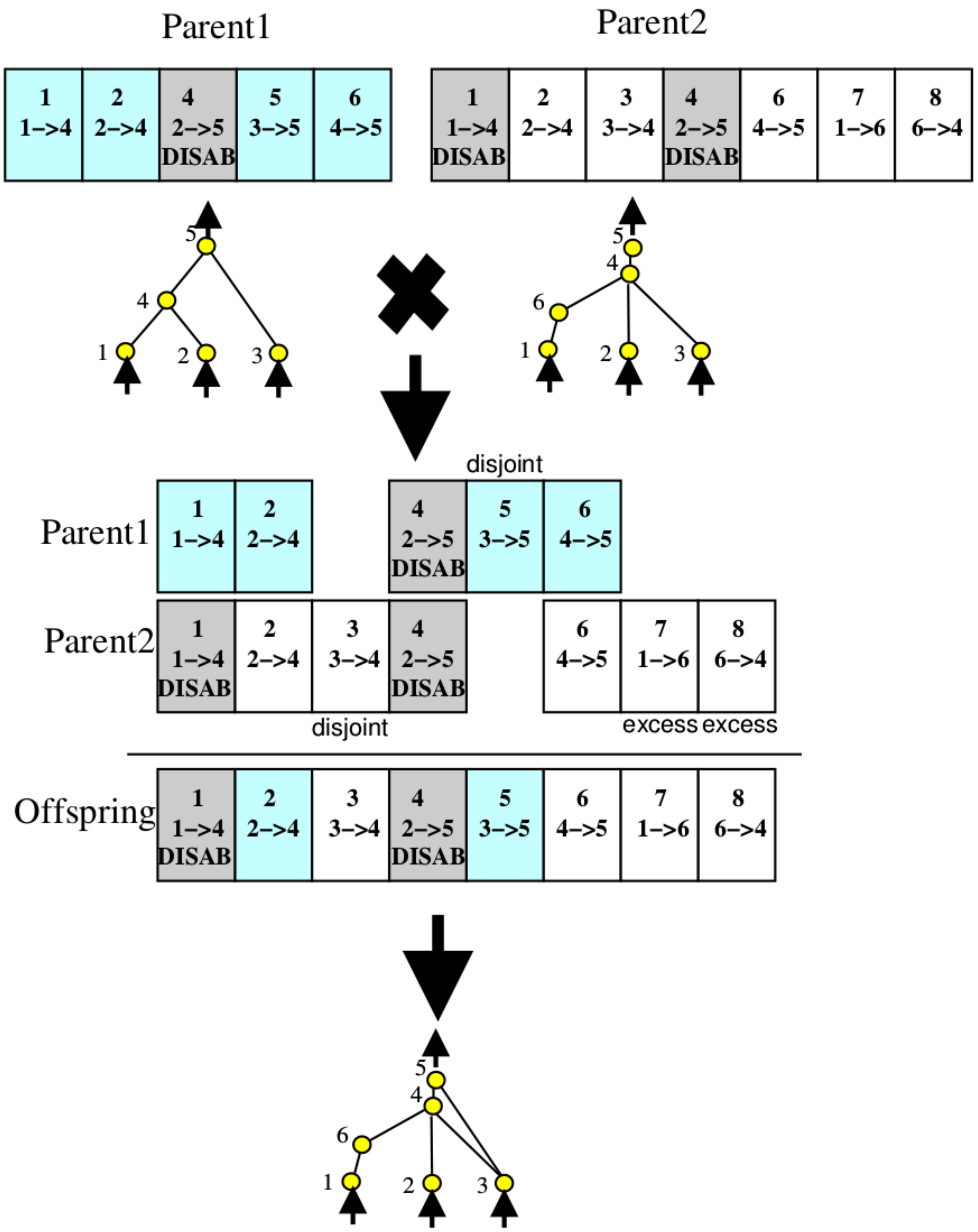
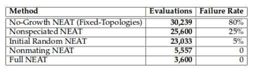


Figure 2.5: Example crossover of two NEAT genotypes. Each genotype posseses a set of genes, each labeled with an innovation number. The offspring inherits genes 4 and 1, which appear in both genotypes, and the disjoint genes from the fitter parent (left).(Stanley and Miikkulainen 2002b)



Method	Evaluations	Failure Rate
No-Growth NEAT (Fixed-Topologies)	**30,239**	80%
Nonspeciated NEAT	**25,600**	25%
Initial Random NEAT	**23,033**	5%
Nonmating NEAT	**5,557**	0
Full NEAT	**3,600**	0

Figure 2.6: NEAT ablation study, showing the average number of evaluations for a solution in the Double Pole-Balancing with Velocities task. Failure rate is used to gauge the importance of the algorithm in the absence of each of its components.

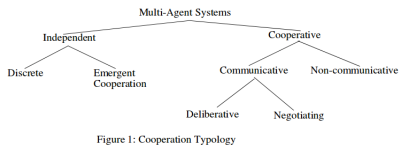


Figure 2.7: A proposed taxonomy of Cooperation in Multi-Agent Systems. (Franklin and Graesser 1997)

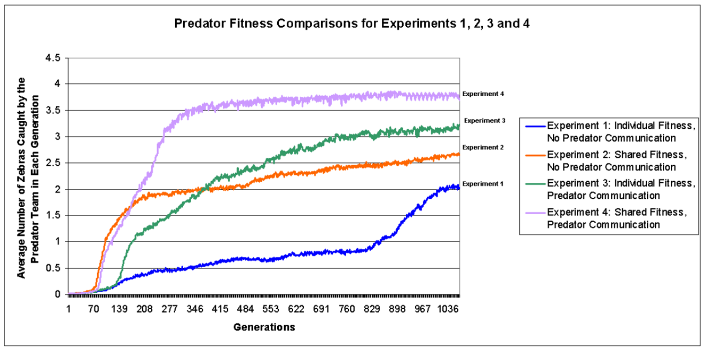


Figure 2.8: Number of prey caught for four different task setups, averaged over 6000 trials per generation. In both commmunicative and non-communicative tasks, the hyenas caught significantly more prey on average when reward was shared amongst the team. (Rajagopalan et al. 2011)

Chapter 3

Methods

3.1 Research Questions

The main research questions for this project, as stated in the Introductory section, are:

1. Can artifically distorting the sensory input of agents increase performance on a foraging task that requires cooperation?
2. Are the beneficial effects of 1 affected by the team composition and reward structure?
3. Are the beneficial effects of 1 consistent over tasks requiring increasing levels of cooperation?

3.1.1 Can artifically distorting the sensory input of agents increase performance on a foraging task that requires cooperation?

To test this hypothesis, we chose a single, fixed controller evolution algorithm. An ANN was selected to represent the controller for the robots. ANNs have been widely used in ER because of several desirable properties. Firstly, their output response is tolerant of noise in the environment, as the activation of any node is subject to the sum of several inputs. Secondly, Neural Networks are able to receive and output both digital and analogue signals (Nolfi and Floreano 2000). Thirdly, the evolutionary primitives in Neural Networks (such as synaptic weights and node configuration) are low-level enough to allow for precise fine-tuning of fitness without being biased by the coarseness of high-level programmatic semantics such as in Koza's Genetic Programming (Cliff, Husbands, and Harvey 1993). Each robot sensor feeds into an input node of the ANN and each output node in the ANN feeds into the wheel drive actuator (See Figure 4.7).

