Honours Literature Review: Adaptive Sensory Configuration in Robot Teams

Ntokozo Phillip Zwane, University of Cape Town

There has been an increase in work aimed at using robot teams to complete a range of non-trivial collective behaviour tasks. The task we will pay particular attention to is a search and retrieval (foraging) task. We will construct the task such that the team of robots would have to make use of cooperative behaviour, which promotes cooperation between the individual robots. In the real world, such a task would involve a significant amount of noise and incomplete information which would require the team of robots to adapt to the environment through a suitable learning process. We would thus expect our team of robots to exhibit some form of Artificial Intelligence (AI) [Russell and Norvig 2003]. The concepts of Evolutionary Computation (EC) are used to achieve this goal, which builds the foundations of Evolutionary Robotics (ER).

EC covers a range of methods that create control systems that exhibit a desired behaviour, and for our particular task, Neuro-Evolution (NE) is a suitable EC method which has been shown to be high performing in control tasks of this nature.

Historically, there has been large focus on agent controller evolution, and little work done to explore the evolution of the sensor configuration (*morphology*) of the agents. We will simulate the behaviour of a team of robots with adapting morphologies for each robot's static, pre-defined controller.

1. AGENT CONTROLLER DESIGN

There are various approaches to the problem, and one that has shown promise is modelling the biological central nervous systems, thus employing their behavioural and adaptive features. These computational models are called Artificial Neural Networks (ANNs). They are compromised typically of several interconnected process units (*neurons*) which have a given number of inputs and outputs.

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

The learning methods used for an ANN control model are based on a subset of the Darwinian evolution principles, called Evolutionary Algorithms (EAs) [Back 1996]. This subset that revolves around the evolution of ANNs is called Neuro-Evolution(NE). Useful algorithms or algorithm parameters are found by iteratively selecting and refining a population of possible solutions.

The foraging task that the team of robots will complete will require cooperation between the robots, since some of the resources to be collected will require multiple robots to return it to the collection area. This type cooperation between multiple robots is called a *multi-agent system* (MAS) [Panait and Luke 2005].

2. ARTIFICIAL INTELLIGENCE

2.1. Machine Learning

The field of AI tries to understand intelligent entities and build them based on the level of understanding reached. There are multiple definitions of what AI is, [Russell and Norvig 2003] have grouped some of these into four categories: systems that think like humans; systems that act like humans; systems that think rationally; systems that act rationally.

Alan Turing proposed The Turing Test [Russell and Norvig 2003], which roughly tested intelligence based on the intelligent entity's ability to achieve human-level cognitive tasks, good enough to fool an interrogator. On of the capabilities the intelligent entity would require in order to pass the test is Machine Learning (ML), in order to be able to adapt new circumstances and fit patterns existing knowledge.

ML is a subfield of AI that is concerned with programs that learn from experience. ML is applied to the design of controllers where the task is too complex for human-designed procedures or a set of heuristics.

2.2. Learning Methods

The different approaches to learning are divided into three main categories: supervised learning; unsupervised learning and reward-based learning. The categories are defined by the type of feedback given to the learning algorithm.

Algorithms that are trained using supervised learning are given sets of input and the corresponding correct output data, and each iteration attempts to find a function that closely represents the relationship between the input and output value. Unsupervised learning is the exact opposite, where no output values given, the algorithm is trained with no feedback. The previous percepts are used to predict the future percepts. Supervised learning is generally not used to train algorithms used for multi-agent systems due to the large sets of possible output data.

Reward-based learning can be further divided into two sub-categories: reinforcement learning and stochastic search methods. Algorithms trained with reinforcement learning approximate the output function values using previous estimates, and are given feedback in the form of a reward or penalty depending on how they perform in the environment. Reward-based learning relies on the principles of dynamic programming to estimate the value (*utility*) resulting in taking a certain action based on the current control parameters (*states*). Using reinforcement learning to train algorithms that control MASs is difficult because a large state space is created by the interactions between the agents, this it becomes hard to map states to utilities, however, this approach is not obsolete.

Stochastic search methods do not estimate value functions, they instead learn behaviours directly. A strategy that has been found to be effective in generating control parameters and decision rules for robots display collective behaviour is Evolutionary Computing (EC) [Eiben and Smith 2003]

2.3. Evolutionary Algorithms

Evolutionary Algorithms are a subfield of EC and is aimed at adapting solutions based on the concepts of Darwinian evolution.