A TWEANN was chosen to evolve the Neural Network controllers because it is difficult to determine beforehand the number of nodes that is needed for a complex task such as collective box-pushing. Fixing the number of nodes at a large number increases the search space for optimising the connection weights between them. Furthermore, in unstructured tasks for which the relationship between the number of nodes and the performance ceiling is unknown, one has no basis for choosing the number of nodes so that the search space encompasses such a ceiling(Stanley and Miikkulainen 2004).

NEAT was chosen as the controller evolution method, since NEAT is a well-established method for evolving robotic controllers in multi-agent tasks (Gomes, Urbano, and Christensen 2013; Martín, Lope, and Santos 2007). Additionally, Luke et al. Luke and Spector (1996) found that in multi-agent tasks requiring cooperation, heterogeneous teams using a basic form of restricted breeding consistently outperformed heterogeneous teams whose agents were allowed to breed freely. The explanation was offered that homogeneous teams with free breeding suffered due to the competing conventions problem described in Section 2.1.2, resulting in crippled offspring. Although the experiments in the aforementioned paper were conducted using Genetic Programming to evolve agent controllers, the competing conventions problem and its applicability to crossovers in TWEANNs mean that an evolutionary algorithm which includes speciation, such as NEAT, would be particularly suitable for the heterogeneous teams tested in this research.

Algorithm 2: Evolutionary Loop for NEAT as used in this project.

```
Initialise population with minimal topologies for an initial connection density ;
for do Number of Epochs
    Evaluate Population ;
    Speciate population ;
    Calculate average adjusted fitness of each species ;
    Adjust species sizes based on comparison with global fitness ;
    Select fittest $r\%$ of each species as elite pool;
    Add elite genotypes to next generation's species ;
    while Size of next generation's species ¡ $n$ do
        Randomly choose individuals from elite pool to mutate and recombine ;
    end
end
```

NEAT is an algorithm consisting of three main components, in addition to the evolutionary loop structure shown in Figure 2:

Genotypic Representation of ANNs for Meaningful Recombination

In order to solve the competing conventions problem for recombining networks as described in Chapter 2, NEAT marks each structural mutation with a globally-incremented *innovation number*. NEAT uses direct encoding and represents the individual's genotype as two chromosmes, one listing node information and one listing connection information. The node chromosome enumerate each node and describe the node type. The connection chromosome consist of a list of 5-tuples, each listing in-node, out-node, connection weight, enabled or disabled, and innovation number. The network described by the genotype represents the

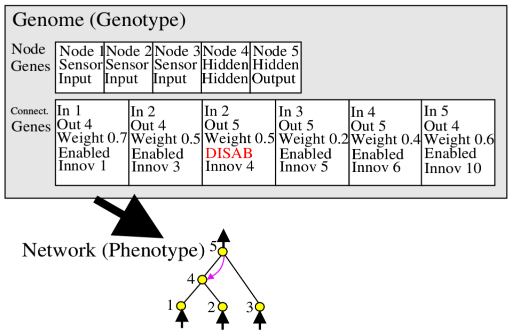

Genome (Genotype)
Node Genes
Node 1 Sensor Input
Node 2 Sensor Input
Node 3 Sensor Input
Node 4 Hidden Hidden
Node 5 Hidden Output
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight 0.5 Enabled Innov 3
In 2 Out 5 Weight 0.5 DISAB Innov 4
In 3 Out 5 Weight 0.2 Enabled Innov 5
In 4 Out 5 Weight 0.4 Enabled Innov 6
In 5 Out 4 Weight 0.6 Enabled Innov 10
Network (Phenotype)
5
4
1
2
3

Figure 3.1: Genotypic Representation in NEAT.

ANN's behaviour.

Protecting Innovation Through Speciation

NEAT creates new nodes by splitting a connection between two pre-existing nodes and placing the new node in between them. The weight into the new node remains the same while the weight out of the node is initialised randomly. Because this changes the input-output mapping of the genotype and may cause temporary lapses in fitness while the new network weights are unoptimised, NEAT protects innovation by dividing the population into m *species* of networks with similar topologies. The individuals then compete with other individuals in the same niche, giving them time to optimise before competing with networks of dissimilar topologies. NEAT does this by introducing some additional identifiers for gene types. When comparing two individuals, each unmatched gene from one individual's chromosome is classified as either disjoint from (D), or extraneous to (E) to the other individual's chromosome. The following formula is used to measure δ, the structural distance of two genotypes given in equation 2.3:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W} \tag{3.1}$$

where c_1, c_2 and c_3 are constant coefficients that weight the importance of each term, and N, the number of genes in the larger chromosome, is used to normalise the unmatched gene count against the chromosome size.

The calculated value δ is used to speciate the next generation of candidate genotypes according to a similarity threshold, δ_t. A candidate genotype is tested against a random individual from a given species in the previous generation and is placed into the first species for which the candidate achieves a structural distance of less than δ_t. As part of the reproductive step, NEAT also uses explicit fitness sharing amongst individuals of the same species. This is done to prevent any one individual from dominating the species. The size of species S_j, N_j, is then adjusted based whether its average adjusted fitness is higher or lower than the global average. The new size of the next generation of species S_j, N_j', is calculated by Equation

3.2:

$$N'_j = \frac{\sum_{i=1}^{N_j} f_{i,j}}{\overline{f}} \tag{3.2}$$

(Stanley and Miikkulainen 2002b)

where $f_{i,j}$ is the adjusted fitness of individual i in S_j and $\overline{f}$ is the adjusted mean global fitness. The adjustment of the species sizes are given by Equation 2.5:

$$\sum_{j=1}^{m} N_j = \text{PopulationSize} \tag{3.3}$$

The fittest $r\%$ of S_j is then chosen to breed N'_j children to replace the previous individuals in the species.

Minimizing Topologies

The NEAT algorithm initialises the population as networks with no hidden nodes, ie. the input and bias nodes are connected directly to the output nodes. The number of connections between the input and output nodes is determined by an initial connection density parameter $D_{initial}$, with a value of 0 indicating no connections and a value of 1 indicating a fully connected graph.

We aimed to test the potential of sensory distortion to improve the evolution of cooperative behaviour by comparing the task performance of morphologies with altered linear gain parameters with a morphology in which the signal processed by the sensors remain unaltered.

Sensor Transducer Gain

The distortion of the signal processed by the sensors was achieved with a linear parameter. Although the linear parameters could theoretically be any real-valued number, the testing of every possible linear parameter would result in an infinite search space. To decrease the search space, the variations on the linear parameters were limited by a range and a resolution. The resolution was a measure of the granularity of change in the linear parameter value. Furthermore, we reasoned that for the purposes of this experiment, the linear parameters to be tested could be quantised as either augmenting the signal, diminishing the signal, or having no effect on the signal. The range and resolution were selected to reflect this (See Section 4.5).