The Darwinian process of selection and survival of the fittest is applied to a set of candidate solutions (*populations* or *individuals*) by refining them through processes of selection, recombination and mutation at each iteration (*generation*) of the algorithm. These processes are known as genetic operators, the populations will be refined on each iteration until a termination condition is met, which is usually when either an individual that fits the criteria is found or a certain amount of time has elapsed.

Evolutionary Algorithms are divided into several classes, which are categorized by the data structure used to represent the solutions. Genetic Algorithms (GAs) [Goldberg and Holland 1988]; Evolutionary Strategies (ESs) [Eigen 1973] Genetic Programming (GP) [Koza 1992] and Evolutionary Programming (EP) [Fogel et al., 1966][Fogel et al. 1990].

GAs and ESs are generally used for searching multidimensional space, GAs represent their individuals with using a bit-string, and ESs a real-valued vector. GPs evolve computer programs by refining the expression trees representing them, and EPs evolve the state tables for finite state machines (FSMs) - machines with state tables that can be written as a truth table to represent the transitions between states.

3. NEURO-EVOLUTION

3.1. Artificial Neural Networks

The inner workings of the biological brains in nature are the fundamental building blocks for artificial neural networks [Nolfi and Floreano 2000]. ANNs are computational models written for software or specialized hardware devices in attempt to replicate the behavioural and adaptive features of biological nervous systems.

The ANN structure is composed of multiple interconnected units (*neurons* or *nodes*) which have some input and output links (which can be thought of as synapses between neurons in a brain). In essence, each node implements a neuron model. In the simplest case, the neuron model is a sum of the weights of the incoming signals, transformed by some static *transfer function* - typically a nonlinear function. Factors like discrete-time or continuous time dynamics are considered for more sophisticated neuron models.

Given that an ANN can be represented as a directed graph, Section 1, the strengths of the connections associated with the edges of the graphs connecting two neurons are called *synaptic weights*, the neurons that with connections to the external environment are called *input* or *output neurons* (see Fig. 1), any neuron without any connections to the external environment are called *hidden neurons*. The *architecture* or *topology* of the neural network is defined by the number and type of neurons and the set of possible interconnections between them.

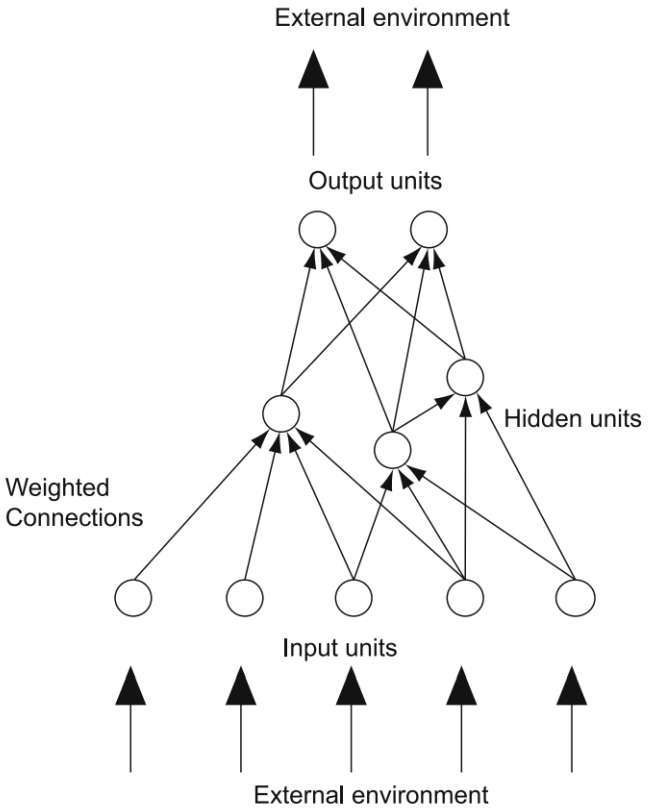

Fig. 1. A generic neural network architecture. It consists of input and output nodes which are connected to the external environment and Hidden units which connect to other neurons but are not directly connected to the environment [Dario et al., 2008]

Each neuron in the ANN has one or more inputs, and a single output. The output sends a signal to another neuron and its value depends on the value or values of the inputs. Each node has weight assigned to each input, and a weighted sum is calculated using each of these values with the *transfer function*,

$$net = \sum_{i \in I} w_i x_i \tag{1}$$

where w_i is the weight for the ith input, and x_i is the ith input.