Both the infrared proximity sensors and the typed sensors used in the experiments were toy sensors, in that their output signal were a simplified approximation of a real-life sensor response. The output response for an infrared proximity sensor S with a field of view of input fov was described by Equation 3.4.

$$S(\text{fov}) = 1 - \arg\min(\frac{Dist_{(\text{robot,closestobject})}}{Range}, 1.0) \tag{3.4}$$

The output response for a typed proximity sensor was simply the output response of the proximity sensor multiplied by the linear sensor gain assigned to that sensor as in Equation 3.5.

$$S_{type}(\text{fov}) = 1 - \arg\min(\frac{Dist_{(\text{robot,closestobject})}}{Range}, 1.0) * gain_{type} \tag{3.5}$$

3.1.2 Are the beneficial effects of Reseach Question 1 affected by the team composition and reward structure?

It was not known whether or not team composition and reward structure affects the improvements that sensory distortion is hypothesized to impart upon the evolution of cooperation in robots. In order to study this, we tested two different types of team composition, each with its own corresponding reward structure.

Team Composition, Selection Method and Reward Structure

Algorithm 3: Population Evaluation Procedure for the Homogeneous Setup.

```
for Each genotype g in Population do
    Clone $g$ $K$ times to make up a team ;
    Run team through forage task $R$ times;
    Obtain team score as genotype score ;
end
```

The homogeneous team setup used team learning. Each genotype from the candidate pool at each generation was cloned K times, where K was the number of robots in the team. Fitness was assigned to the genotype of which the team was comprised. Each box had a value corresponding to the number of robots needed to move it (See Section ASDF). The same team was then tested R times on the forage task, and the performance averaged over R runs. The fitness for the homogeneous setup was calculated according Formula 3.6:

$$Fitness_{genotype} = \frac{\sum_{i=1}^{R} \frac{\text{TeamFitness}_i}{Totalnumberofresources} * 100}{R} \tag{3.6}$$

Each genotype corresponded with all members of a team; therefore the evolution used team selection.

Algorithm 4: Population Evaluation Procedure for the Heterogeneous Setup.

```
for Number of teamsets $H$ do
    Randomly divide Population into teams of $K$ random genotypes ;
    Run team through forage task $R$ times;
end
Average fitness of each genotype over $R$ team evaluations ;
```

The heterogeneous team setup was devised in order to study the behaviour of agents that were given a choice to cooperate or behave selfishly. Theoretically, the most rigorous test cooperation in heterogeneous teams would be to use concurrent learning. All combinations of teams for a given candidate pool would be generated and the individual performance of each individual in every team it is assigned, as well as the overall performance of every possible team, would be tracked. Because of the large search space in testing all possible combinations of teams for a given candidate population (for example, the number for all possible team combinations for a population of size P and a team size of K would be $\binom{P}{K} = \frac{P!}{K!(P-K)!}$), a heuristic compromise was used instead. Because of this, the possibility of speciating according to specialised roles in the team left unexplored. The evaluation procedure for the heterogeneous setup is given in Figure 4. A population could be randomly divided into Q teams of K genotypes. Such a division is referred to as a teamset. The parameter H specified the number of teamsets to be generated from the candidate population at each generation. Each team in the teamset was then run R times on the forage task. In contrast to the homogeneous setup, each genotype in the heterogeneous setup was rewarded individually based on its participation in a box being pushed into the target area.

To encourage active cooperative behaviour, the values of the boxes at timestep 0 were set as V_{max} and then decreased in proportion to the number of timesteps that had elapsed when it was collected. Moving a box into the target area flagged it as having been collected. The robots that had participated in the move are credited with a portion of the box's time-discounted value. If n robots had pushed a box into the target area, the fitness awarded to each robot $A_i, i \in [1, n]$ that participated in collecting a box at timestep t is defined by Equation 3.7:

$$Reward(A_i) = \frac{V_{max} - 0.9 * V_{max} * \frac{t}{t_{max}}}{N} \tag{3.7}$$

Moving a box from the target area back into the forage area flagged it as uncollected, and was penalised by a negative value, with the effective penalty equal to that of the credit gained from the last time the box was pushed into the target area. Moving a previously-collected box from the target area back into the forage area resulted in the participants being credited with a recalculated, time-discounted reward for the box. The final score for each genotype at the end of simulation $i, i \in [1, R]$ is calculated by Equation 3.8:

$$Fitness_{genotype} = \frac{\sum_{i=1}^{R} \frac{\text{IndividualFitness}_i}{\text{TotalNumberofResources}} * 100}{R} \tag{3.8}$$

The selection criterion used by NEAT for the heterogeneous setup was the individual genotypic fitness value assigned by the evaluation process. However, for the purposes of this research, we still use team performance to gauge overall task success of a team according to Equation 3.6.

We planned to measure the difference in impact of the sensory distortion across team composition and reward structure by comparing difference in their performance across corresponding task setups. Because of logistical constraints, we did not, however, vary reward structure for each team setup.

3.1.3 Are the beneficial effects of Research Question 1 consistent over tasks requiring increasing levels of cooperation?

We would like to investigate whether or not the ability of sensory distortion hypothesised to improve team performance is retained across all levels of cooperation. Of particular interest is whether or not the best set of linear gain complements for the typed sensors are different for tasks at each level of cooperation.

To test this, we planned to compare the degrees of improvement hypothesised in Research Question 1 across the different task setups.

Chapter 4

Experiments

The codebase for this project can be roughly split into the following modules:

Shared Simulation Platform:

- Realistic physics and sensor simulation
- Accurate visualisation of simulation platform
- Heuristics

Shared Experiment Platform:

- Experiment configuration
- Morphology configuration
- Simulation configuration
- Entry points into simulation
- Support for differing task setups

Individual Experiment Platform:

- Implementation of Evolutionary Algorithms
- Implementation of Morphologies
- Statistics

4.1 Languages and Tools

Java 8 was used to develop the simulation and experiment platforms.

Requirements for Shared Simulation Platform:

- Written in a language in which all three team members were proficient
- High modularity; must allow use of custom evolutionary algorithms
- High configurability
- Sufficiently accurate physics simulation
- Visualisable
- Simulation artefacts must be serialisable

Additional Libraries for Shared Simulation Platform:

- JBox2D 2.3.1
- args4j 2.0.29

Additional Libraries for Shared Experiment Platform:

- JBox2D 2.3.1

Additional Libraries for Individual Experiment Platform:

- Encog 3.2.0
- Logback 1.1.2
- Apache Commons Math3 3.3.3
- Apache Commons Lang3 3.3.2

Originally the MASON Simulation Platform (Luke, Cioffi-Revilla, et al. 2005) was chosen for its promising visual interface, extensive documentation and well-established support community. However, the code structure of the MASON backend made it difficult to extend to a multi-agent simulation platform with the required portrayal structures and abstraction facilities. MASON was also inadequate in providing accurate simulation of physical interaction, so the JBox2D (Murphy 2014) library, a Java port of the popular Box2D project (Catto 2014), was adapted to provide this functionality. In the end, we used MASON mainly for its simulation and scheduling facilities, as well as its simulation portrayals for visualisation and debugging.

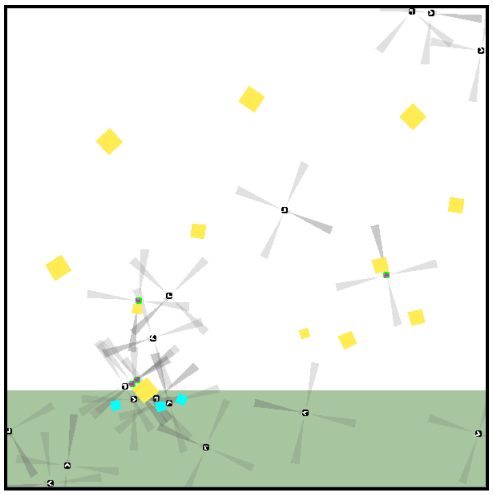


Figure 4.1: A top-down visualisation of the forage task, with small, medium and large boxes.

4.2 Simulated Environment

The experiments for this research were conducted in a 2D physics simulation. To preserve the accuracy of the simulation, the units used were scaled to correspond with metric units in real life. Figure **??** shows an example of the visualisation for the simulation.

Forage Task

The forage task required agents to move square resource objects to the designated target area. The task environment, which encompassed both the forage and the target area, was a 20m by 20m rectangular space enclosed by black walls. The task specified a single target area, which may be placed in any orientation of $NORTH$, $SOUTH$, $EAST$ and $WEST$ of the task environment. The positions and orientations of the resources were initalised randomly outside of the target area using a Mersenne Twister(Matsumoto and Nishimura 1998) random number generator implemented by Sean-Luke Panait(Luke, Cioffi-Revilla, et al. 2005), which is k-distributed(Matsumoto and Nishimura 1998). Robotic agents were randomly spawned with the Mersenne Twister outside of the target area at the beginning of the simulation.

The simulation ended either when all the boxes had been pushed into the target area, or when 10000 timesteps had elapsed. We specified three classes of forage objects: large, medium and small boxes. The box values and parameters are supplied in Table 4.2.

Box Size	Number of Robots Required to Push (N)	Maximum Box Value (V_{max})
Small	1	1
Medium	2	2
Large	3	3

Table 4.1: The maximum value assigned to each resource class. The values corresponded with the number of robots needed to push the box.

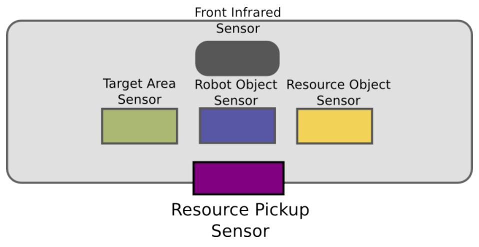


Figure 4.2: The schematics to the front of the robot.

4.3 Robot Morphology

The core design for the robots was meant to be as similar to the Khepera III (Lambercy and Tharin 2013) robot as possible. The robot itself was a circular body with a mass of 700g, equipped with sensors around its periphery. The robot used two actuators, one on the left and one on the right.

The robots were equipped with two main types of sensors: infrared proximity sensors, which were responsive to all objects in the environment, and typed proximity sensors, which were responsive to only one type of object in the environment. The mechanism of action of the typed sensors was rationalised as selective receptivity to a single colour in a task setup that painted each object class with a different hue. The exact placements and parameters for the sensors are shown in Figure 4.3 and **??**.

Type	**Bearing**	**Orientation**	**Range**	**Field of View**
Resource Pickup Sensor	0	0	N/A	N/A
Robot Agent Sensor	0	0	3	$\pi/9$
Resource Object Sensor	$-\pi/180$	0	3	$\pi/9$
Target Area Sensor	$\pi/180$	0	3	$\pi/9$
Infrared Sensor	0	0	3	$\pi/9$
Infrared Sensor	π	0	2	$\pi/18$
Infrared Sensor	π	0	2	$\pi/18$
Infrared Sensor	$\pi/2$	0	2	$\pi/18$

Table 4.2: Parameters for the robot sensor morphology used in this experiment. The bearing, orientation and field of view values are given in radians and the range is given in meters.

The sensors are specified according to their type, bearing, range, orientation and field of view. The bearing was a value in the range [0,2π] which specifies the angle offset from the front of the robot, with 0 radians/2pi radians indicating located directly in front. The orientation with a range [-π, π] specifies the angle from which the sensor is facing out from the robot's body, with 0 radians specifying a sensor facing perpedicularly outwards from the body. The field of view in range [0, π] specifies the angle width of the cone observable by the sensor. The range [0, ∞] specifies the maximum distance of objects detectable by the sensor. All three typed sensors were orientated to share overlapping fields of view, with the reasoning that using them in concert would be able to provide a camera-like view of the objects in front of the robot.

4.4 Evolutionary Parameters

	Max Epochs	Population Size	Runs Per Genotype	Runs Per Team
Homogeneous: No Cooperation	100	50	4	4
Homogeneous: Altruistic Cooperation	100	50	4	4
Homogeneous: Mandatory Cooperation	150	50	4	4
Heterogeneous: No Cooperation	100	80	4	4
Heterogeneous: Altruistic Cooperation	150	120	5	5
Heterogeneous: Mandatory Cooperation	200	120	5	5

Table 4.3: EA parameters for each task setup. Parameters had to be ajdusted with consideration to time constraints and performance.

4.5 Linear Transducer Gain Complements

For each run of the experiment, the robot morphology was fixed as described in Section 4.3. Because we only needed three cases for a linear gain, we fixed the range for the parameters as [0.5,1.5]. The generator assigned these values with a resolution of 0.5, ie. each typed sensor could receive one of three values: 0.5, 1.0, 1.5. Sensor gain complements for the three typed sensors were then generated. The total number of complements which needed to be tested for each task setup was thus $3^4 = 27$.

The control cases for the experiments are the complements with all three gain values at 1.

	Small Boxes	Medium Boxes	Large Boxes
No Cooperation	15	0	0
Cooperation	0	5	10
Altruistic Cooperation	5	5	5

Table 4.4: Resource Distribution of the three forage tasks.

4.6 Controller

4.6.1 Heuristics

We allowed robots to 'attach' to a resource using a specialised heuristic pickup sensor located at the front of the body. Each uncollected resource had only one side that robots could attach to, which was set as the side approached by the first robot to collide with the resource. Once attached, the bots were heuristically guided to the target area. The robots were made to detach from the resource after it was pushed entirely into the the target area, upon which control was returned to the neural network. This was designed in order to avoid the potential deadlock situation in which robots would push the same resource with equivalent opposing forces, resulting in a deadlock.