The *activation function* is then used to calculate the new value of the current node's weight. The result of the transfer function is used in the activation function,

$$\varphi(t) = \frac{1}{1 + e^{-\mu t}} \tag{2}$$

where t is the weighted sum from the transfer function, ans μ is the slope constant.

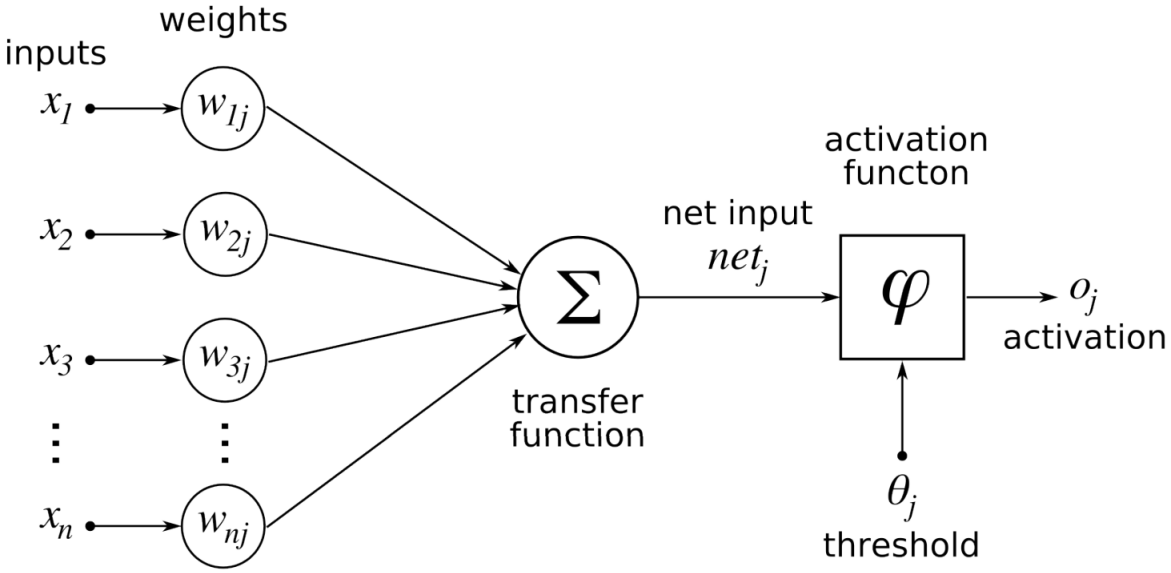

Fig. 2.

Simple ANNs are networks constructed with input nodes that feed information directly to output nodes, more sophisticated networks contain additional nodes between the input and output nodes, these nodes (that have no direct connection with the external environment) are called *hidden nodes*, and they create a *hidden layer* in the network. Networks that contain at least one hidden layer are called *multi-layer networks*

There are various ways in which the connections within a network are structured. This is the notion of *connectedness*. A networks is said to be *fully-connected* if each node in a given layer is connected to every node in the next layer. If a node is connected to any other node within the same layer, or a previous layer, we call this a *recurrent connection*.

Fully-connected networks hold a special characteristic. [Hornik 1991] showed that a fully-connected network with just a single hidden layer is a *universal function approximator* (see Fig. 3), which means it is able to approximate any continuous function. This behaviour is particularly useful a wide range of problems, where the approximation of complex control functions that map a set of sensor inputs to a set of control outputs is required.

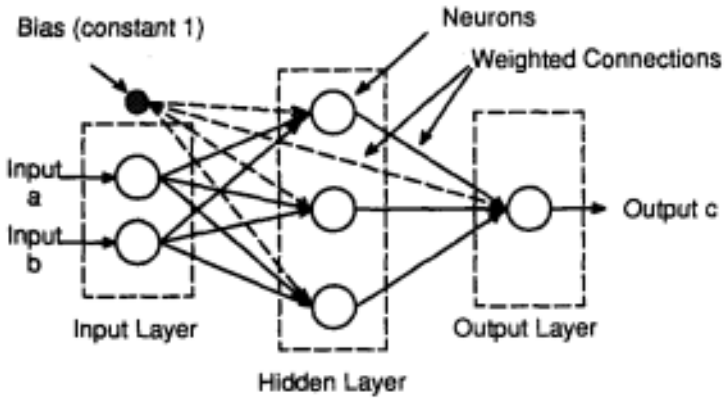

Fig. 3. A fully connected artificial neuron network, with a bias node [Chen et al. 1992].

3.2. Conventional Neuroevolution

There are various 'flavours' of Neuroevolution. One of the most simple, and the basis model for these different approaches to Neuroevolution is Conventional Neuroevolution (CNE). This is an extension of the traditional single population Neuroevolution. The population is made up of complete neural networks, and the weights of each network are adapted after each evolutionary step. How this differs from the traditional approach is that the weights are real numbers, as opposed to binary numbers.

3.3. NeauroEovlution of Augmenting Topologies

Stanley and Miikkulainen [2002] proposed that Neuroevolution of Augmenting Topologies (NEAT) outperforms the best fixed-topology method on a challenging benchmark reinforcement learning task. They claim that the increase in efficiency lies on the fact that NEAT takes on a principled method of crossover of different topologies; making use of *speciation* to protect structural innovation; and incrementally growing from a primitive structure. NEAT is an important GA because it shows how solutions can be optimized and complexified simultaneously through evolution, which strengthens the analogy between EAs and natural evolution.

NEAT aims to address three main challenges: whether or not there is a genetic representation that allows crossover of desperate topologies in a meaningful way; how topological innovation that require only a few generations to be optimum can be protected, so that it does not disappear from the population too soon; and how topologies can be minimized throughout evolution, without the need of a measure of fitness through a special fitness function.

NEAT is able to find solutions faster than other Neuroevolution methods because of the fact that small structure optimise faster, and it incrementally refines structure in a stochastic manner from a minimal starting point.

3.4. Symbiotic Adaptive Neuroevolution

Symbiotic adaptive Neuroevolution (SANE) differs from traditional Neuroevolution approaches in that the population of neurons is evolved (see Fig. 4), as opposed to a complete neural network [Dawson et al. 2006]. Since no single individual represents a solution - individuals need to be grouped together to form a solution - each neuron is tasked with creating connections with other neurons in the population, to construct an

operational neural network [Dawson et al. 2006]. This approach promotes higher operational specialization in the final network, which, coupled with the resulting increase in diversity, aides in convergence towards a global optimum, rather than possible sub-optimal solutions.

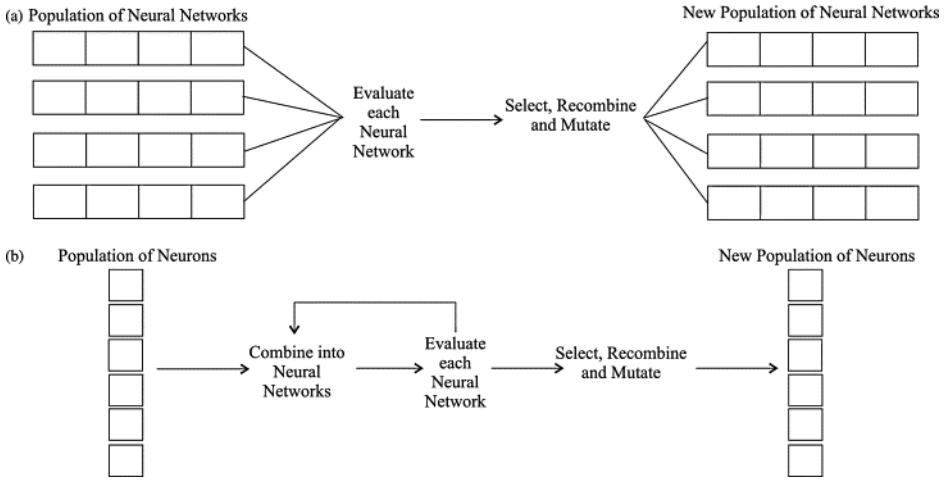

Fig. 4. A comparison of the traditional Neuroevolutionary approach (a) and the SANE approach (b) to evolve neural networks [Dawson et al. 2006].

3.5. MESP

3.6. my section stuffs

4. AGENT BASED AND MULTI-AGENT SYSTEMS

Ferber [1999] defines an agent by its ability to act, perceive its environment and communicate with other agents. It is autonomous and is equipped with skills to reach its goals or exhibit a certain behaviour. Ferber [1999] further states that changes to the universe is only caused by time or actions that are taken by the agents in the system. This universe contains an environment, objects and the agents that act in it.