In addition to the underlying neural network, the robots were also equipped with a hierarchical, conditional schedule of heuristics for the following tasks:

1. Collision avoidance of walls
2. Navigation to sticky side of resource
3. Navigation to target area after attaching to a resource

The ANN provided the actuator outputs to the robot only if the robot's state did not satisfy any of the conditions of the heurstics on the schedule. By the end of the development phase, only the heuristic for 3 remained. Visualisation of results showed that for most sensor complements, the robots developed rudimentary collision-avoidance behaviour, turning appropriately when faced with a wall. (See Section 5).

4.7 Controller Evolution

4.7.1 Encog's NEAT Implementation

The Encog Machine Learning Library is a collaborative, Open-Source Java framework by Jeff Heaton.(Heaton 2010) The NEAT implementation provided by the Encog Library was used as a basis for the controller evolution. The simulation platform was plugged into

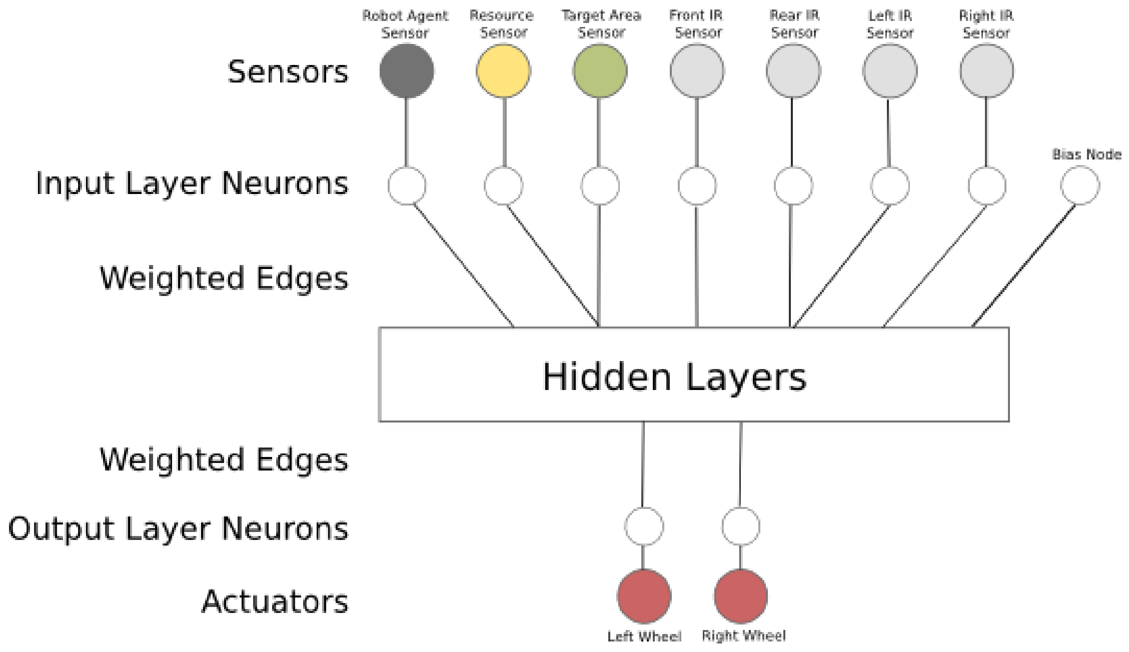


Figure 4.3: The Neural Network Architecture for the robots. The sensors each process a signal, which is passed to the Neural Network. The activation values provided by the input neurons is given to the hidden layers of the network evolved by the NEAT algorithm. The output nodes translate the values from the network into actuator values for the left and right wheels of the robot.

the fitness evaluation system used by the library. Additionally, the source code from the library was altered to adjust the evolutionary parameters and rewritten to implement the heterogeneous teaming strategy described in Section 3.1.2.

By default, Encog's NEAT Implementation uses four activation cycles per input vector. This is necessary because the implementation allows recurrent networks to form, and the additional activation cycles allow the post-activation values of the network to converge over cycles. However, since the network is evaluated once per timestep in the simulation, it is not clear what advantage recurrent networks may offer over non-recurrent networks in our simulation setup.

Encog has an elitism rate of 30% for each species unless the species contains fewer than 6 genotypes, in which case all individuals are placed in the elite pool. Additionally, it chooses randomly from the elite pool of each species to reproduce, either sexually or asexually, until the species is populated with N'_j individuals. A roulette-selection method is used to choose the reproduction method for each parent, with the following probabilities:

Operation	**Probability**
Crossover Two genotypes	0.5
Weight Mutation	0.475
Add Node	0.01
Add Link	0.01
Remove Link	0.005

Table 4.5: NEAT Operator Probabilities used to evolve the controllers.

The weight mutation operator may choose to perturb up to three links at a time, or some number proportional to 2% of the number of links in the network. The activation function used for all neurons was the steepend sigmoid, shaped according to Equation **??**:

Chapter 5

Results and Conclusion

Because of the large number of tests that needed to be run and the asynchronous manner in which the morphologies were tested across different cluster computing platforms, not all morphologies could be tested the recommended 10 times for each task setup. In order to reach a compromise between robustness and breadth of morphologies tested, the tests for some morphologies were discarded and the morphologies with numbers of runs constant within the task setup were analysed.

To answer Research Question 1, we compared the task performance of sensory-distorted robots with the control case, which was a robot with all three typed sensor parameters set to 1.0. This was done for all six task setups. The best-performing experiment morphology was calculated by comparing the average performance of all the tested morphologies over the runs for that task setup. This was then compared to the control morphology.

From the limited number of runs procured for the experimental morphologies, it was shown that the best experiment morphology consistently outperformed the control (See Figure 5).

Due to the necessity of altering EA parameters for each task setup, it was not possible to test Research Question 2 and Research Question 3 rigorously. Furthermore, in the absence of such adjustments, it would nevertheless have been difficult to control for the differences in evaluation algorithms between the homogeneous and the heterogoenous setups.

In conclusion, we believe the scope of this research was overly ambitious, giving consideration to the search space and time constraints involved in the completion of this project.

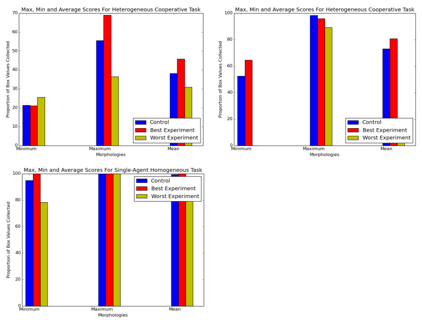


Figure 5.1: Bar charts showing the best, worst and average performance of the best experiment morphology, the worst experiment morphology, and the control morphology. In some cases the controller scored the overall best score, but the experiment morphology still had a greater average performance over the runs.