Multi-Agent Systems (MAS) - a subsection of agent based systems - can be built in various ways, two of which being with the use of cognitive agents, or reactive agents. Reactive agents have reflexes, they act based on their encounters with their environment, they are simple agents whose co-ordination emerges in time; whereas cognitive agents construct predefined plans for their actions. They are initially intelligent individuals with organised communication.

Achieving cooperation in MAS and ABS, as an effective problem solver, is challenging with many approaches. Given the wide application of MASs, it is difficult to define what exactly *co-operation* is, and how an observer can go about detecting or measuring co-operation in a system. Ferber [1999] has attempted to give an objective definition of co-operation, one that does not depend on the kind of MAS, whether co-operation occurs intentionally or not. Ferber [1999] states that co-operation among agents exists if the addition of new agents improves the system's situation, or potential conflicts are mitigated. In contrast with how Panait and Luke [2005] define co-operation loosely in

terms of the experimenter's *intent*.

Secondly, multi-agent learning may involve *multiple learners*, each learning and adapting with respect to the others - an added level of complexity is introduced (as it naturally comes when using evolutionary controller adaptation approaches) - and the team behaviours that evolve (emerge) cannot always be predicted.

4.1. Multi-agent Learning

The application of machine learning to multiple agents is known as *multi-agent learning* [Panait and Luke 2005]. Multi-agent learning is considered as a separate field from machine learning [Panait and Luke 2005] due to the fact that, firstly, the search space can be very large - because the problem domain involves multiple agents; small changes can result in unpredictable changes in the characteristics of the learned behaviour due to the interaction of the multiple agents. Panait and Luke [2005] ignore the class of MAS in which a single agent learns and the other agents' behaviours are unchanged. This is when the behaviours are evolved for only one agent, rather than for the entire agent distribution, and is known as *single-agent learning*.
Co-evolutionary algorithms(CEAs) Panait and Luke [2005] are a seamless approach to applying evolutionary computation to refine multi-agent behaviours. The interaction of an individual with other individuals determines its fitness, thus the fitness assessment is sensitive to the context, and subjective to each individual. In cooperative co-evolution, individuals succeed or fail together in collaboration. A standard approach to applying co-operative co-evolutionary algorithms (CCEAs) to an optimisation problem begins with a decomposition of the problem representation into subcomponents, then each subcomponent to a separate population of individuals [Panait and Luke 2005] - each individual is a *genotype*, a representation of an agent controller.

5. MORPHOLOGY EVOLUTION

EAs have been successfully used in the construction of artificial neuron networks for a variety of applications. One direct application of EAs is in the field of evolutionary robotics, where EAs are used to design controllers for the robots. These designs do work well in general, however, they have limited performance due to the number, placement, quality, efficiency and reliability of the sensors that robots are fitted with [Balakrishnan and Honavar 1996]. A similar approach to controller design should be applied to sensor configuration (placement and range) in order to produce robots of higher fitness.
It might not seem as though it would be beneficial to evolve the sensor configuration of the robots, however, Bongard [2011]'s approach to show this improvement is evident was to evaluate the effectiveness of two differently evolved organisms. One that had a controller and morphology evolved with an initial, legged, body; and another evolved with initially an anguilliform - the form of an eel - body plan. The results showed that those organisms that started off with the anguiliform evolved to more robust robots, as opposed to the other robots. This suggests that change in morphology is an important process in producing robust controllers [Bongard 2011]. It was also evident that the complexity of the bodies was influenced by the complexity of the environment the organisms were exposed to. To truly test the effectiveness of evolving the agents' bodies, Pfeifer [2002] the evolution of agent bodies for 10 different evolved agent controllers. It is evident that agents that had strong physical characteristics (eg. mass, number of legs etc.) of a certain type had large behavioural differences to agents with characteristics of a different type [Pfeifer 2002], which supports the idea of including morphological evolution in the agent training process. Balakrishnan and Honavar [1996] showed that far fewer sensors were used by the evolutionary system

than the number made available - this behaviour was observed without any penalties given for use of an excessive number of sensors. Balakrishnan and Honavar [1996] suggested that introducing sufficient noise to the system will decrease the chances of robots having low fitness, due to them getting stuck in repetitive or cyclic paths, and these designs can be successfully transferred on to real world robots.
The problem lies in attempting to redefine the sensor characteristics, as opposed to sensor positions and how many of each type is present. The sensor characteristics may be confined to the underlying architecture of the physical sensor, and its capabilities may be limited. We do, however still have full freedom to modify the positions, number and types of sensors on each iteration, the only constraint we will thus have to take into considerations with that respect are: the size of the robot (how many sensors can fit on it) (See Fig. **??**), and a reasonable level of power consumption.