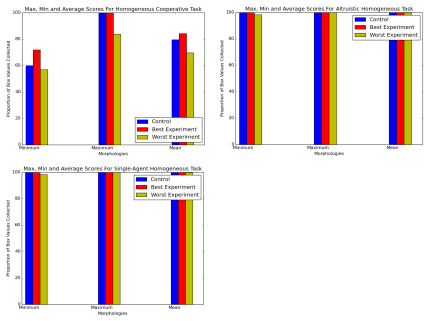


Figure 5.2: Bar charts showing the best, worst and average performance of the best experiment morphology, the worst experiment morphology, and the control morphology. As in the heterogeneous teams, some cases the controller scored the overall best score, but the experiment morphology still had a greater average performance over the runs.

	Sensor Gain Complement	Highest Score
Heterogeneous Single-Agent: Control	1.0 1.0 1.0	99.4047619048
Heterogeneos Single-Agent: Best Experiment	1.0 1.5 0.5	100.0
Heterogeneous Altruistic: Control	1.0 1.0 1.0	73.0
Heterogeneous Altruistic: Best Experiment	0.5 1.5 0.5	80.8
Heterogeneous Cooperative: Control	1.0 1.0 1.0	38.203125
Heterogeneous Cooperative: Best Experiment	0.5 0.5 0.5	45.7222222222
Homogeneous Single-Agent: Control	1.0 1.0 1.0	100.0
Homogeneous Single-Agent: Best Experiment	1.5 1.0 1.5	100.0
Homogeneous Altruistic: Control	1.0 1.0 1.0	100.0
Homogeneous Altruistic: Best Experiment	1.5 1.0 1.5	100.0
Homogeneous Cooperative: Control	1.0 1.0 1.0	79.6428571429
Homogeneous Cooperative: Best Experiment	1.5 1.0 1.5	84.2857142857

Table 5.1: Performance of control morphologies vs. experiment morphologies. The best-performing experiment morphology performed consistently as good or better than the control. A steep performance dropoff was noted in both setups as the task increased in cooperative requirements. However, homogeneous teams still performed better than heterogeneous teams on all three tasks.

Chapter 6

Further Work

The preliminary results for the experiments showed that it may be promising to explore signal processing as a way to aid controller evolution. The work in this research project used an exhausive search of sensor complements in order to find ones that could aid the evolution of cooperative behaviour. Because this search space is of a combinatoric size, testing all complements has been demonstrated to be costly and time-consuming. It may be possible to apply EAs to the search for a good set of sensor gains.

More rigorous statistical testing needed to be done in order to determine whether or not the experimental morphologies truly performed better than the controls. The Shapiro-Wilk test could be used to test against the normal distribution and an appropriate test selected to fit the parametric properties of the control and experiment samples for each task. The individual vs. team fitness needed to be examined at more depth. For heterogeneous altruistic tasks, it was suspected from the results that there was in inversely proportional relationship between the best individual's performance and the overall performance in the team it was in. Additionally, for the tasks requiring no cooperation, a better measure of performance could be devised involving the number of epochs required to collect all boxes in all team trials.

The sensor gain ranges were between 0.5 and 1.5. It would be of interest to explore whether or not allowing zero values (ie. a deactivated sensor) could simplify the morphology of the robot by removing unnecessary sensors. However, this may be more relevant to the domain of EHW.

Bibliography

Balch, Tucker (1997). "Learning Roles: Behavioral Diversity in Robot Teams". In: AAAI, pp. 7–12.

— (1999). "Reward and Diversity in Multirobot Foraging". In: *In IJCAI-99 Workshop on Agents Learning About, From and With other Agents.*

Bjørkøy, Olav (2011). "Competitive and Cooperative Behavior in Bio-Inspired AI". In:

Boers, Egbert J.W. and Herman Kuiper (1992). *Biological Metaphors and the Design of Modular Artificial Neural Networks.*

Bongard, Josh C. (2009). "The Impact of Jointly Evolving Robot Morphology and Control on Adaptation Rate". In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation.* GECCO '09. Montreal, Qu&#233;bec, Canada: ACM, pp. 1769–1770. ISBN: 978-1-60558-325-9.

Brooks, Rodney A. (1992). "Artificial Life and Real Robots". In: *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life.* Ed. by Francisco J. Varela and Paul Bourgine. Cambridge, MA, USA: MIT Press, pp. 3–10.

Bugajska, Magdalena D. and Alan C. Schultz (2002). "Coevolution of Form and Function in the Design of Micro Air Vehicles". In: *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02).* EH '02. Washington, DC, USA: IEEE Computer Society, pp. 154–. ISBN: 0-7695-1718-8.

Carse, Brian, Terence C. Fogarty, and Alistair Munro (1996). "Evolving Fuzzy Rule Based Controllers Using Genetic Algorithms". In: *Fuzzy Sets Syst.* 80.3, pp. 273–293. ISSN: 0165-0114.

Castelfranchi, Cristiano (2000). "Through the agents' minds: Cognitive mediators of social action". English. In: *Mind & Society* 1.1, pp. 109–140. ISSN: 1593-7879.

Catto, Erin (2014). *Box2D: A 2D Physics Engine for Games.*

Cliff, Dave, Phil Husbands, and Inman Harvey (1993). "Explorations in Evolutionary Robotics." In: *Adaptive Behaviour* 2.1, pp. 73–110.

Crawford, Elisabeth and Manuela Veloso (2005). "Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling". In: *In the working notes of the Multiagent Learning Workshop (to appear), AAAI.*

Darwin, Charles (1859). *On the origin of species by means of natural selection; or, The preservation of favoured races in the struggle for life / by Charles Darwin.* http://www.biodiversitylibrary.org/bibli New York :D. Appleton and Co., p. 474.

DePauli, J.F. (1989). *Infrared sensor signal conditioner.* US Patent 4,851,681.

Doran, J. E. et al. (1997). “On Cooperation in Multi-agent Systems”. In: *Knowl. Eng. Rev.* 12.3, pp. 309–314. ISSN: 0269-8889.

Dorigo, Marco and Uwe Schnepf (1993). “Genetics-based machine learning and behavior-based robotics: a new synthesis.” In: *IEEE Transactions on Systems, Man, and Cybernetics* 23.1, pp. 141–154.

Duch, Wlodzislaw and Norbert Jankowski (1999). “Survey of Neural Transfer Functions”. In: *Neural Computing Surveys* 2, pp. 163–213.

Durfee, Edmund and Victor R. Lesser (1991). “Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 21, pp. 1167–1183.

Eiben, Agoston E. and J. E. Smith (2003). *Introduction to Evolutionary Computing.* Springer Berlin Heidelberg.

Ficici, Sevan G., Richard A. Watson, and Jordan B. Pollack (1999). “Embodied Evolution: A Response to Challenges in Evolutionary Robotics”. In: *Eighth European Workshop on Learning Robots*, pp. 14–22.

Fogel, Lawrence J. (1999). *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming.* New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0-471-33250-X.

Franklin, Stan and Art Graesser (1997). “Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents”. In: *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages.* ECAI ’96. London, UK, UK: Springer-Verlag, pp. 21–35. ISBN: 3-540-62507-0.

Goldberg, Dani and Maja J Mataric (2001). “Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks”. In: *Robot Teams: From Diversity to Polymorphism.* A K Peters Ltd, pp. 315–344.

Gomes, Jorge, Paulo Urbano, and AndersLyhne Christensen (2013). “Evolution of swarm robotics systems with novelty search”. English. In: *Swarm Intelligence* 7.2-3, pp. 115–144. ISSN: 1935-3812.

Gomez, Faustino John (2003). “Robust Non-linear Control Through Neuroevolution”. AAI3116311. PhD thesis.

Grefenstette, John (1987). “The Evolution of Strategies for Multi-agent Environments”. In: *Adaptive Behavior* 1, pp. 65–90.

Gruau, F. (1992). “Genetic synthesis of Boolean neural networks with a cell rewriting developmental process”. In: *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pp. 55–74.

— (1993). “Cellular encoding as a graph grammar”. In: *Grammatical Inference: Theory, Applications and Alternatives, IEE Colloquium on*, pp. 17/1–1710.

Gruau et al. (1994). *Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm.*

Hansen, Nikolaus and Andreas Ostermeier (2001). “Completely Derandomized Self-Adaptation in Evolution Strategies”. In: *Evolutionary Computation* 9, pp. 159–195.

Harp, Steven Alex, Tariq Samad, and Aloke Guha (1989). “Towards the Genetic Synthesis of Neural Network”. In: *Proceedings of the Third International Conference on Genetic Algorithms.* George Mason University, USA: Morgan Kaufmann Publishers Inc., pp. 360–369. ISBN: 1-55860-006-3.

Harvey, Inman et al. (1997). “Evolutionary Robotics: the Sussex Approach”. In: *Robotics and Autonomous Systems* 20, pp. 205–224.

Heaton, Jeff (2010). *Programming Neural Networks with Encog 2 in Java*. Heaton Research, Inc. ISBN: 1604390077, 9781604390070.

Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*. second edition, 1992. Ann Arbor, MI: University of Michigan Press.

Holland, Owen and Chris Melhuish (1999). "Stigmergy, Self-organization, and Sorting in Collective Robotics". In: *Artif. Life* 5.2, pp. 173–202. ISSN: 1064-5462.

Jacob, Christian and Jan Rehder (1993). "Evolution of neural net architectures by a hierarchical grammar-based genetic system". English. In: *Artificial Neural Nets and Genetic Algorithms*. Ed. by RudolfF. Albrecht, ColinR. Reeves, and NigelC. Steele. Springer Vienna, pp. 72–79. ISBN: 978-3-211-82459-7.

Kassahun, Yohannes et al. (2009). "Incremental Acquisition of Neural Structures through Evolution". In: *Design and Control of Intelligent Robotic Systems*. Ed. by Dikai Liu, Lingfeng Wang, and KayChen Tan. Vol. 177. Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 187–208. ISBN: 978-3-540-89932-7.

Kicinger, Rafal, Tomasz Arciszewski, and Kenneth De Jong (2005). "Evolutionary Computation and Structural Design: A Survey of the State-of-the-art". In: *Comput. Struct.* 83.23-24, pp. 1943–1978. ISSN: 0045-7949.

Kitano, H. (1990). "Designing neural networks using genetic algorithms with graph generation system". In: *Complex Systems* 4, pp. 461–476.

Knack, Stephen (1998). "Mark Irving Lichbach, The cooperator's dilemma. Ann Arbor: University of Michigan Press, 1996." English. In: *Public Choice* 97.1-2, pp. 209–212.

Koehn, Philipp (1994). *Combining Genetic Algorithms and Neural Networks: The Encoding Problem*.

König, Lukas, Sanaz Mostaghim, and Hartmut Schmeck (2009). "Online and Onboard Evolution of Robotic Behavior Using Finite State Machines". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '09. Budapest, Hungary: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1325–1326. ISBN: 978-0-9817381-7-8.

Koza, JohnR. (1994). "Genetic programming as a means for programming computers by natural selection". English. In: *Statistics and Computing* 4.2, pp. 87–112. ISSN: 0960-3174.

Lambercy, F. and J. Tharin (2013). *Khepera III User Manual*. Version 3.5. K-Team Corporation.

Lindenmayer, Aristid (1968). "Mathematical models for cellular interaction in development: Parts I and II." In: *Journal of Theoretical Biology* 18.

Luke, Sean, Claudio Cioffi-Revilla, et al. (2005). "MASON: A Multiagent Simulation Environment". In: *Simulation* 81.7, pp. 517–527. ISSN: 0037-5497.

Luke, Sean and Lee Spector (1996). "Evolving Teamwork and Coordination with Genetic Programming". In: *Proceedings of the 1st Annual Conference on Genetic Programming*. Stanford, California: MIT Press, pp. 150–156. ISBN: 0-262-61127-9.

Lund, H.H., J. Hallam, and Wei-Po Lee (1997). "Evolving robot morphology". In: *Evolutionary Computation, 1997., IEEE International Conference on*, pp. 197–202.

Mahfoud, Samir W. (1995). *Niching Methods for Genetic Algorithms*. Tech. rep.

Mandischer, Martin (1993). "Representation and Evolution of Neural Networks". In: *Artificial Neural Nets and Genetic Algorithms*. Ed. by R. F. Albrecht, C. R. Reeves, and U. C. Steele. Berlin; New York: Springer-Verlag, pp. 643–649.

Martín, JoséAntonioH., Javier de Lope, and Matilde Santos (2007). "Evolution of Neurocontrollers for Multi-link Robots". English. In: *Innovations in Hybrid Intelligent Systems*.

Ed. by Emilio Corchado, JuanM. Corchado, and Ajith Abraham. Vol. 44. Advances in Soft Computing. Springer Berlin Heidelberg, pp. 175–182. ISBN: 978-3-540-74971-4.

Matarić, MajaJ. (1996). "Learning in multi-robot systems". English. In: *Adaption and Learning in Multi-Agent Systems.* Ed. by Gerhard Weiß and Sandip Sen. Vol. 1042. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 152–163. ISBN: 978-3-540-60923-0.

Matsumoto, Makoto and Takuji Nishimura (1998). "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator". In: *ACM Trans. Model. Comput. Simul.* 8.1, pp. 3–30. ISSN: 1049-3301.

Mautner, Craig and Richard K. Belew (1999). "Evolving Robot Morphology and Control". In: *In 4th International Symposium on Artificial Life and Robotics. REFERENCES 31 Miglino.*

McCulloch, Warren S and Walter Pitts (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133.

Michie, D., D. J. Spiegelhalter, and C.C. Taylor (1994). *Machine Learning, Neural and Statistical Classification.*

Miikkulainen, Risto (2010). "Neuroevolution". In: *Encyclopedia of Machine Learning.* Springer, pp. 716–720.

Miikkulainen, Risto et al. (2012). "Multiagent Learning Through Neuroevolution". In: *Proceedings of the 2012 World Congress Conference on Advances in Computational Intelligence.* WCCI'12. Brisbane, Australia: Springer-Verlag, pp. 24–46. ISBN: 978-3-642-30686-0.

Montana, David J. and Lawrence Davis (1988). *Training Feedforward Neural Networks Using Genetic Algorithms.* Tech. rep. Cambridge, MA: BBN Systems and Technologies, Inc.