Sensory system design allows us to determine efficient, often counter-intuitive designs for the control mechanisms, and this EA-based design can easily be extended further to cope with different types of sensors. Balakrishnan and Honavar [1996] suggests further that sensory system design is necessary in designing robots that can handle hazardous and unpredictable environments (e.g. nuclear waste clean-up; toxic chemical handling processes, space exploration etc.), where sensor failures might occur quite frequently.

The result of evolution of visual sensing morphologies and sensory-motor controller-networks for visually guided robots presents a number of networks that result from using incremental evolution with variable-length genotypes [Cli et al. 1993]. There are, however, two (of the many) networks - emerging from an evolution approach of this form - that display striking characteristics. The behavioural observations of these two networks are the same, however, the sensing morphologies and sensory-motor controllers are very different. Which means that this evolutionary process converges at a behavioural level, and is coupled with divergent evolution at the morphological level. The process of evolution used here holds many similarities to analysing biological nervous systems in the field of *neurothology*[Cli et al. 1993].

6. DISCUSSION

The task we are presented with is to test weather or not the evolution of the morphology of a team of robots results in any improvements in the system. We will simulate robots that will need to co-operatively complete a foraging task. The robot controllers will be pre-defined and kept fixed/ static throughout the evolution of the morphology. We will likely employ a reactive learning method [Ferber 1999] given the unpredictable nature of the environment that the robots will be exposed to, and the fact that an agent's actions is dependent on the interactions it has with the other agents.
Work on morphology evolution is still relatively new, however, it shows promise in taking the field of evolutionary robotics a step further, by obtaining robots whose controller-morphological design displays a behaviour with high fitness. Robots evolved in such a manner have shown promise also improve the power consumption in that they use the minimum required set of sensors for optimum fitness [Balakrishnan and Honavar 1996].
The method used to evolve the morphology should be able to cope with a larger scope of types of sensors, thus is favourable to our task.
The full effects and benefits/drawbacks of morphology evolution are yet to be fully understood, however, it is evident that it is an approach that is worth considering.

REFERENCES

Thomas Back. 1996. *Evolutionary algorithms in theory and practice*. Oxford Univ. Press.

Karthik Balakrishnan and Vasant Honavar. 1996. On Sensor Evolution in Robotics. In *Proceedings of the 1st Annual Conference on Genetic Programming*. MIT Press, Cambridge, MA, USA, 455–460. http://dl.acm.org/citation.cfm?id=1595536.1595616

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

S-T Chen, David C Yu, and Alireza R Moghaddamjo. 1992. Weather sensitive short-term load forecasting using nonfully connected artificial neural network. *Power Systems, IEEE Transactions on* 7, 3 (1992), 1098–1105.

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

Christian W Dawson, Linda M See, Robert J Abrahart, and Alison J Heppenstall. 2006. Symbiotic adaptive neuro-evolution applied to rainfall–runoff modelling in northern England. *Neural Networks* 19, 2 (2006), 236–247.

Agoston E Eiben and James E Smith. 2003. *Introduction to evolutionary computing*. Springer Science & Business Media.

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

Jacques Ferber. 1999. *Multi-agent systems: an introduction to distributed artificial intelligence*. Vol. 1. Addison-Wesley Reading.

David B Fogel, Lawrence J Fogel, and VW Porto. 1990. Evolving neural networks. *Biological cybernetics* 63, 6 (1990), 487–493.

David E Goldberg and John H Holland. 1988. Genetic algorithms and machine learning. *Machine learning* 3, 2 (1988), 95–99.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991), 251–257.

John R Koza. 1992. *Genetic programming: on the programming of computers by means of natural selection*. Vol. 1. MIT press.

Stefano Nolfi and Dario Floreano. 2000. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press.

Liviu Panait and Sean Luke. 2005. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems* 11, 3 (Nov. 2005), 48.

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

Russell and Norvig. 2003. A modern approach. *Intelligence, Artificial* (2003).

Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.