Moriguchi, Hirotaka and Shinichi Honiden (2012). "CMA-TWEANN: Efficient Optimization of Neural Networks via Self-adaptation and Seamless Augmentation". In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation.* GECCO '12. Philadelphia, Pennsylvania, USA: ACM, pp. 903–910. ISBN: 978-1-4503-1177-9.

Mouret, J. -B. and S. Doncieux (2012). "Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study". In: *Evol. Comput.* 20.1, pp. 91–133. ISSN: 1063-6560.

Murphy, Daniel (2014). *JBox2D: A Java Physics Engine.*

Nitschke, Geoff (2011). "Neuro-Evolution Methods for Gathering and Collective Construction". In: *Advances in Artificial Life. Darwin Meets von Neumann.* Ed. by George Kampis, István Karsai, and Eörs Szathmáry. Vol. 5777. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 115–123. ISBN: 978-3-642-21282-6.

Nolfi, Stefano and Dario Floreano (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, MA, USA: MIT Press. ISBN: 0262140705.

— (2002). *Synthesis of Autonomous Robots Through Evolution.*

Nolfi, Stefano, Dario Floreano, et al. (1994). "How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics". In: *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV).* Ed. by Rodney A. Brooks and Pattie Maes. Cambridge, MA: MIT Press.

Ostergaard, Esben H., Gaurav S. Sukhatme, and Maja J. Matari (2001). "Emergent Bucket Brigading: A Simple Mechanisms for Improving Performance in Multi-robot Constrained-space Foraging Tasks". In: *Proceedings of the Fifth International Conference on Autonomous Agents.* AGENTS '01. Montreal, Quebec, Canada: ACM, pp. 29–30. ISBN: 1-58113-326-X.

Panait, Liviu and Sean Luke (2006). "Cooperative Multi-Agent Learning: The State of the Art." In: *Autonomous Agents and Multi-Agent Systems* 11.3, pp. 387–434.

Parker, M. and B.D. Bryant (2009). "Lamarckian neuroevolution for visual control in the Quake II environment". In: *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pp. 2630–2637.

Polikarpova, N.I., V.N. Tochilin, and A.A. Shalyto (2010). "Method of reduced tables for generation of automata with a large number of input variables based on genetic programming". English. In: *Journal of Computer and Systems Sciences International* 49.2, pp. 265–282. ISSN: 1064-2307.

Rajagopalan, Padmini et al. (2011). "The Role of Reward Structure, Coordination Mechanism and Net Return in the Evolution of Cooperation". In: *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011)*. Seoul, South Korea.

Rawal, Aditya et al. (2010). "Constructing competitive and cooperative agent behavior using coevolution". In: *in IEEE Conference on Computational Intelligence and Games (CIG '10*.

Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem.* Tech. rep. Royal Air Force Establishment.

Rumelhart, David E. et al. (1995). "Backpropagation". In: ed. by Yves Chauvin and David E. Rumelhart. Hillsdale, NJ, USA: L. Erlbaum Associates Inc. Chap. Backpropagation: The Basic Theory, pp. 1–34. ISBN: 0-8058-1259-8.

Russell, Stuart J. and Peter Norvig (2003). *Artificial Intelligence: A Modern Approach.* 2nd ed. Pearson Education.

Sánchez-Parra, Marino and René Vite-Hernández (2002). "Use of a Rule-Based System for Process Control: Flow Instabilities in Axial Compressors Case Study". In: *Proceedings of the Second Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence.* MICAI '02. London, UK, UK: Springer-Verlag, pp. 494–505. ISBN: 3-540-43475-5.

Sher, Gene I. (2011). "Evolving Chart Pattern Sensitive Neural Network Based Forex Trading Agents". In: *CoRR* abs/1111.5892.

Siciliano, Bruno and Oussama Khatib (2007). *Springer Handbook of Robotics.* Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN: 354023957X.

Sivanandam, S. N. and S. N. Deepa (2007). *Introduction to Genetic Algorithms.* 1st. Springer Publishing Company, Incorporated. ISBN: 354073189X, 9783540731894.

Sofge, Donald et al. (2007). "Challenges and Opportunities of Evolutionary Robotics". In: *CoRR* abs/0706.0457.

Specht, Donald F. (1991). "A general regression neural network". In: *Neural Networks, IEEE Transactions on* 2.6, pp. 568–576. ISSN: 1045-9227.

Stanley, Kenneth O. and Risto Miikkulainen (2002a). "Evolving Neural Network through Augmenting Topologies." In: *Evolutionary Computation* 10.2, pp. 99–127.

— (2002b). "Evolving Neural Networks Through Augmenting Topologies". In: *Evol. Comput.* 10.2, pp. 99–127. ISSN: 1063-6560.

— (2004). "Competitive Coevolution Through Evolutionary Complexification". In: *J. Artif. Int. Res.* 21.1, pp. 63–100. ISSN: 1076-9757.

Steeb, R. et al. (1988). "Distributed intelligence for air fleet control: Architectures for distributed air traffic control". In: *Readings in Distributed Artificial Intelligence* 101.3, pp. 90–101.

Steels, L. (1990). “Cooperation between distributed agents through self-organisation”. In: *Intelligent Robots and Systems ’90. ’Towards a New Frontier of Applications’, Proceedings. IROS ’90. IEEE International Workshop on*, 8–14 supl.

Sutton, Richard S. and Andrew G. Barto (1998a). *Introduction to Reinforcement Learning.* 1st. Cambridge, MA, USA: MIT Press. ISBN: 0262193981.

— (1998b). *Reinforcement Learning: An Introduction.* MIT Press. ISBN: 0262193981.

Waibel, Markus, Laurent Keller, and Dario Floreano (2009). “Genetic Team Composition and Level of Selection in the Evolution of Cooperation”. In: *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* 13.3, pp. 648–660.

Watkins, C. (1989). “Learning from Delayed Rewards”. PhD thesis. University of Cambridge,England.

Watson, Richard A., Sevan G. Ficici, and Jordan B. Pollack (2002). “Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots”. In: *Robotics and Autonomous Systems* 39, pp. 1–18.

White, David W. (1994). “GANNet: A genetic algorithm for searching topology and weight spaces in neural network design”. PhD thesis. University of Maryland College Park.

Whitley, Darrell (1989). “The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best”. In: *Proceedings of the Third International Conference on Genetic Algorithms.* Ed. by J. D. Schaffer. San Mateo, CA: Morgan Kaufman.

Widrow, B. and M.A. Lehr (1990). “30 years of adaptive neural networks: perceptron, Madaline, and backpropagation”. In: *Proceedings of the IEEE* 78.9, pp. 1415–1442. ISSN: 0018-9219.

Wolfram—Alpha (2014). *Plot Sigmoid. From MathWorld—A Wolfram Web Resource.*

Yao, Xin (1999). “Evolving artificial neural networks”. In: *Proceedings of the IEEE* 87.9, pp. 1423–1447. ISSN: 0018-9219.